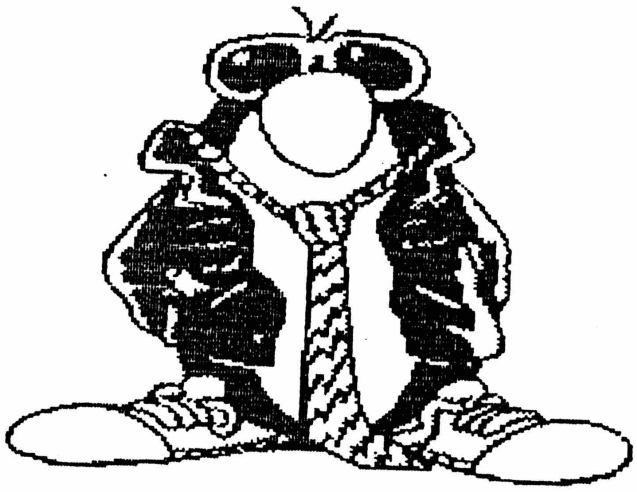
INTERNATIONAL TI-LINES

Volume 4, Issue 1

June 1st., 1987



ISN'T IT ODD
HOW YOU CAN
ALWAYS SPOT A
FELLOW TI USER ?

Formerly OXON TI USERS

A PUBLICATION OF THE INTERNATIONAL TI USER GROUP

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

OXFORD 510822

Index to articles Page

READ THIS FIRST !.....3

The renamed editorial gets off to a long-winded start

SORTING IN MACHINE CODE.....9

An object code file designed to order

SOFT WARE SHUFFLE.....10

Based on a letter from NORMAN HANCOCK

LOGO, BASIC, AND RECURSION.....14

BASIC can't do recursion, can it ?

DERBY REPORT.....17

On TI-EXCHANGE's last meeting

THE INFINITE MONKEYS THEORY.....19

Darwinian approach to pseudorandom text generation

XMODEM PROTOCOLS.....21

This month's major item is a discussion of XMODEM

CAPTION COMPETITION - RESULTS.....30

SCOTT and JO ANN COPELAND present the winners, runners-up, and a few rib-ticklers worthy of mention

ADVERTISEMENT.....31

4FRONT from NEW DAY COMPUTING

NOTICE BOARD.....32

TI bits for sale

READ THIS FIRST !

Hello and welcome to the fourth year (and 39th issue of TI-LINES...).

Evidence is accumulating from the returned questionnaires (and partly also from the TI-EXCHANGE meeting at Derby, of which more elsewhere), that a sizeable number of ITUGers either didn't read the Burble at all last year (they skipped to the "more interesting parts") or only read and inwardly digested part of it.

The editorial (lately called BALDIE'S BURBLE and now to be called READ THIS FIRST! to try and emphasize its role) contains information on a wide range of subjects: notifications of upcoming events, errors which have been detected in previous issues, requests for information, requests for ARTICLES, and other information which is of importance to ITUGers (even though they may not always realise it), but which does not come under the heading of "article" - for example, reminders, brief advice, and particularly important news from outside our own extremely small sphere of interest. I would be failing in my responsibility if I did not attempt to make readers aware that there is a whole Universe outside the TI-99/4A, and almost all of it concerns us; reading about tiny bits of it in editorials may ensure that you belong to the Knows, rather than to the Know-not/Care-not brigade.

On numerous occasions I have found myself explaining elementary things over the phone/in letters which I know I covered to the same depth in a Burble, and when I have ventured to suggest that the enquirer might find enlightenment in the Burble, it has often transpired that they have rarely, if ever, read the editorials. That's almost criminal (but not quite...).

I have had a few suggestions to the effect that, if space is tight (it usually is) I should cut or drop the editorial and squeeze in articles instead (which is fine if there is no editorial worth mentioning and if the articles fit the space exactly - Baldie's First Law dictates that no article fits the space exactly), but if I do drop the editorial, how am I supposed to pass on all sorts of information ? I could, I suppose, do away with the verbose and highly informal writing style and adopt something more terse and compact. It would make everything even harder to read and understand, and newcomers, or those not used to reading such reference-manual format material, would be virtually excluded entirely.

If you've had difficulty understanding TI's manuals, let me advise you that they have been written in a form more understandable than most!

At least only a few indicated that they DIDN'T find IT's articles generally understandable, which would have made matters even more complex - how do you simplify that which is already simplified ?

A few ITUGers have indicated a need for "linking articles" to fill in the gaps between Beginners and Boffins. As very few of either Beginners or Boffins are at all interested in writing articles, this task falls to me. However, in the past, when I have asked for specific details of subjects which ITUGers want explained, there has been no useful response

>>MORE>

and I have been reduced to telepathy to try and discover what causes headaches for some of you. (This lack of useful feedback is apparently a universal experience: I often see despairing pleas for articles or subjects in overseas magazines).

Much material was covered in earlier volumes of TI-LINES, and although I apologise for the lack of detailed index (I have had no time to prepare one, but new ITUGer PETER KILLICK is kindly going to try and provide a very detailed issue by issue index to volumes 1 and 2), I have pointed out to some recent subscribers that their best bet is to buy an entire volume or two and spend a few nights burning the midnight oil. If not TI-LINES, then buy a couple of books covering Computer Science generally (there are thousands, too many for me to point at one or two and say "I heartily recommend this").

Bear in mind that those who already have volumes 1 and 2 will not take kindly to seeing the same articles being churned out year after year. They expect (as will you, eventually) to see fresh ground being broken in virtually every issue.

Some respondents suggested that there was a dearth of product reviews, and that I ought to publish in-depth User reports to help others decide on what (and if) to buy.

Fine - when the Users write the in-depth reports I will eagerly publish them. The last major review, if memory serves me correctly, was of the MYARC Disk Controller by ALLEN BURT, many months ago (volume 2 ?).

I have been PROMISED in-depth reviews, but "promised" and "given" are worlds apart.

Others have been justifiably critical of the fact that some series' start off but don't continue, or articles are promised but never manage to materialise (or Booklets are being outlined but are never seen).

This happens because muggins here tries to do two things at once (both edit and contribute), and that situation is caused by a lack of ready-made articles (at all levels). I rely heavily on material from overseas and although I have a backlog of articles/information, it all has to be typed up in the standard format, proofread, altered to conform to simple English spelling/syntax/grammar, and updated where appropriate. I also have to research, format, write, and proofread any original work, and sometimes a large, unexpected article come in which requires editing, thus halting my own work.

In addition, I have all the other aspects of a User group to cope with (and which cannot really be delegated, for a variety of reasons, most of them relating to finance, efficiency, and editorial policy) such as answering letters (can't be delegated unless someone has a carbon copy of my brain!) and enquiries (some of that seems to be delegated because I sometimes refer enquirers to appropriate contacts, who sometimes never hear from the enquirer, or if they do, sometimes don't manage to respond to the enquirer's needs), debugging hardware (would cost a small fortune in postage and insurance to ship it around) or software (most of the enquirers never send their bugged software to me if they cannot obtain satisfaction over the phone, so I wouldn't expect them to behave any differently towards any other Debugger - only two people this year have actually sent me the items we discussed over the phone!), looking for

>>MORE>

information relating to software or hardware projects (most ITUGers don't want to pay exhorbitant prices for commercial items, but at the same time don't want to see articles in IT which show them how to build it/patch it for themselves), and generally being on the lookout for all sorts of bargains, information, contacts in a number of fields, etc.

I have had kind (and very generous) offers of "free" photocopying on occasion, but the problems which arise are enormous: unreliable and varying quality within an issue; unreliable availability; cost of postage of masters and copies; risks involved in sending valuable original materials through the post anyway; and I would be totally unable to exercise any kind of control over production.

I have had offers to proofread articles, but generally (a) nobody else knows exactly how I would tidy up a given article and (b) few, if any, ITUGers follow the standardly-accepted conventions on spelling, syntax, grammar, etc. If several different people proofread articles you end up with several different styles, and what the reader doesn't need is to be made conscious of that fact. The form and style should be consistent throughout, wherever possible, and this includes altering "foreign" spellings to conform to English conventions.

It's the kind of job which ALL editors in the "real" world do every day, but the effect is one which you don't notice until it is absent - when the editor has fallen down on the job.

Occasionally I have been sent a printed, formatted article for publication which has had to be retyped (a) because the print has been far too faint to photocopy satisfactorily (I have problems enough with fresh print!), (b) the format hasn't conformed to the IT standard, (c) the spelling, syntax, grammar, or general semantics have been faulty and (d) sometimes the information has been wrong or out of date.

I used to operate a "cut and paste, publish and be damned" policy which led to problems over quality and consistency, and after the last set of questionnaires (in 1986) I altered the presentation of IT to respond to readers' criticisms. The new look, new presentation style actually attracted some approval, but even now some ITUGers have indicated that presentation quality is still not good enough. At present, to achieve the kind of quality which I would like, it would cost me at least £600 for the proportional printing, high quality reproduction, daiseywheel that I would require.

