TI*MES

52

Issue 52

Spring 1996

Contents

All contributions for issue 53 must be submitted by May 4th 1996
You can use your modem to call the MOBB Bulletin Board on 01623 491282

Chairman		
Trevor Stevens	249 Southwell Road East, Rainworth, Notts, NG21 OBN	01623 793077
Vice Chairman		
Mark Wills	41 Broxtons Wood, Westbury, Nr. Shrewsbury, Shropshire, SY5 9QR	01743 884780
Gen. Secretary		
Richard Twyning	24 Peel Road, Mansfield, Nott, NG19 6HB	01623 27670
Treasurer		
Alan Rutherford	13 The Circuit, Wilmslow, Chesire, SK9 6DA	01625 524642
Disk Librarian		
Stephen Shaw	10 Alstone Road, Stockport, Cheshire, SK4 5AH	
Module Librarian		
Francesco Lama	14 Granville Court, Cheney Ln, Oxford OX3 0HJ	01865 721582
Hardware		
Gary Smith	55 Boundary Road, Newark, Notts, NG24 4AJ	01636 706787

Disclaimer

All the views expressed by the contributors of this magazine are strictly their own, and do not represent those of the committee. Contrary opinions are very welcome and errors will be corrected on request.

Editorial

Spring is in the air and the days are at last getting longer. Unfortunately, the same cannot said of my elderly console, the power supply of which is becoming ever more erratic. Still, with any luck, I should be able to rectify that at the Sandbach workshop in March.

I hope that some of you will be able to make to Sandbach on the 16th of March - if I can make it from the Sussex coast, then those of you who live in the Cheshire area have no excuses! I should have some spare seats, so if anyone in the Horsham area (or can get to Horsham) would like a lift, please get in touch.

I'm looking forward to hearing your comments / complaints about TI*MES, if you feel the mag has taken a turn for the worse under my editorship, then please let me know! I can't fix problems unless you tell me what they are!

One thing I have been asked for is more Basic and Extended Basic listings, so if you've got a program or two you've written, dig it out and send it in! As always, I can only print what it sent to me.

And now the usual apologies. The first must go to John Murphy who kindly supplied me with two disks full of material to use (and believe me, this issue I really could have used it!) Unfortunately the editorial system refused to even look at the directory of the disks, let alone retrieve any information. Richard Twyning was able to get a directory listing, but couldn'' extract the files. So now it looks like it'll up to Trevor. Hopefully it'll all be fixed in Sandbach.

That reminds me, did I mention the Sandbach workshop? The map's inside the back cover.

Also, I must apologise to the people who've written to me and not had a reply. I think I'm up to date on the letters front - but if you're sitting there fuming and feeling ignored; please get in touch! I now have so little time outside working hours (which have now hit the fourteen hours a day mark including travelling), that I simply cannot face even thinking about computers when I get home!

That reminds me, some of you may recall that I am by trade a Microsoft Windows programmer. There is a common misconception that this is a difficult system to program for. Not so. There are two development environments that are very easy to use and don't require expensive machines to run on.

The first is Visual Basic. This is Microsoft's baby and is in fact embedded in most of their applications (it is also the macro language used in their Word Processor and Spreadsheet amongst others.) The best thing about this particular program is that if you're used to Extended Basic, you'll have no problems with this whatsoever! It's just the same!

The second is Borland Delphi. This is a pascal based system, and so isn't quite so easy to convert to from Extended Basic (although is more logical once you get used to it). It's also very fast since it produces compiled programs instead of interpreted ones.

What's more, both are availabe *free* on the coverdisks of most of the current batch of computer mags

The point I'm making is that all the knowledge you've gained by writing compact and performance-tuned Basic or Extended Basic programs on the TI to fit in 13k and run acceptably on a cripped processor means that we TIers are probably the best Basic programmers in the world! But then we knew that already, didn't we?

So lets see some of those killer programs!

I'm waiting...

Richard Speed

Rambles

Greetings and (belated) New Year Greetings.

Once more no Basic queries!

Disk Library

News from the Disk Library first-

New disk: From Bruce Harrison, a machine code WORD SEARCH creator, and if you remember the Basic one that was around a while back, this is nothing like it.