I do try to respond positively to readers' criticisms, but I am unable to provide 220 customised issues every month, much as I would like to.

As a response to several suggestions, here is a brief summary:

I have toyed with the idea of splitting IT into two publications, one for Beginners, one for Boffins. That won't work: ITUGers are Boffins in some areas and Beginners in others, and what do those who fit neither category do ? Who would pay £20 a year for both publications ? Who would do all the work and provide all the articles ?

I looked at the possibility of providing "theme" issues - each issue devoted to a particular topic - but that would mean that only a handful of ITUGers would be catered for each month - a recipe for disaster.

>>>MORE>

I tried producing Supplements - that came a cropper after the first one, because no-one fancied being relegated (as they saw it) to a smaller circulation publication.

I looked at dividing the magazine up into discrete sections: so many pages for Beginners, so many for Boffins, but that collapsed because of the imbalance in available material (lots of highly technical stuff if I had the time to type it all in, or redraw it).

I looked at publishing a larger magazine on a bimonthly or quarterly basis, but from past experience (six years) I know that this makes the magazine slow to respond to changes in the market, and makes it very inflexible. It also doesn't provide a quick turnaround for responses from readers. By the time many adverts appear in such magazines, the items are either sold or the price has increased.

If you look at the "real" world, there are no bimonthly or quarterly publications at our level; they are weekly, or fortnightly, or mostly monthly (and some of those have a three month lead time).

I have by no means exhausted the possibilities, but I have carefully considered all suggestions (and countless ideas of my own, most of them rubbish). To give you some idea of how we stand, only ONE subscriber out of over 150 said he would gladly pay £15 a year for a 40 page IT.

Some thought I could break even on running costs by maintaining the subscription cost and magazine size but publishing less frequently; would you willingly accept half the material for the same money ?

Even the special offers and "bargains" which we find don't always find favour with everyone. As a specific example, take the ROMOX cartridges to which TONY BOWDEN alerted me. I will shortly have all of the stock available - another 300 cartridges - and then that's it. Tony is to be congratulated on having given such valuable service to ITUGers.

I thought I'd explained what the ROMOX cartridges were and the chance they represented, but not so, it seems. A surprisingly large number of ITUGers, especially those with unexpanded systems (who stand to benefit most) are either unclear as to what advantage they confer, or are not at all interested. This is tantamount to offering a drowning man a lifebelt, and having him angrily throw it back at you. Even the price of 50p was apparently considered too expensive by a minority.

If some of you think that I bought them for nothing and am happily making reams of profit, then talk to Tony. He will tell you how much they really cost.

Back to IT. The questionnaires to date indicate, once again, that ITUGers are split in their perception of IT. Roughly half (so far) have indicated that there is too much for the Boffins in the magazine, while the other half reckon there is too much for the Beginner. Help!

An overall view, then:

I can publish lots and lots and lots of articles for Beginners and Boffins and anyone in between. They can be simple, complex, witty, friendly, comprehensive, summarising, devastatingly informative, BUT...

>> MORE>

Only if someone writes them in the first place. Contrary to popular belief, I am not a font of wisdom and knowledge concerning the 4A, and I have spent the last 5 years (a) trying to get anyone who is, to share it with the rest of us, and (b) trying to learn a little more about just about everything (which is difficult when I have no time free!) so that I can try to fill in the gaps.

Writing is easy; writing well is not. It takes time to get an article right, and I have offered help to recalcitrant writers in the past, but to no avail. I can only publish what I am sent plus whatever I manage to find time to write (badly) myself. The more YOU write, the more time I get free to pursue some of the articles I started and haven't yet finished.

ITUG can offer all sorts of startling bargains and special offers, but only if someone does the footwork first, so keep your eyes and ears peeled (Beginners as well as Boffins). Don't say "nothing's available for the TI in my area" until you have personally visited every shop, warehouse, and residence, read every magazine, local newspaper, newsletter, and internal company bulletin, and have spoken with every living soul in the district.

THEN you can say "there is absolutely nothing for the TI in my area today" and be sure that you are right. But what about tomorrow ? TI bits and pieces do circulate, and a local trader near you might be worth tapping on a regular basis in case something comes in.

Spread the word: if your local paper or company bulletin has a FREE advertisement section, advertise ITUG (and yourselves) in it. I am trying to get out to as many non-members as possible, but it is very difficult when I don't know who, or where, they are!

It has been suggested to me by several ITUGers that if one direct subscriber to a group actually represents a dozen or more others who are interested in reaping the benefits but not in contributing either financially or otherwise, then those who aren't subscribing may lose out in the long run because the group does not have their direct support. If there are too few direct subscribers, the lower is the incentive for others to subscribe, and the lower is the incentive for anyone to produce hardware/software or to provide any services.

Whether this philosophy applies to ITUG I couldn't say, largely because I don't know how true it is of ITUG. Certainly TI-LINES has been read by far more people than actually buy it, to judge by some of the letters and phone calls I have received over the years.

At Derby, I was mildly reprimanded for having an Adventure Hints column in the magazine when most of the "fun" is in being frustrated and quite depressed by an inability to solve some of the riddles.

I publish an article index in every issue, so all anyone who DOESN'T want to be exposed to the hints does is to check which pages contain the material he/she doesn't want to be exposed to, and avoid them like the plague. Let your fingers do the walking...

Once or twice I have been advised that readers found certain articles condescending (mainly Beginning BASIC) and insulting to the intelligence while others said those same articles proved too complex for them and

>>MORE>

they have put them to one side against the day that they acquire enough learning to understand them. Some readers say that they learned TI BASIC years ago and cannot see the usefulness of articles covering such an "elementary" aspect. The majority of new subscribers over the last year have been totally new to computing and to TI BASIC - such articles are aimed at them, not at the old hands.

Articles in IT have barely begun to scratch the surface of the mountain of information relating to the 99/4A, and I do have a backlog of useful but unproofed material to release upon you. If you are patient (i.e., very, very patient) you will eventually see everything that has been promised.

Alternatively, you could try researching a few things yourself and putting pen to paper to let us all know - it's surprising how addictive having your name in print can become.

I will prepare a Guide To Contributors, if that will help, so that what budding writers provide won't require too much editorial work.

In the meantime, keep voicing your criticisms. In many cases I will probably have an explanation as to why something is not feasible, but from time to time someone is bound to come up with an improvement which benefits us all. Remember that I cannot take the "like it or lump it" approach - one ITUGer suggested that I should only publish TI-LINES if I had sufficient UK articles to do so in a given month, and that it would somehow galvanise dormant authors into action.

Unfortunately, people rarely respond to that treatment, and anyway, I couldn't stop myself from writing articles if I tried.

I've become addicted to seeing my name in print...

Well, I allowed myself 6 pages to provide some background to the kind of problem which I (and other User groups) face, so hopefully some ITUGers will now be able to say "Ah, I hadn't realised that..." and then help out on the writing front... I live in hope (and in Oxford).

I don't know whether it is worth pointing this out, but the subject of Superconduction has made an appearance on the "up to date" TV programme Tomorrow's World. You might vaguely remember that you read about it in IT a month ago. Well, I try to stay one step ahead...

Make a date in your diary. The next BLOXWICH WORKSHOP organised by GORDON PITT of the WEST MIDLANDS TI USERS is planned for SEPTEMBER 5th. More details will follow in due course. Contact Gordon on 0922 476373 if you have specific queries.

ITUG featured in PERSONAL COMPUTER WORLD recently. Alas, almost all the details were wrong! Can we EVER win ?

>> END

SORTING IN MACHINE CODE

This object code file was published in the JANUARY 1986 issue of the newsletter of the TI NOVA SCOTIA USER GROUP

The file had been downloaded from TIBBS and the author was presumed to be JOHN CLULOW

If you don't feel like straining your eyes and typing it all in (which is just what I did!) then send me a disk with return postage and I will record the necessary files for you.

First things first: the eye strain...

*see below

```
00CA0         A0000A07D0A08D0A09D0A0AD0A0AF0B0000B0001BFF00BC80B7F2ADF        0001
A0AF8C0AF0B02E0C0AD0B04C0B0201B0002B0420B200CB0200B4041B90207F2D1F        0002
A0B0EB834AB1316B06C0B9020B834AB1304B0200B1300B0420B2034B02027F2FBF        0003
A0B24B0064B04C3B04C4BD0E0B834BB06C3B38C2B04C3BD0E0B834CB06C37F254F        0004
A0B3ABA103B1004B04C4BD120B834BB06C4B04CAB04C9BC820C0AF2C00027F270F        0005
A0B50BC804C0004B058AB028AB0000B1602B0460C0C58B060ABC24AB0A197F2AEF        0006
A0B66BC049B0A11BC321C0002BC361C0004BC00CB0201B0001B0202C07D07F2D9F        0007
A0B7CBD820C0AF4C07D0B0420B2014BC3CCBC38DB058EB060EB83CEB133A7F21DF        0008
A0B92BC00EB0201B0001B0202C08D0BD820C0AF4C08D0B0420B2014B02017F2D1F        0009
A0BA8C07D0B0202C08D0B06A0C0C68B0280B0001B1301B10E9BC00FB02017F2B6F        0010
A0BBEB0001B0202C08D0B0420B2010B058FB83CEB131CBC00FB0201B00017F2B3F        0011
A0BD4B0202C09D0BD820C0AF4C09D0B0420B2014B0201C07D0B0202C09D07F2B0F        0012
A0BEAB06A0C0C68B0280B0002B1301B10E9BC00EB0201B0001B0202C09D07F2C3F        0013
A0C00B0420B2010B10C3BC00FB0201B0001B0202C07D0B0420B2010BC04D7F2FEF        0014
A0C16B604FB0281B0002B110BBC24AB0A19BC04FB0581BC089B0A12BC3817F27BF        0015
A0C2CC0002BC88DC0004B058ABC04EB604CB0281B0002B110BBC24AB0A197F27EF        0016
A0C42BC089B0A12BC88CC0002BC04EB0601BC881C0004B058AB0460C0B567F286F        0017
A0C58B04C0BC800B837CB02E0B83E0BC2E0C0AF0B045BB04C3BD0D1BC0027F23FF        0018
A0C6EB9452B1401BD0D2B06C3B0581B0582B9452B1A0BB1B0DB0603B15F97F27AF        0019
A0C84BC080B9812C07D0B1A04B1B06B0200B0003B045BB0200B0001B045B7F2DBF        0020
A0C9AB0200B0002B045B7FB76F                                                0021
50AF6SORT  7FD1FF                                                         0022
:       99/4 AS                                                           0023
```

And an example routine demonstrating its speed:

```
100 CALL CLEAR :: CALL INIT :: CALL LOAD("DSK2.SORTO"):: DIM T$(100)
110 FOR I=1 TO 100 :: T$(I)=RPT$(CHR$(INT(RND*26+65)),INT(RND*64+1))
    :: PRINT T$ (I) :: NEXT I :: INPUT "READY":Z$
120 CALL LINK("SORT",T$(),100) :: INPUT "DONE":Z$ :: FOR I=1 TO 100
    :: PRINT I: :T$(I) :: NEXT I
```

The command format is CALL LINK("SORT",array name followed by (), and highest numbered element up to which to sort). The SORT segment will sort from element 1 up to and including the specified element number, so you can sort an array selectively if you wish.

))END

* 2022 note: Type the above into TI Writer editor. DO NOT SaveFile.
Instead, PrintFile to: F DSK1.SORTO
the F forces a Display Fixed file. Stephen Shaw

S O F T W A R E S H U F F L E

A follow-on from GENERATING PSEUDORANDOM NUMBERS UNIQUELY from MAY 1987

Based on a letter from NORMAN HANCOCK, whose routine is also presented

This is what it's all about, as they say. It's certainly what I like to see: a response to an article in IT which demonstrates an alternative approach to the solution of a problem and which is of benefit to us all.

In this particular case, I misunderstood the specific nature of Norman's routine and thought he'd sent me a flawed SHUFFLING routine.

A subsequent phone conversation set me on the right path, but the idea of shuffling is one I'd overlooked (it was on the agenda for a future continuation of the SORTING & SEARCHING series - you DO remember that one, don't you ?) and so I've stuck in my four penn'orth on using a shuffling algorithm.

If you were with us during Volume 3, then you'll remember the tortuous article I wrote concerning the generation of unique pseudorandom numbers (issue 12, pages 28 et seq.). The routine was designed to conserve memory at the expense of speed. The routines presented here operate quickly, but tend to eat memory.

Norman Hancock wrote to me with a far simpler solution to the problem of generating unique pseudorandom numbers, a routine for which appears here and, as you will see, is considerably more powerful than my original approach.

It is limited by available memory, but for most purposes it will be more than adequate.

Here is Norman's routine:

```
10 REM THIS ROUTINE CHOOSES THE NUMBERS 0 TO 999 RANDOMLY & UNIQUELY
20 REM NBH 870518
100 DIM A(999)
110 FOR B=0 TO 999
120 A(B)=B
130 NEXT B
140 FOR B=999 TO 0 STEP -1
150 N=INT(RND*(B+1))
160 A(N)=A(B)
170 NEXT B
```

The first step is to fill the array A() with the numbers to be selected. Accordingly lines 110 to 130 do just that, storing values 0 to 999 in elements with respective subscripts. Thus A(0) contains 0, A(1)

››MORE›

contains 1, ... A(999) contains 999.

The next step is to run through the array from the largest subscript to the smallest, using a straightforward decrementing loop.

A pseudorandom number (PRN) is generated in line 150 using the loop variable to restrict the maximum value which can be generated. When B is 999, the PRN ranges between 0 and 999; when B is 998, the PRN ranges between 0 and 998; and so on.

Here comes the smart bit. Let us suppose that N evaluates to 126 on the first execution of line 150. The contents of element A(126) are therefore selected as the chosen value. In this case, A(126) actually contains 126. Somewhere between lines 150 and 160 therefore, you would insert whatever process you require - e.g., printing A(126), assigning it to an element in another array, storing it in a disk or cassette file and so on.

As the contents of A(126) have been selected, you don't want them to be selected again. However, the contents of A(B) - presently A(999) - have not been selected, and when B is decremented they will never be selected (N can't evaluate to a number greater than B), so the contents of A(B) are assigned to A(N) - i.e., to A(126).

This means that if N evaluates again to 126, the contents of A(126) will be different (at this moment, the number 999), and no duplication will occur.

After one complete pass through the loop, all the values will have been selected pseudorandomly and uniquely, and, most important of all, QUICKLY.

The contents of the array A(), after the loop's execution, cannot be used again under this algorithm.

However, using a modification of Norman's technique - a SHUFFLING ALGORITHM - the array can be used again and again. In fact, you can shuffle away to your heart's content; the probability that you will shuffle the array back to its original (sorted) state is extremely tiny. The modification and its effect is shown later.

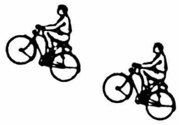
As Norman pointed out in his letter, the array need not contain numbers only. You could use a string array and select names, alphanumeric items (like ordering codes), playing cards, segments of text, anything. You could use two or more arrays in parallel, selecting corresponding elements from each of them (a TAGGED selection). You could also fill the array(s) from DATA statements using READ.

My modification to Norman's routine turns it into a shuffling routine. At the conclusion of a shuffle, the array A() still possesses its original contents, and second (and subsequent) shuffling can occur.

The major difference lies in the equivalent of line 160 in Norman's routine. Instead of displacing the contents of A(N) with those of A(B), the two elements' contents are EXCHANGED. This requires the use of an additional variable of the same type (numeric or string) as temporary storage for the contents of one of the elements.

))MORE)

```
100 DIM A(999)
110 INPUT V
120 FOR B=0 TO V
130 A(B)=B
140 NEXT B
150 FOR B=V TO 1 STEP -1
160 N=INT(RND*B)
170 T=A(N)
180 A(N)=A(B)
190 A(B)=T
200 NEXT B
210 FOR B=0 TO V
220 PRINT A(B);
230 NEXT B
240 GOTO 110
```



If you find it difficult to visualise the execution of this routine, here is a worked example:

I have incorporated the use of a variable V to hold the maximum number to be generated (up to 999) so for testing purposes you can use a 10 or 20 or whatever you're prepared to wait for.

The first section is very close to Norman's original: the A() array is filled with the numbers 0 to 999 so that A(0) contains 0, A(1) contains 1, ... A(999) contains 999.

Let us suppose that the loop in 150 has just begun. B is therefore V (say, 999). We will be exchanging the contents of A(B) (i.e. A(999)) with the contents of another element.

Let us further suppose that line 160 evaluates N to be 126. We will therefore exchange A(B) with A(N) or A(999) with A(126). Lines 170 to 190 achieve this.

We then begin the loop again, decrementing B. If B was 999 last time, it will be 998 this time. We will therefore exchange the contents of A(998) with the contents of another element.

If the Fates decide that N should evaluate to 126 again, we will exchange A(998) with A(126). Remember though that A(126) now holds what A(999) used to, and A(999) now holds what A(126) used to. After lines 170 to 190 have executed, A(126) will now hold what A(998) used to, and vice versa. Confusing, isn't it ?

If, for some perverse reason, N kept on evaluating to 126, you would end up with a "run" of sequential numbers occurring in the array. This is unavoidable, but might be less likely to occur if several shuffles were performed. If the RND function is not too biassed, lengthy "runs" should not appear.