This is FAST. Really really fast! This first version (amendments have been requested! and later versions will be supplied if available) has a fixed format of 25 x 25 characters and uses a word list you create with your favorite DV80 editor.

If you would like the words listed in alphabetical order below the grid there is a little utility to sort your word list for you (one word per line, DV80 format). Of course you could use the sort routine for other files!

Words are placed in all the usual directions, and print routines are available for 9pin and 24 pin dot matric printers. The 24 pin routines work with a number of other more modern printers too (such as the Canon BJ10 inkjet) with spacing set as optimally as possible to make solving easy.

Because the word placement is random you can use the same word list to produce a number of very different puzzles! Several word lists are supplied or supply your own! If the program reaches a place in your word list where it cannot fit a word you may ask it to restart from scratch, end the word placement there, or just skip the word and go on to try the next.

This program is first class and compares well with very expensive programs for other machines. Available on a single sided disk from your Group Disk Library!

Q*Bert Revisited

I now have confirmation that the revised Q*Bert in the disk library DOES function on an 80 column card!

Puzzle Solution

The puzzles in the last issue were from Personal Computer World Magazine. In January I received an update confirming that they had omitted one of the rules, and advising that there were five solutions as printed. I had already received a mathematic proof (not a program!) from Walter Allum shoping that there were six solutions. I think I will avoid PCW puzzles in future!

Puzzle

Simple puzzle- program required to solve it. I will list a program NEXT issue (editor (and time) permitting!):

Hymn Numbers

The other Sunday the hymn numbers on the board appeared like this:

1 9 2
3 8 4
5 7 6

All of the digits were different! The second number is twice the first amd the third number is the sum of the first two! Are there any other sets of three THREE DIGIT numbers with these properties?

Taken from "Brainteasers for Basic Computers" by Gordon Lee, Shiva Publishing Ltd, 1983

Wanted: Fast programs, or colourful programs, or animated programs, ANY program to solve the puzzle!

[Trivia- I travel to work daily on a 192 bus. During WW2 my father served on the Bryony, Corvette K192]

Harder Puzzle

In the year 2083 the User Group has five members. They get together with all their modules with the intention of sharing them out equally. They went out to the pub. One at a time they came back to the module pile....

The first returnee separated the modules into five equal piles. There was one left over which he threw away. He took the modules in one pile, put them in his bag, and then pushed the remaining four piles back together.

The second returnee did the same thing- divided the modules into five piles, threw away one left over, put one of the five piles in his case, then pushed the other four back together. The third, fourth and fifth returnees did the same thing!

Now all the members return together, divide the pile into five piles, throw away the odd one left, each takes one of the five piles and they go off home. Of course some have more modules than others....

The query is this: What is the MINIMUM total of modules to start with? A good maths student can solve this without a computer (and even produce a general formula!) but can you solve it with a computer? Sample listings to the Editor or myself, and a worked solution and program in next issue... maybe.

[The problem is old but has been taken from "Computers in Mathematics: A Sourcebook of Ideas" edited by David H Ahl, article by D T Piele and L E Wood. published 1979 by Creative Computing Press]

Basic

My programming language is Basic, so my solutions to the above puzzles will be given next month in Basic. Other languages (preferably commented!) are very welcome! If anyone has any questions on Basic programming, or would welcome a Basic tutorial; if anyone has something they would like to do in Basic but how do you.... please drop me a line and you may provide a spring board to a longer article next time! Personal direct replies MAY be subject to delay, and always an SAE will help a reply get to you!

Take a look at the latest PCs with their very highly advertised Windows 95 operating systems. You buy one. You want to write a simple program... what do you do???? You have not purchased a PC before so you do not have an older version of DOS which came with a Basic language. Perhaps you have never programmed before....

Older PC users with Quickbasic and a library of utilities can produce some VERY impressive programs cheaply and easily, but the next generation? Are going to be persuaded that 50 megabyte programs are in some way well written and worthwhile because they are 50 megabytes. Of course they require lots of investment in large hard disks, huge amounts of memory, and very very fast processors.... but do you really need them?