The modifications in lines 150 and 160 of this routine take two things into account. If N evaluates to the same number as B, then the contents of an element will be swapped with itself (this applies also to Norman's

>>>MORE>

routine). Using RND*B ensures that N will always be in the range 0 to B-1, so N and B will never be equal.

Similarly, the loop need only decrement to 1 and not 0, since if B is 1, N will be 0 and if B is 0, N will still be 0, causing the final exchanges to be made between element 0 and itself. (Technically, even going down to a value of 1 for B is unnecessary, since it is a foregone conclusion that if B is 1, N must be 0!).

The final segment of the routine prints the shuffled array for you to examine, but of course you could replace this segment with whatever process you require.

Note that with Norman's routine your own processing MUST occur WITHIN the group of lines executing the B loop, while with the shuffle you MUST wait until the B loop has finished executing before beginning your own processing.

An apparently perfect shuffle would place the original contents of each element as far away as possible from the starting location, but the resulting array would actually be sorted, albeit in a rather perverse way, and therefore the sequence of numbers (or whatever) would ultimately be predictable (not a good idea).

The use of the variable V in the shuffle routine allows you to experiment and investigate using values much smaller than 999 (say, 10), and therefore pursue any improvements you may propose, with the minimum delay.

Having said that, the shuffling, or selection in Norman's case, of 1000 values doesn't really take that long. Only two passes through the array are needed - one to fill it up, and one to select from/shuffle it.

Norman also provided a brief modification to his routine which permits a graphic proof of the successful operation of the selection process. If any duplication occurs, you won't know which values were duplicated, but "holes" will appear in the screen image where missing values have failed to be represented.

In Norman's routine, change all references from 999 to 599, and add the following lines:

```
 50 CALL CLEAR
152 X=INT(A(N)/20)+2
154 Y=A(N)-INT(A(N)/20)*20+4
156 CALL HCHAR(Y,X,42)
```

The effect is to fill a 30 column by 20 row block with asterisks at random (or pseudorandom). If no errors occur, the block should be filled, with no spaces (or "holes") anywhere.

If anyone has further suggestions to make, you are most welcome to submit them for possible publication.

››END

LOGO, BASIC, AND RECURSION

The TRIADIC KOCH CURVE...

LOGO is what is known as a LIST PROCESSING language, and for those BASIC buffs who have never experienced LOGO, I suppose you could say that basically LOGO concentrates on the SEG$() side of things.

That's a gross over-simplification, but LOGO possesses "operators" (FIRST, BUTFIRST, LAST, etc.) which can best be compared with a combination of POS() and SEG$() and a chunk of BASIC in between.

However, LOGO is more likely to be remembered for its TURTLE GRAPHICS (although there are now forms of BASIC which provide the same capability), and that facility forms part of the subject of this article.

The main subject is RECURSION, and turtle graphics will be used to give a demonstration of its use and power.

Now, if you haven't limited your reading matter to TI BASIC alone, you may be aware that the pundits say that BASIC is incapable of recursion. In truth, "ordinary" BASIC finds it difficult - but not impossible - to emulate recursion; however, Extended BASIC (and BBC BASIC among a few others) CAN perform a limited form of recursion through the use of the User-definable subprograms (or Procedures in BBC BASIC).

What is this recursion, and what's so great about it anyway ?

Some will tell you, wrongly, that recursion is simply "self-referencing" and that it can easily be implemented by making a subroutine GOSUB to itself:

```
100 LEVEL=LEVEL+1
110 PRINT "ENTRY TO ROUTINE AT LEVEL";LEVEL
120 GOSUB 100
130 RETURN
```

This is a poor example (and will it ever execute line 130 ?) and it will run for so long before stopping with a MEMORY FULL error message.

(Although it will take quite a while before it does so!)

Many people have the feeling that recursion or self-referencing is a totally pointless exercise, precisely because the apparent BASIC equivalent will run into trouble and crash, and if LOGO doesn't, then LOGO is cheating somewhere along the line.

The truth is that recursion - or self-referencing - is nothing like the example above. LOGO programs too will run out of memory if they are instructed to refer to themselves continually without an IF-THEN somewhere to enable an exit:

>>>MORE>

```
100 LEVEL=LEVEL+1
110 PRINT "ENTERY TO ROUTINE AT LEVEL";LEVEL
120 IF LEVEL>10 THEN 140
130 GOSUB 100
140 PRINT "RETURNING"
150 RETURN
```

Again, a poor example, especially since it STILL doesn't emulate recursion. Note that the routine as referred to itself to a depth of 11 levels, and that there are 11 returns followed by a CAN'T DO THAT in 150!

What really happens when a recursive procedure is called ?

Well, in the first place you shouldn't get the idea that all LOGO programs are recursive. Recursion is a special property which is not always called upon - how many times have you had to use OPTION BASE 1 or RANDOMIZE N in your BASIC programs ?

When a LOGO procedure calls itself, all the variables which it has used are preserved, and, if you like, the procedure is run again as if it had not been run before. Although the same procedure is being used, it has a fresh set of variables - with the same names as before, but with whatever contents were supplied when the procedure began.

The analogous situation occurs when using a User-definable subprogram in Extended BASIC. If you define say H$="0123456789ABCDEF" in the main (or "calling" routine), and then CALL your own subprogram - say, CALL HEXDEC, - and attempt to use H$, you'll find that it is "empty" - it is a null string. When your subprogram finally returns control to the main or calling program, H$ will still contain "0123456789ABCDEF".

Whenever an Extended BASIC program enters a User-defined subprogram (UDS for short), a fresh variable list is created - a clean page, if you like. The only way that you can make use of a variable from the main program is to "pass" it. For example, CALL HEXDEC(H$,V$,V) where H$ is needed in the UDS.

This CALL passes three variables' contents to the subprogram HEXDEC, where H$ contains "0123456789ABCDEF" and V$ might contain a hex string to be converted to decimal, the result being assigned to V and passed back to the main program.

This is the true basis of recursion (and unfortunately Extended BASIC doesn't permit a UDS to call itself) where every time a routine calls itself recursively a new variable list is created.

So, recursion IS rather like a subroutine which GOSUBs to itself, but it is much more than that. For each time that a LOGO procedure calls itself, all the variables belonging to the previous "version" are preserved (unless you exceed memory capacity). The only way that standard BASIC (or Extended BASIC) can imitate this process is if all variables (bar a select few) are declared as ARRAYS.

Thus H$ would become H$(n), and V$() and V() similarly. Each time that the BASIC routine called itself, a general counter of the depth to which it had referred to itself - say, LEVEL - would need to be

>>MORE>

incremented (have 1 added to it) and it would be used to refer to all arrays: H$(LEVEL) instead of just H$, and V$(LEVEL) and V(LEVEL) similarly.

When an exit is made from the recursive routine (i.e., a return to the previous level), the variable LEVEL would need to be decremented (i.e., have 1 subtracted from it).

Passing of values from one "level" to another would be by simple transfer of the contents of one element in an array to the next. For example, if a routine calls itself and passes values for H$(), V$(), and V(), the process would be:

```
H$(LEVEL+1)=H$(LEVEL)
V$(LEVEL+1)=V$(LEVEL)
V(LEVEL+1)=V(LEVEL)
LEVEL=LEVEL+1
GOSUB routine starting line number
```

Similarly, immediately before exit, passing variables back would involve this process:

```
H$(LEVEL-1)=H$(LEVEL)
H$(LEVEL)=""
V$(LEVEL-1)=V$(LEVEL)
V$(LEVEL)=""
V(LEVEL-1)=V(LEVEL)
V(LEVEL)=0
LEVEL=LEVEL-1
RETURN
```

Horrendous, isn't it ? Once an exit is made from one level back to a previous one, all the variables associated with that level are lost - hence the assignment of null strings and zero above.

As you can see, recursion can be implemented in BASIC, but it requires a great deal of thought, and if you know beforehand that a routine is likely to call itself recursively say 20 times, you will need to DIM all arrays (bar LEVEL) to 20 elements.

If you were wondering how arrays themselves are handled, they simply need another dimension, and that dimension is used with the LEVEL variable. For example, Z$(a) would become Z$(LEVEL,a), and B$(a,b) would become B$(LEVEL,a,b).

This will upset the use of any three-dimensional variables in TI BASIC, or seven dimensional variables (!) in Extended BASIC. Also, the variable LEVEL must not be altered for any purpose other than its use prior to calls to, and returns from, the recursive routine.

Technically it shouldn't even be used in other equations/expressions either, as it is functioning as a SYSTEM VARIABLE.

LOGO requires none of this forethought and preparation, and only enough memory to allow it to operate.