Warning- PC stuff follows

Thanks to Mike Poskitt I now have some new and revised files for the TI emulator version 5.01.

FILES ADDED TO TI EMULATE DIRECTORIES IN 1996:

MODULES: Saguaro City - original format pre module release of Tombstone City, Video Games, Aggressor, Astroblitz, Centipede, Hustle.

SMARTCOPY Alexander Hulpkes excellent print program for TI Artist format pictures and also pictures in YAPP/Geneve format. Docs are in PC format!

TIBASE...upgrade version to 3.02 (last issued)- complete in this directory.

A number of Extended Basic Kaleidoscope programs, some TI Artist fonts and pictures, Squeezer for TI Artist pictures, the excellent Mona Lisa printer and Jim Petersons Glamour Girl print program; and the XB utility Extractor.

Machine code object files (in TI Parlance DF80 files) which require the "Load and Run" option of the Editor Assembler module or the Funlweb loaders. These files are autostart files and do not require you to key in the entry point:

Galaxia and Entrapment- shoot the falling aliens before they reach the bottom.

TI CASINO modified to latest version, this needs to be configured as being on DSK2 - automatically done for you if you select from ADMAST menu.

MODULES- Extended Basic load menus are being added over a period but will not work with all modules in the directory some of which REQUIRE Funnlweb or Editor Assembler load routines. Go into Triton Super Extended Basic and load and run DSK3.LOAD to select from those which are happy from XB!

Let me know if you are interested in any of these or in the full emulator set of files (probably take 5 PC HD disks now! for the full set).

Short this month- short period, busy period.....

Magazine scanned 2022 by Stephen Shaw

Solution to last issue's puzzles

Walter Allum submitted the solution below to the puzzles Stephen set in issue 51

Problem 1

An answer is 36. But, unless there is an unspoken limit on the number of digits comprising the number, I do not see a way of proving by computer that 36 is unique. So far as I can see, doing this for a general *n* digit number would require solving a Diophantine equation of n^{th} degree - well beyond my capability. Perhaps I've blinded myself with science and missed a simple computerizable procedure.

Problem 2

There are six possible solutions

Cows	28	24	20	16	12	8
Geese	11	9	7	5	3	1
Hens	34	29	24	19	14	9
Horses	21	18	15	12	9	6
Sheep	18	16	14	12	10	8
Totals	**112**	**96**	**80**	**64**	**48**	**32**

I didn't need to write a program. One uses standard procedure for indeterminate simultaneous linear equations. It is helpful but not vital to have a matrix inversion program available to save heavy arithmetic.

Walter Allum

From The Chairman's Chair

How time flies by when you are enjoying yourself. I just can't believe that another year has gone by. I hope you all enjoyed the festive season and didn't get too drunk or over fed.

The BBs was well used over the Christmas period. The Board was on from 23rd Dec 1995 to 2nd January 1996 24hrs a day. There has not been much in the way of new users however. This I hope will increase as Ken Hughs has set about putting the BBs on the map with some advertising. The BBs at this time has two hard drives running. A 42meg and a 52meg. This gives us 94 meg of space. I have as a standby a spare 20meg which I will start to back up the BBs main programs on. I hope that the files I have on the two disks will fit on it. Yes the BBs is getting that full at this time. I never thought the TI would have so much data to store.

Talking of Hard drives, I have done a modification to the Hard disk controller. This was because Myarc made a cock up of the pre write compensation. This caused all sorts of problems. One drive liked the hard card, another didn't. So I had to get round the first hurdle by hopping the wire to the large data lead at No 2 if the drive would not format. Then came the problem of unreliable formats. This was why I went for the mod. This now allows me to format up and I believe passed the 136 meg barrier. This remains to be seen. However the standard card though it advertised the 136meg would not allow you to get there. If you are good with this sort of thing then here is the mod.

Look at your hard drive board out of its shell and find the Chip U9. Remove from dill holder and bend up pin 5. or if in the board cut this pin out. Now with the same chip bend up pin 12. Look again for chip U17 take out from dill and bend up pin 5. (This chip is just a short distance from U9.) With wire wrap, solder the U17 pin 5 to the wire and connect the wire to the pin 12 of U9.

All should now be seated well into the dill's and the board checked for any stray solder and funny touch connections. All will now act as it should do. This Mod was supplied by Myarc some time after the card went into production. Due extra labour costs was never implemented, so I am informed. Having discussed hardware and mods I would like to tell you that Ross Bennett is now pretty sure he understands most of the faults encountered on the TI including the controller cards etc. If you have any problems then he maybe able to help. He is on the BBs and can be contacted via the Message areas. He is also selling cheap and very good 5 & Quarter disks. See last months mag for full details, or look in the sales area of the BBS.

The AGM

This will be at Derby again as it appears to be the most central point for all concerned. The place is:-

THE ST.JOHNS AMBULANCE TRAINING CENTRE, Trinity Street, Derby
The Date: Sat 18th MAY 1996
The times: 9am set up, 10am doors open
5pm Close of AGM.

I look forward to seeing you there. Bring any problems, including the kids, and we see if we can solve them (no guarantees), but we will do our best. If you have any ideas about the group bring them along and air them at the meeting.

I will have the BBs on line for you to look and play with. If you have disks you may down load or upload a file if you wish. I can also give you a helping hand around the board, and show you how I make the screens in full living colour. I will also answer any questions along with Richard Twyning about your comms problems.

Now I suppose you will want something to test your brain. Well I have been a bit stingy this month after my Christmas gift to you. However I came across these two little programs a few years back and it prints out to your printer in reverse video. That is to say the background is black and the letters are in white. To do this you print the background and not the letter. You need a printer of course to make these little programs work.

```
Written by James Aaron.
1 !program  1
9 !Convert each character of p
attern to reverse Video
10 OPEN #1:"PIO" :: X$="012345
6789ABCDEF" :: FOR A=33 TO 127
 :: CALL CHARPAT(A,A$) :: FOR
B=1 TO 16 :: B$=SEG$(X$,17-POS
(X$,SEG$(A$,B,1),1),1) :: C=PO
S(X$,B$,1)-1
19 !CONVERT TO BINARY
20 IF C THEN C=C/2 :: D=C-INT(
C):: C=INT(C):: D$=SEG$(X$,D*2
+1,1)&D$ :: GOTO 20 ELSE E$=E$
&RPT$("0",4-LEN(D$))&D$ :: D$=
""
30 C$=C$&B$ :: NEXT B :: C$=""
39 !CONVERT TO THE PRINTER
40 FOR B=1 TO 8 :: E=128 :: FO
R F=B TO 63 STEP 8 :: G=G+VAL(
SEG$(E$,F,1))*E :: E=E/2 :: NE
XT F :: P(B)=G :: G=0 :: NEXT
B :: E$=""
50 PRINT #1,USING "### ### ###
 ### ### ###":P(2),P(3),P(4),P
(5),P(6),P(7):: NEXT A

1 !PROGRAM 2
9 !SET LINE SPACING TO ZERO; S
ET TO UNI-DI PRINTING; SET TO
EMPH,BOLDFACE & UNLI
10 OPEN #1:"PIO" :: Y$=CHR$(27
):: Z$=CHR$(0):: PRINT #1:Y$;"
3";Z$;Y$;"U";"1";Y$;"!";CHR$(1
52)
19 ! LOAD CHARS INTO RAM
20 FOR X=33 TO 127 :: READ A,B
,C,D,E,F :: PRINT #1:Y$;"&";Z$
;CHR$(X);CHR$(X);CHR$(139);CHR
$(A);Z$;CHR$(B);Z$;CHR$(C);Z$;
CHR$(D);Z$;CHR$(E);Z$;CHR$(F):
: NEXT X
29 ! SELECT DOWN LOAD CHARSET
;SET LINE SPACING TO 0 VERTICA
L SPACE BET LINES
30 PRINT #1:Y$;"%";"1";Z$;Y$;"
3";CHR$(27)
```

The first program prints 95 sets of printer codes to be put in DATA statements to follow the second second program, which will then be loaded by the second program, which will be loaded into the printer RAM. The reason the programs were written this way is that conversion from TI to the printer codes takes almost 20 mins to run. if the codes were loaded into ram immediately after conversion, this would take 20 minutes each time the printer was turned on. With the codes loaded from data statements, the load takes under a minute. Since the printer does not redefine the space character the char 127 (Ctrl V) is used. So if you use a CTRL V in your program instead of CHR$(127), the program will run without any problems. The only problem is when you list the program to a printer. CTRL V does not print out. Other fonts and scripts are able to be used. Like all TI users and programmers experiment and find out what you can do. You will be surprised and will find that when it comes to the other computers you have MORE CONTROL at your elbow.