))TO BE CONTINUED

D E R B Y   R E P O R T

On TI-EXCHANGE's Derby meeting held to decide its future

The "day" began late on Friday, when RICHARD SIERAKOWSKI travelled from Marlborough to Oxford, picked me up, and ferried us both up to GORDON PITT's Halfway House For Itinerant 99ers in Bloxwich. The three of us then discussed Life, The Universe, and Everything TI, prepared paperwork for Saturday, and swapped ideas until about 4.30 in the morning when we staggered to bed.

Three hours later we were up again (some of us with more alacrity than others) and when TREVOR DAVIES arrived we piled into his car and rocketed up to Derby. We arrived later than we had intended, but luckily the meeting had not been convened.

There were fewer people than I had expected, and I had been given to understand that the morning was to be given over to the meeting to decide the future of TI-EXCHANGE (where I might be called on to reply to any questions), while the afternoon was to have had "dealer support" and an auction. I had concluded that we were unlikely to see dealers travelling all the way to Derby if visitors were likely to hold off buying until after the auction, but in the event PARCO ELECTRICS traded continuously throughout the day and the auction was held towards the end (I missed it entirely - I never heard it announced).

The gathering of parties interested in determining the future of the group began and eventually called for me to respond to questions from the floor, which I hope I managed to answer adequately. There seemed to be considerable misunderstanding of both the nature of ITUG and of the nature of my invitation to TI-EXCHANGErs should they decide to wind up the group - some appeared to think I was attempting a take-over, while others thought I was proposing a merger. A few seemed to think that I was about to go bankrupt - I don't know where that one came from, but it now appears to have caused disquiet. For the record, bankruptcy is not a consideration. If a goblin grandmother (as opposed to a fairy one) waved its wand and zapped all the world's TIs I would STILL not be declared bankrupt, but my tax relief MIGHT come a cropper. I hope that ends the rumour.

I did find that I was asked basically the same question several times, and after a somewhat shaky start, with more discussion between members on the floor than between the Chair and the floor, finally a decision was taken to continue TI-EXCHANGE in some form of club. By the time that point was reached however, more than half of those gathered had walked away, so the vote was 29 to 5 in favour of continuing as a club, which was what I had expected to happen. Group members may not all be active participants, but if someone offers them the choice of closing down or changing course, the former is not a viable option!

Subsequently a break was taken for lunch, and afterwards those concerned reconvened to decide on committee structure and other details. In the

>>>MORE>

meantime I got on with a backlog of items which had accumulated during my spell under the spotlight (first time I have ever had to use a mike in public, so it was a sort of baptism by fire...).

I spent a tiring but enjoyable afternoon trying to multiplex my chatter between several ITUGers and others who fancied a chinwag, and the time, as usual, fled by until it was close to our moment to leave. At one point I even managed to buy a large wicker basket for £2 from Parco, much to the amusement of all present - except that they didn't know that in Oxford such things cost £4 and cannot easily be carried by a bald individual on a motorcycle...

I had not been aware that some of our party had another engagement to get back for or I would have made alternative arrangements, because in the end I missed the one thing I would really have liked to see - a link up between America, Canada, and the UK via modem.

I understand that the cost of this would have been exhorbitant had it been fully paid for at the UK end, so part of the cost (if not all - I don't have full details) was borne by the Canadians.

Alas, problems with the designated telephone meant that NEVILLE BOSWORTH had no success in even getting out of the building, electronically speaking, and eventually those interested had to move into the manager's office in order to try and overcome at least one of the hurdles.

I stayed long enough to see the initial success at logging into part of the network (it appears to have been a tortuous task for Neville to negotiate the different links between the world's networks, what with all the passwords and User IDs, etc., and he deserves full credit for all the work he put into the task), but then finally had to leave before contact was fully established.

I learned afterwards that ultimately success had eluded those present, and the planned link-up unfortunately did not take place. Perhaps we'll have better luck another time.

Finally, as an aside, the questionnaires being returned to date show a total lack of interest by almost all ITUGers in communication by modem, not so much now because of the cost of the modems (which are slowly becoming more realistic in their pricing) but overwhelmingly because of the cost of using the telephone.

I don't know about you, the reader, but when I see adverts placed by BT which extol the virtues of the UK network and its supposed low cost, I wonder just who is telling the truth. According to my local BT office, for example, Americans visiting this country do not know the true cost of making a call, which is why they complain bitterly when they receive their bills (this is a direct quote from a manager). However, logically IF overseas phone networks ARE so much more expensive than ours, then it follows that visitors to this country would be expected to be overjoyed at the low bills they receive. Instead, many of them send money to their friends and relatives back home to cover the bill incurred by REVERSING THE CHARGES from the UK. I am assured by family and friends alike that this is cheaper than using BT.

I suspect that modem communication will never take off in this country until someone manages to reduce the obstacle presented by BT's charges.

››END

THE INFINITE MONKEYS THEORY

Peter Brooks

May 1987

From time to time you may have heard or read odd quotes involving the obscure "Infinite Monkey" theorem. In fact, if the DIS/VAR WARS series had received support for its continuation, the theory would have been detailed in all its glory.

However, after being prompted by a HORIZON TV programme some months back I spent a little time following up the "Darwinian Evolution" approach to the Infinite Monkey theory.

Now, whether you accept Darwinian Evolutionary theory when applied to biological species or not, the principle of operation holds good for a wide range of activities, particularly in engineering. It can also have application in Philosophy, where it has some similarity to "Occam's Razor" (not the language used to program the parallel-processing Inmos Transputer!), and hence also in computer programming.

This is beginning to get a little too deep, and I have yet to discuss the principle of the Infinite Monkey theory...

The theory goes something like this: if you were to take an infinite number of monkeys, and sit each one down in front of a typewriter and leave him/her to their own devices (typing, naturally...) then within a finite period of time the full works of Shakespeare would be produced by at least one of them (or maybe they'd collaborate and produce a play or sonnet apiece!).

You can emulate this particular non-event by using a simple program on your TI, creating at random (or pseudorandom) letters of the alphabet and punctuation marks and stringing them together to form "words" and ultimately, sentences.

In fact, if you cast your eyes back to Volume 3, Issue 9, pages 13 to 17 and the article on pseudorandom prose generation, you will perceive the initial requirements of such a program.

You will also see that just creating a simple sentence at pseudorandom is an enormous task, requiring a tremendous amount of processing time, and with no guarantee of success. The "brute force" approach would be to cycle through all the possible combinations of letters and punctuation marks, and even assuming that you restrict your letters to 26 upper case characters and your punctuation to a single space, even a simple sentence like "FRIENDS ROMANS COUNTRYMEN LEND ME YOUR EARS" which has 43 letters/spaces, requires 3.54E61 iterations (using TI's notation) or 27 raised to the power 43!

The Darwinian approach is to define the target sentence which is to be generated, and instead of simply creating random selections of letters, a comparison is made between the target letter, the letter most recently selected, and the letter which has just been produced. The difference between the previous attempt and the target, and the current attempt and

>>>MORE>

the target, are compared, and if the current attempt lies closer to the target than before, it replaces the previous attempt.

This is reflected in the evolutionary approach to selection (in any closed system) where any new characteristic is only retained (but may not always be utilised) if it confers some improvement over the previous characteristic. This "improvement" may not always appear to be so; it will depend upon whether it carries "survival value". The "survival of the fittest" tenet behind such evolution is often (and deliberately) misunderstood - the word "fittest" here does not mean "fit" as in strong and healthy, but "fit" as in "most suitable" - e.g. "fit and proper".

In this particular instance, the "most fit" characteristic is that letter/punctuation mark which is identical to the target letter/etc.

A brief Extended BASIC program will serve to demonstrate the process. (It can readily be converted into TI BASIC, which task I leave to the interested reader.)

```
+-----------------------------------------------------------+
|                                                           |
|  DARWINIAN APPROACH TO PSEUDORANDOM TEXT GENERATION       |
|                                                           |
| 100 CALL CLEAR                                            |
| 110 COUNT=0                                               |
| 120 ACCEPT AT(12,1)VALIDATE(UALPHA):S$                    |
| 130 A$=RPT$(" ",LEN(S$))                                  |
| 140 DISPLAY AT(10,10):"COUNT=";COUNT+1                    |
| 150 DISPLAY AT(14,1):A$                                   |
| 160 FOR LOOP=1 TO LEN(S$)                                 |
| 170 C=65+INT(RND*26)                                      |
| 180 V=ASC(SEG$(A$,LOOP,1))                                |
| 190 W=ASC(SEG$(S$,LOOP,1))                                |
| 200 IF W=V THEN 260                                       |
| 210 IF C=V THEN 260                                       |
| 220 D1=ABS(W-V)                                           |
| 230 D2=ABS(W-C)                                           |
| 240 IF D1<=D2 THEN 260                                    |
| 250 A$=SEG$(A$,1,LOOP-1)&CHR$(C)&SEG$(A$,LOOP+1,255)      |
| 260 NEXT LOOP                                             |
| 270 COUNT=COUNT+1                                         |
| 280 IF A$<>S$ THEN 140                                    |
| 290 DISPLAY AT(14,1):A$                                   |
| 300 DISPLAY AT(16,1):"MATCHED AFTER";COUNT;"ATTEMPTS"     |
| 310 DISPLAY AT(18,1):"AGAIN (Y/N) ?"                      |
| 320 ACCEPT AT(18,16)VALIDATE("YN"):R$                     |
| 330 IF R$="Y" THEN 100                                    |
| 340 CALL CLEAR                                            |
|                                                           |
+-----------------------------------------------------------+
```

If line 120 is modified (delete the AT(12,1) then longer text strings may be entered.