Disk Drive Setups

Disk drives are the thing that I am always asked about. The problem seems to be when folks try to connect more than one drive to the controller. You must first of all make sure that the id number of the drive is set. This is normally done with jumpers. The first and second if fitted should have the terminator packs removed and the last in the line have the terminator pack in place. However some drives appear not to have terminator packs on them. They do but they are engaged /disengaged with a small jumper, normally near to where you set your id numbers. The next thing is to make sure the data cable is on the right way round. Very important this, as you can blow the buffer chips on the disk controller. Remember the golden rule. ***Red line on the tape is number one.*** and ***number one is the end near the slot on the interface board.*** In sort red to slot side.

Well that is it for this time I am afraid, Brain empty Fctn Quit.

Cartesian Plots

Some time ago I wrote an article about solving advanced mathematical problems with the TI99/4A, which some of you may remember. It appeared in issue 48 of TI*MES and permitted the solution of differential equations up to 10th order by a numerical method called the Kutta-Simpson integration procedure. You need to know very little about differential equations in order to use that program and you can run it in TI-BASIC, if you like, but you can store considerably larger arrays if you run it in X-BASIC.

The main drawback of that version of the program is that, since it was written to work without any expansion, it saves the data file to cassette, a thing which can certainly be long and tedious. I therefore recommend that, if you have an expanded system with disks you modify the following lines to read:

```
10 DIM YD(10),YD0(10),K1(10),K
2(10),K3(10),K4(10),Y(10,168)
180 FOR J=0 TO NEQ-1
```

(Modify this line in any case)

```
206 PRINT TAB(1);X0;TAB(16);YD
(0)
```

(Modify in any case)

```
208 PRINT TAB(1);YD(1);TAB(16)
;YD(2)
```

(Modify in any case)

```
350 OPEN #1:NAME$,SEQUENTIAL,I
NTERNAL,OUTPUT,FIXED
```

The above changes should enable you to generate files which outline the behaviour of the integral function which is the general solution of the differential equation you have chosen by entering it at line 6000 of the program. Incidently, I would like to point out another error which appeared in on page 52 of that article, about 2/3 of the way down the page. The end of the sentence starting "Lines 6000 and 6010 ....." should read "-10*SIN[YD(0)]".

Back to the present now! I had always wanted to write a program which would enable one to plot the results of DIFFeq (the differential equations program) in the form of an X-Y plot which could then be dumped onto paper, but all the versions of BASIC which permit the plotting of points and lines on the screen in a reasonably straightforward manner (such as TRITON SUPER EXTENDED, Mechatronic X-BASIC PLUS etc.) are somewhat rare, and therefore programs written for them are of little general use. This is why I decided to write the following program to work with The Missing Link X-BASIC environment. This means that any TI owners with an expanded system can use it, and, provided the file specifications are adhered to, can plot any curve they like in Cartesian form.

TML works within the Extended Basic environment, but this does not mean that all XB commands/statements are accepted. In fact all screen I/O statements have been replaced by machine language routines which, in a rather awkward and roundabout way, perform similar functions in BIT MAP mode. For this reason the program will seem rather longer than might be expected.

It is difficult to estimate how many user group members already own TML, but it is quite inexpensive and is supplied in disk form (+manual, essential!) by a number of outlets in the US. I got mine from Ramcharged Computers, PO Box 81532, Cleveland OH 44181, but I suspect they may no longer be in business. Other possible sources can be obtained either by checking through the ads in Micropendium, or, possibly, by contacting our group secretary ..... sorry Richard!

The TML code, given below, is extensively comented, but I will now give some details about the different routines and their function.

Lines 50-160 are devoted to variable initialization. This is why, when you run the program, it takes a little while before anything comes up on the screen.

Cartesian Plots

Line 163 sets the largest size of window in which text can output. Lines 165-167 ask how many files one is going to load. If the answer is 1 lines 220-330 carry out the loading of a whole file into the matrix array Y(10,168). This means that, as shown in line 240, the first element in your file must be the number of rows (1-11) in your matrix (NEQ), and the second the number of columns (NP, 1-168). XINP and HP are the starting value of X and they increment respectively, since this X-Y plotting routine is written for points equally spaced in X. A modification which will work for random X-Y pairs will be uploaded to the bulletin board by the end of the month.

Lines 375-430 ask whether the plotting of a particular row (here called LINE) is required and it goes through all rows of values from 0 to NEQ. At line 440 the program branches to the plotting routine (line 690).

When more than one file is loaded, individual rows are loaded from each file, so as to obtain superimposed plots of different data. Lines 480-650 carry out this operation 2-11 times, depending on the value of B specified in 167.

Lines 690-770 carry out the normalization procedure. Very often it is desirable to compare data which are dimensionally different, but expressed as functions of the same variable X (eg Voltage and Current in a circuit as functions of Time). This is done in order to compare where the maxima and minima of these functions occur and to get a general feel for their relative behaviour. Since one quantity may be expressed in terms of very large numbers compared with another, simple plotting of the two together might make one of the two graphs almost entirely disappear into the X-axis. In order to avoid this, it is common procedure to locate the element in each row which has the largest absolute value, and to divide all elements in the row by it. In this way the maximum and minimum possible values of elements in each row will be +1 and -1 respectively (only one need exist, not both).

Lines 820-979 allow one to view the loaded files on screen. Because of the absence of a scrolling facility in TML, it is necessary to press C at the end of each screenful, in order to go on to the next lot. It must be remembered that, since only Y values are stored, direct comparison of loaded data in this way is only possible if the XINP and HP values were chosen to be identical for all the files, when they were generated.

Lines 1008-1011 allow you to choose whether you want to plot the data or not. If NOT, line 1020 branches to 1610 where, if you decide to load new data, you are sent back to line 60 at the beginning of the program. NOTE that this branch address is incorrect in the version currently uploaded to the BBS and should therefore be changed.

Lines 1030-1080 determine the maximum ranges of X and Y for all the rows of data you have decided to plot, and lines 1090-1121 print these values on the screen. Subsequently lines 1130-1132 ask you for the range of X values you wish to plot. This is because no error will ensue from choosing to plot over a greater or smaller range than the one given in the data. The same remains true for Y at lines 1180-1182.

Lines 1140-1141 allow you to input the length of the divisions which you wish to appear on the X-axis and line 1150 calculates the number of such divisions.

Lines 1160-1161 are purely cosmetic. It is customary to have longer division markers every 10 divisions and intermediate ones every 5. This may be rather meaningless in some cases (eg when the length of X divisions is 0.25 of the unit), but, since you can always choose to stick to the convention, I decided to incorporate it into the program.

Lines 1170-1171 permit you to enter a label for the X-axis. Ideally this should incorporate the quantity plotted along this axis, the range of values, and the units used. A maximum of 30 characters are acceptable.

Lines 1190-1221 go through the same rigmarole for the Y-axis.

Lines 1230-1231 allow you to input a maximum of 30 characters as a title for the plot, and lines 1240-1242 ask you for the coordinates of the origin. This is necessary because the X and/or Y ranges may not include 0 (eg you may want the axes to cross at 10,-5).

Lines 1255-1256 ask whether you want the screen to be marked by a grid similar to the one used in graph paper, and lines 1260-1261 allow you to put in a request for a hard copy of the plot.

Lines 1310-1330 and 1410-1430 plot the X and Y axes respectively, in conjunction with subroutines 1680 and 1730 which calculate the screen values of the X and Y coordinates, and subroutines 2000 or 2500 and 3000 or 3500 (one or the other, depending on whether or not a full grid is required) which plot the division markers on the axes.