As a point of interest, the FRIENDS ROMANS ... etc., string can be generated after only about 122 iterations - far fewer than 3.54E61!

>>END

X M O D E M   P R O T O C O L S

{This article seems to have been subjected to the usual Formatter gremlins - that is, there are incongruous statements and arithmetic expressions which appear to have lost certain symbols. I can only guess as to the nature of those symbols at present, and I cannot guarantee that my guess would be any better than that of the informed reader. PB}

[2022: Article proofread and corrections made, see note at bottom of page 26.]

Published in the JULY 1985 newsletter of the TI NOVA SCOTIA USER GROUP

Originally written by WARD CHRISTENSEN (around 1977 - before the TI)

Submitted by TERRY ATKINSON (who kindly allowed me to publish his RS232 and PIO DSR disassembly along with that of our own COLIN HINSON in the first (and only) International Supplement)

Since an XMODEM emulator is now available for the TI-99/4A, I think it is only fair that this article, written by the author of the protocol, {also called Christensen or CP/M modem protocols. PB} WARD CHRISTENSEN, be put in the TINS newsletter. Perhaps one of the Forth nuts out there would care to make a Forth Emulator using these protocols. TA

Rev: 08/09/82 WARD, C. Change ACK to 06H

Rev: (preliminary 1/13/85) JOHN BYRNS

At the request of Rick Mallinak on behalf of the guys at Standard Oil with IBM PCs, as well as several previous requests, I finally decided to put my modem protocol into writing. It had previously been formally published only in the AMRAD newsletter.

Table Of Contents

1. Definitions (Ward)
1B. Added definitions (John)
2. Transmission medium level protocol (Ward)
3. Message block level protocol (Ward)
4. File level protocol (Ward)
5. Data flow example including error recovery (Ward)
6. Programming tips (Ward)
7. Overview of CRC option (John)
8. Message block level protocol, CRC mode (John)
9. CRC calculation (John)
10. File level protocol compatibility (John)
11. Data flow examples with CRC option (John)

1. DEFINITIONS

(SOH)	01H
(EOT)	04H
(ACK)	06H
(NAK)	15H
(CAN)	18H

1B. ADDITIONAL DEFINITION

(C)	43H

2. TRANSMISSION MEDIUM LEVEL PROTOCOL

Asynchronous, 8 data bits, no parity, one stop bit.

The protocol imposes no restrictions on the contents of the data being transmitted. No control characters are looked for in the 128 byte data messages. Absolutely any kind of data may be sent - binary, ASCII, etc. The protocol has not been formally adopted to a 7 bit environment for the transmission of ASCII-only (or unpacked hex) data, although it could be simply by having both ends agree to AND the protocol-dependent data with 7F hex before validating it. I am referring specifically to the checksum, and the block numbers and their ones-complement.

Those wishing to maintain compatibility with the CP/M file structure, i.e., to allow modemming of ASCII files to or from CP/M systems, should follow this data format:

* ASCII tabs used (09H); tabs set every 8

* Lines terminated by CR LF (hex 0D hex 0A)

* End of file indicated by CTRL Z = ^Z = (hex 1A) = (CHR$(26)

{It is at this point that an omission is evident. It is likely that a caret (^) symbol, indicating "CONTROL", should precede the Z: ^Z. This problem occurs frequently, so I have inserted ^ where appropriate. PB}

* Data is variable length, i.e., should be considered a continuous stream of data bytes, broken into 128 byte chunks purely for the purpose of transmission.

* A CP/M peculiarity: if the data ends exactly on a 128 byte boundary, i.e., byte 127 is a CR, and byte 128 is an LF, a subsequent sector containing the ^Z EOF characters is optional, but is preferred. Some utilities or User programs still do not handle EOF with ^Zs.

* The last block sent is no different from others, i.e., there is no "short" block.

>>MORE>

3. MESSAGE BLOCK LEVEL PROTOCOL

Each block of the transfer looks like this:

(SOH)(BLK #)(255-BLK #)(--128 data bytes--)(CKSUM)

in which:

(SOH) = 01 hex.

(BLK #) = Binary number, starts at 01, increments by 1, and wraps OFFH to OOH (not to 01).

(255-BLK #) = BLK # after execution of an 8080 "CMA" instruction; i.e., each bit in the 8 bit block number is complemented.

Formally, this is the "ones complement".

(CKSUM) = The sum of the data bytes only. Toss any carry.

4. FILE LEVEL PROTOCOL

4A. Common to both sender and receiver:

All errors are retried 10 times. For versions running with an operator, (i.e., NOT with XMODEM), a message is typed after 10 errors asking the operator whether to "retry or quit".

Some versions of the protocol use (CAN), ASCII ^X, to cancel transmission. This was never adopted as a standard, as having a single "abort" character makes the transmission susceptible to false termination due to (ACK), (NAK), or (SOH) being corrupted into a (CAN) and cancelling transmission.

The protocol may be considered "receiver driven"; that is, the sender need not automatically retransmit, although it does in the current implementations.

4B. Receive program considerations:

The receiver has a 10 second timeout. It sends a (NAK) every time it times out. The receiver's first timeout, which sends a (NAK), signals the transmitter to start. Optionally, the receiver could send a (NAK) immediately, in case the sender was ready. This would save the initial 10 second timeout. However, the receiver MUST continue to timeout every 10 seconds in case the sender wasn't ready.

Once into a receiving block, the receiver goes into a one second timeout for each character and the checksum. If the receiver wishes to (NAK) a block for any reason (invalid header, timeout receiving data), it must wait for the line to clear. See "PROGRAMMING TIPS" for ideas.

>>MORE>

Synchronising: if a valid block number is received, it will be (1) the expected one, in which case everything is fine, or (2) a repeat of the previously received block. This should be considered OK, and only indicates that the receiver's (ACK) got glitched, and the sender retransmitted; (3) any other block number indicates a fatal loss of synchrony, such as the rare case of the sender getting a line glitch that looks like an (ACK). Abort the transmission, sending a (CAN).

4C. Sending program considerations:

While waiting for transmission to begin, the sender has only a single very long timeout - say, one minute. In the current protocol, the sender has a 10 second timeout before retrying. I suggest NOT doing this, and letting the protocol be completely receiver driven. This will be compatible with existing programs.

When the sender has no more data, it sends an (EOT) and awaits an (ACK), resending the (EOT) if it doesn't get one. Again, the protocol could be receiver driven, with the sender only having the high-level 1 minute timeout to abort.

5. DATA FLOW EXAMPLE INCLUDING ERROR RECOVERY

Here is a sample of data flow, sending a 3 block message. It includes the two most common line hits - a garbaged block, and an (ACK) reply getting garbaged. (XX) represents the checksum byte.

```
SENDER                                             RECEIVER
------                                             --------

                                          times out after 10 secs
                              <---                 (NAK)
(SOH) 01 FE -data- (XX)        --->
                              <---                 (ACK)
(SOH) 02 FD -data- (XX)        --->          (data gets line hit)
                              <---                 (NAK)
(SOH) 02 FD -data- (XX)        --->
                              <---                 (ACK)
(SOH) 03 FC -data- (XX)        --->
   ((ACK) gets garbaged)      <---                 (ACK)
(SOH) 03 FC -data- (XX)        --->                (ACK)
(EOT)                          --->
                              <---                 (ACK)
```

6. PROGRAMMING TIPS

* The character-receive subroutine should be called with a parameter specifying the number of seconds to wait. The receiver should first call it with a time of 10, then (NAK) and try again, 10 times.

After receiving the (SOH), the receiver should call the character receive subroutine with a 1 second timeout, for the remainder of the mesage and the (CKSUM). Since they are sent as a continuous stream,

>>>MORE>

timing out of this implies a serious line glitch that caused, say, 127 characters to be seen instead of 128.

* When the receiver wishes to (NAK), it should call a "PURGE" subroutine to wait for the line to clear. Recall that the sender tosses any characters in its UART buffer immediately upon completing sending a block, to ensure no glitches were misinterpreted.

The most common technique is for "PURGE" to call the character receive subroutine, specifying a 1 second timeout, and looping back to PURGE untl a timeout occurs. The (NAK) is then sent, ensuring the other end will see it.

* You may wish to add code recommended by JOHN MAHR to your character receive routine - to set an error flag if the UART shows framing error, or overrun. This will help catch a few more glitches - the most common of which is a hit in the high bits of the byte in two consecutive bytes. The (CKSUM) comes out OK since counting in 1 byte produces the same result of adding 80H + 80H as with adding 00H + 00H.

7. OVERVIEW OF CRC OPTION

The CRC used in the Modem Protocol is an alternate form of block check which provides more robust error detection than the original checksum. ANDREW S. TANENBAUM says in his book COMPUTER NETWORKS that the CRC - CCITT used by the Modem Protocol will detect all single and double bit errors, all errors with an odd number of bits, all burst errors of length 16 or less, 99.997% of 17 bit error bursts, and 99.998% of 18 bit and longer bursts.

The changes to the Modem Protocol to replace the checksum with CRC are straightforward. If that were all that we did we would not be able to communicate betwen a program using the old checksum protocol and one using the new CRC protocol. An intitial handshake was added to solve this problem. The handshake allows a receiving program with CRC capability to determine whether the sending program supports the CRC option, and to switch it to CRC mode if it does. This handshake is designed so that it will work properly with programs which implement only the original protocol. A description of this handshake is presented in section 10.

8. MESSAGE BLOCK LEVEL PROTOCOL, CRC MODE

Each block of the transfer in CRC mode looks like this:

(SOH)(BLK #)(255-BLK #)(--128 data bytes--)(CRC HI)(CRC LO)

in which:

(SOH) = 01 hex.

(BLK #) = Binary number, starts at 01, increments by 1, and wraps 0FFH to 00H (not to 01).

>>MORE>

(255-BLK #) = Ones complement of BLK #

(CRC HI) = Byte containing the 8 high order coefficients of the CRC

(CRC LO) = Byte containing the 8 low order coefficients of the CRC
See the next section for CRC calculation.

2022: This sectiondeals with the XMODEM CRC method, technically known as CRC_16_CCIT. There are other CRC 16 methods.

9. CRC CALCULATION

9A. Formal definition of the CRC calculation:

To calculate the 16 bit CRC, the message bits are considered to be the coefficients of a polynomial. This message polynomial is first multiplied by X^16 and then divided by the generator polynomial (X^16 + X^12 + X^5 + 1) using modulo two arithmetic. The remainder after the division is the desired CRC. Since a message block in the Modem Protocol is 128 bytes or 1024 bits, the message polynomial will be of the order X^1023. The high order bit of the first byte of the message block is the coefficient of X^1023 in the message polynomial. The low order bit of the last byte of the message block is the coefficient of X^0 in the message polynomial.

9B. Example of CRC calculation written in C

```
*/
This function calculates the CRC used by the "Modem Protocol".  The
first argument is a pointer to the message block.  The second argument
is the number of bytes in the message block.  The message block used by
the Modem Protocol contains 128 bytes.  The function return value is an
integer which contains the CRC.  The low order 16 bits of this integer
are the coefficients of the CRC.  The low order bit is the low order
coefficient of the CRC.
*/

int calcrc(ptr,count) char *ptr; int count; {
    int crc, i;
    crc = 0;
    while (--count >= 0) {
        crc = crc^(int)*ptr++ << 8;
        for (i = 0; i < 8; ++i)
            if (crc & 0x8000)
                crc = crc << 1^0x1021;
            else
                crc = crc << 1;
}
    return (crc & 0xFFFF)
}
```

2022 corrections:
Replace :0 with 0:
Replace two ^'s with &'s
and reformat.

In this article Peter used a source produced with TI Writer Formatter, which had removed some characters- never use TI Writer Formatter unless you really HAVE to. Formatter will remove any & (ampersand) in your text and underline the next word, and remove any @ symbol and print the next character in bold; in general it makes a mess of a document with #^@&. Listings become nonsense. (2022 note)

>>MORE>

2022 note Peter replaced odd blank spaces with a ^ - pure guesswork, This text has been checked and where necessary errant ^ symbols have been replaced with a & symbol. (To print & in formatter you have to type && in editor...).

10. FILE LEVEL PROTOCOL, CHANGES FOR COMPATIBILITY

10A. Common to both sender and receiver:

The only change to the File Level Protocol for the CRC option is the initial handshake which is used to determine if both the sending and the receiving programs support the CRC mode. All Modem Programs should support the checksum mode for compatibility with older versions.

A receiving program which wishes to receive in CRC mode implements the mode setting handshake by sending a (C) in place of the initial (NAK). If the sending program supports CRC mode it will recognise the (C) and will set itself into CRC mode, and respond by sending the first block as if a (NAK) had been received. If the sending program does not support CRC mode it will not respond to the (C) at all. After the receiver has sent the (C) it will wait for up to 3 seconds for the (SOH) that starts the first block. If it receives an (SOH) within 3 seconds it will assume that the sender supports CRC mode and will proceed with the file exchange in CRC mode.

If no (SOH) is received within 3 seconds the receiver will switch to checksum mode, send a (NAK), and proceed in checksum mode. If the receiver wishes to use checksum mode it should send an initial (NAK) and the sending program should respond to the (NAK) as defined in the original Modem Protocol. After the mode has been set by the initial (C) or (NAK) the protocol follows the original Modem Protocol and is identical whether the checksum or CRC is being used.

10B. Receive program considerations:

There are at least 4 things that can go wrong with the mode setting handshake:

1. The initial (C) can be garbled or lost
2. The initial (SOH) can be garbled
3. The initial (C) can be changed to a (NAK)
4. The initial (NAK) from a receiver which wants to receive in checksum can be changed to a (C)

The first problem can be solved if the receiver sends a second (C) after it times out the first time. This process can be repeated several times. It must not be repeated too many times before sending a (NAK) and switching to checksum mode or a sending program without CRC support may time out and abort. Repeating the (C) will also fix the second problem if the sending program co-operates by responding as if a (NAK) were received instead of ignoring the extra (C).

It is possible the fix problems 3 and 4 but probably not worth the trouble since they will occur very infrequently. They could be fixed by switching modes in either the sending or the receiving program after a large number of successive (NAK)s. This solution would risk other problems, however.

>>>MORE>

10C. Sending program considerations:

The sending program should start in the checksum mode. This will ensure compatibility with checksum only receving programs. Any time a (C) is received before the first (NAK) or (ACK), the sending program should set itself into CRC mode and respond as if a (NAK) were received. The sender should respond to additional (C)s as if they were (NAK)s until the first (ACK) is received. This will assist the receiving program in determining the correct mode when (SOH) is lost or garbled. After the first (ACK) is received the sending program should ignore (C)s.

11. DATA FLOW EXAMPLES WITH CRC OPTION

11A. Receiver has CRC option, sender doesn't

Here is a data flow example for the case where the receiver requests transmission in the CRC mode but the sender does not support the CRC option. This example also includes various transmission errors.

(XX) represents the checksum byte.

```
SENDER                                                  RECEIVER
------                                                  --------

                               <---                        (C)

                                              times out after 3 secs
                               <---                       (NAK)
(SOH) 01 FE -data- (XX)         --->
                               <---                       (ACK)
(SOH) 02 FD -data- (XX)         --->             (data gets line hit)
                               <---                       (NAK)
(SOH) 02 FD -data- (XX)         --->
                               <---                       (ACK)
(SOH) 03 FC -data- (XX)         --->
   ((ACK) gets garbaged)       <---                       (ACK)
                                              times out after 10 seconds
                               <---                       (NAK)
(SOH) 03 FC -data- (XX)         --->
                               <---                       (ACK)
(EOT)                           --->
                               <---                       (ACK)
```

11B. Receiver and sender both have CRC option

Here is a data flow example for the case where the receiver requests transmission in the CRC mode and sender supports the CRC option. This example also various transmission errors. (XXXX) represents the 2 CRC bytes.

(See over page)

>>MORE>

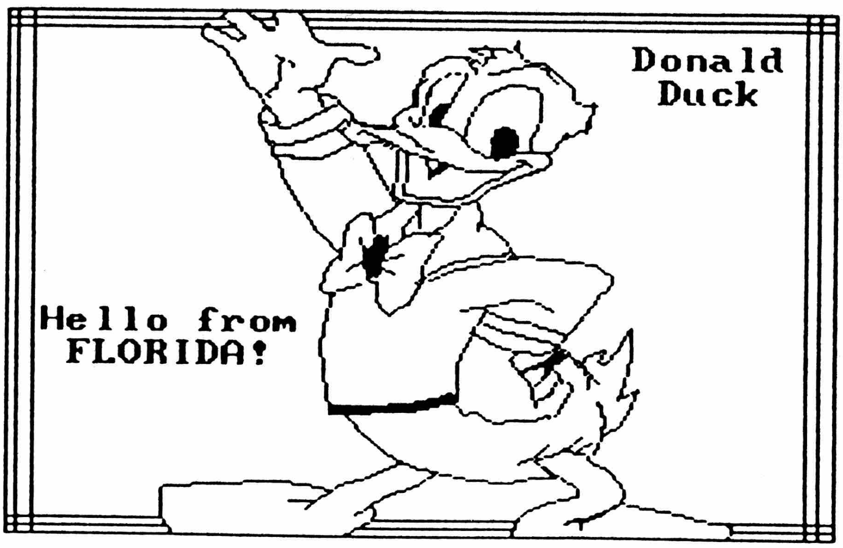
```
SENDER                                                          RECEIVER
------                                                          --------

                                          <---                     (C)
(SOH) 01 FE -data- (XXXX)                  --->
                                          <---                    (ACK)
(SOH) 02 FD -data- (XXXX)                  --->          (data gets line hit)
                                          <---                    (NAK)
(SOH) 02 FD -data- (XXXX)                  --->
                                          <---                    (ACK)
(SOH) 03 FC -data- (XXXX)                  --->
   ((ACK) gets garbaged)                  <---                    (ACK)
                                                     times out after 10 seconds
                                          <---                    (NAK)
(SOH) 03 FC -data- (XXXX)                  --->
                                          <---                    (ACK)
(EOT)                                      --->
                                          <---                    (ACK)
```

{This has been a fairly lengthy piece but XMODEM and CRC protocols have often been mentioned with regard to terminal emulation software. I very much doubt if this article has managed to clear the muddy waters at all, but at least it might act as a stimulus to further research.

On the other hand, you may now be in the throws of early hibernation...

PB}

>>>END



CAPTION COMPETITION: RESULTS

By SCOTT and JO ANN COPELAND (EAST ANGLIA REGION TI USERS)

We are very pleased to anounce the winning entries for the Caption Competition. A very big THANK YOU to everyone who entered!

It was extremely difficult choosing a particular entry as everyone had original captions. After many arguments between Scott and I, we called in an Independent Ajudicator who helped us out. I only wish that we could have given AIRLINE to everyone who took part.

Thanks again for your participation!

W I N N I N G   E N T R I E S

"What do you mean, TI BASIC is slow ?"

DEREK ALLEN, St Columb Major, Cornwall

"And all because he didn't know the password!"

DAVID ARMER, Brewood, Stafford

HONOURABLE MENTION:

"Admit it Peter, you're NEVER going to meet next month's deadline!"

J. S. DUNNING, Oldham, Lancashire

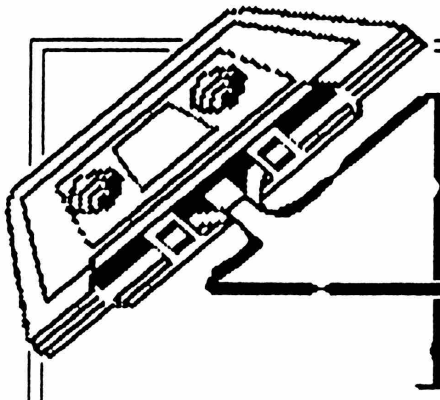
"Hey Old Timer, pull yourself together else it's bye-byes or end up a nobody"

HUGH TORMEY, Dublin, Ireland

Other captions to tickle your innards:

"I told you Airline was addictive!"
"Are you the guy with the suspected short circuit ?"
"I told you to ask Jo Ann for help with your Adventure!"
"Man, that is the WORST case of keyboard cramp I have EVER seen!"
"You really should have called Scott if you were that stuck on Zork!"
"Sorry I'm a bit late... the traffic is murder out there!"

››END

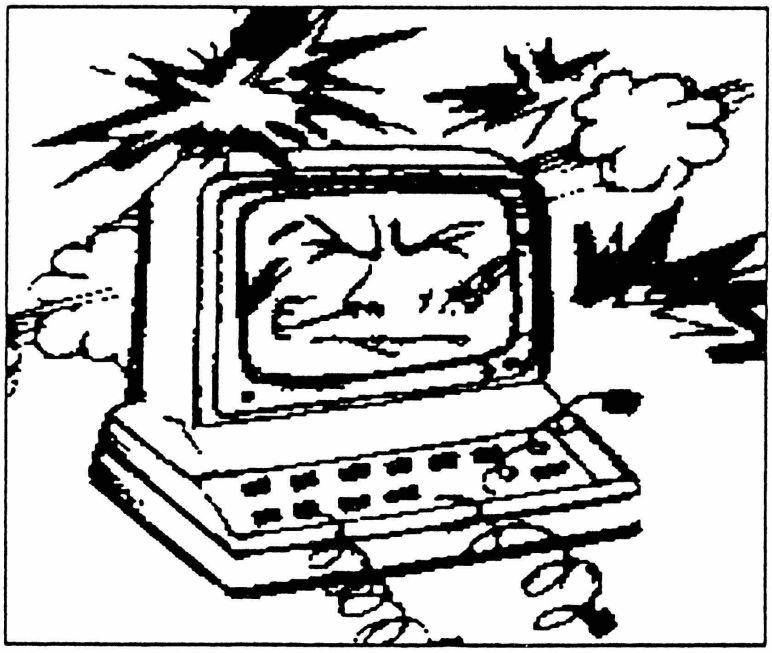


FRONT 4

CAN YOUR SYSTEM STAND IT ???

MAGAZINE ON DISK AND CASSETTE!

TI99/4A



TI99/4A

TAPE VERSION - £6.99 (ADD £2 OUTSIDE UK)
DISK VERSION - £7.99 (ADD £2 OUTSIDE UK)

CASSETTE VERSION RUNS IN EXTENDED BASIC
DISK VERSION REQUIRES EXBAS AND 32K RAM
PLEASE STATE IF DOUBLE SIDED DISK REQD.

New Dey
COMPUTING

Jerrard Close, Honiton, Devon. EX14 8EF (0404)41856

N O T I C E   B O A R D

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

MAURICE RYMILL (021 458 4970) has some items for sale:

MUNCHMAN	£ 3	YAHTZEE	£ 4
INVADERS	£ 3	INDOOR SOCCER	£ 5
PARSEC	£ 4	MUSIC MAKER	£15
PERSONAL RECORD KEEPING	£ 5	HOUSEHOLD BUDGET MANAGEMENT	£ 8
TI LOGO, MANUAL, SAMPLER DISK	£35	SUPER SKETCH (UNUSED)	£25
MINIMEMORY	£25	TI ARTIST (USE WITH MINIMEMORY)	£ 9

ISSUES 1 TO 4 OF TIHCUC MAGAZINE £ 4
ISSUES 2 TO 16 OF TI*MES MAGAZINE £ 5

GETTING STARTED WITH THE TEXAS TI-99/4A BY STEPHEN SHAW £2.50

GERRY AUSTIN (0384 637154) also has some items for sale:

TEII PROTOCOL MANUAL AND PILOT 99 MANUALS, PRINTED OUT	£ 2
GUIDE TO XB HOME APPLICATIONS (COMPUTE!)	£ 4
TI COLLECTION VOLUME 1 (COMPUTE!)	£ 5
GET MORE FROM THE TI-99/4A (MARSHALL)	£ 2
DYNAMIC GAMES FOR YOUR TI-99/4A (VINCENT)	£ 1.50
101 PROGRAMMING TIPS & TRICKS FOR THE TI-99/4A (TURNER)	£ 2
TEXAS PROGRAM BOOK (APPS)	£ 1.50
VIRGINS GAMES FOR THE TI-99/4A (NELSON)	£ 1
TANTALIZING GAMES FOR YOUR TI-99/4A (RENCO/EDWARDS)	£ 1.50
TI WRITER MANUAL	£ 3
EDITOR/ASSEMBLER MANUAL	£ 3.50
TI FORTH MANUAL (PRINTOUT IN RING BINDER)	£ 2
FULL SET OF TI*MES MAGAZINES (UP TO MAY 1987, MINT CONDITION)	£ 6.50

TI-99/4A HOME COMPUTER, BOXED AS NEW, WITH JOYSTICKS £55
HOME FINANCIAL DECISIONS, PARSEC, YAHTZEE, INVADERS, CONNECT FOUR, AND EARLY LEARNING FUN. ALL BOXED, VERY GOOD CONDITION. THE LOT FOR £ 7.50

(PROVISO: THE MODULES WILL ONLY BE SOLD SEPARATELY FROM THE CONSOLE IF THE CONSOLE HAS ALREADY BEEN SOLD)

OLDIES BUT GOODIES GAMES 1, BEGINNERS BASIC TUTOR £ 1.50
GAME WRITERS PACK 2 & BOOK £ 1.50

40 DISKS FULL OF ASSORTED PROGRAMS INCLUDING 4FRONT. THE LOT FOR £20
PHONE FOR DETAILS

10 NEW BLANK FLIPPIES £ 5

OR: SELL THE LOT FOR £100 PLUS POST, PACKING, AND INSURANCE

>>END