Line 1440 prints the X-axis label, and lines 1450-1480 print the Y-axis label vertically. The plot title is printed in line 1490. It must be NOTED that none of these labels will in any way encroach on the plot space, so feel free to use the maximum number of characters!

Lines 1500-1570 join up the points to plot the curve on the screen. Again the X and Y screen coordinates are provided by subroutines 1680 and 1730 respectively.

End of description ..... phew!!!

If you have a modem, you can save yourself the effort of typing the program in. It has been uploaded to the BBS Extended Basic area in archived form under the name FPLOT-ARK. You can easily download it using TELCO, and unarchive the files using ARCHIVER (available from the group disk library). The individual files you will find in it are: DIFFEQ (the differential equations solving program), FPLOTTML (the program described here), TEST1 and TEST2 (two data files with which you can try out FPLOTTML). The result of plotting either TEST1 or TEST2 should be decaying SINE-like curves, which represent the actual behaviour of a damped pendulum initially performing large oscillations. HAVE FUN!

NOTE: WHEN INPUTTING DATA FROM THE KEYBOARD YOU MUST WAIT FOR THE TML CURSOR (BLACK SQUARE) TO BE FLASHING AT THE SCREEN POSITION WHERE THE INPUT IS TO OCCUR.

```
10 REM this program reads up t
o 11 lines from the matrix fil
es produced by DIFFeq and plot
s them on screen.
20 REM the Y array contains al
l the dependent variable value
s for the loaded files, the YM
AX and YMIN arrays contain the
 maximum and minimum
30 REM values respectively for
 a row of Y, the XIN array con
tains the initial X
values of the loaded files,
40 REM the H array contains th
e X increments in each row of
Y, the N array contains the nu
mber of points(-1) in each row
 of Y.
50 DIM Y(10,168),YMAX(10),YMIN
(10),XIN(10),H(10),N(10)
60 CALL LINK("CLEAR")
70 REM
80 REM initialization of the v
ariables
90 REM
100 XMI=1E33 :: XMA=-1E33 :: Y
MI=1E33 :: YMA=-1E33 :: NMAX=0
110 FOR I=0 TO 10
120 YMAX(I)=-1E33 :: YMIN(I)=1
E33 :: N(I)=0 :: XIN(I)=0 :: H
(I)=0
130 FOR J=0 TO 168
140 Y(I,J)=0
150 NEXT J
160 NEXT I
163 CALL LINK("WINDOW")
165 CALL LINK("PRINT",1,1,"HOW
 MANY FILES (MAX 11)?")
167 CALL LINK("INPUT",1,200,B)
```

```
180 IF B>1 THEN 480
190 REM
200 REM complete loading of si
ngle file
210 REM
220 CALL LINK("PRINT",10,1,"FI
LE NAME?")
225 CALL LINK("INPUT",10,100,N
AME$)
230 OPEN #1:NAME$,SEQUENTIAL,I
NTERNAL,INPUT ,FIXED
240 INPUT #1:NEQ,NP,XINP,HP
250 NMAX=NP
260 FOR J=0 TO NEQ
270 FOR I=0 TO NP
280 INPUT #1:Y(J,I)
290 IF Y(J,I)>YMAX(J)THEN YMAX
(J)=Y(J,I)
300 IF Y(J,I)<YMIN(J)THEN YMIN
(J)=Y(J,I)
310 NEXT I
320 NEXT J
330 CLOSE #1
340 B=NEQ
350 REM
360 REM this part of the progr
am allows the plotting of only
 part of the single
file loaded.
370 REM
375 CALL LINK("CLEAR")
376 CALL LINK("WINDOW")
380 FOR J=0 TO NEQ
390 CALL LINK("PRINT",1+J*10,1
,"PLOT LINE")
392 CALL LINK("PRINT",1+J*10,8
0,J)
394 CALL LINK("PRINT",1+J*10,1
10,"(Y/N)?")
400 CALL LINK("INPUT",1+J*10,1
70,A$)
410 IF A$<>"Y" AND A$<>"y" THE
N 430
420 N(J)=NP :: XIN(J)=XINP ::
H(J)=HP
430 NEXT J
440 GOTO 690
450 REM
460 REM multiple file loading
operation. individual rows are
 loaded from each file.
470 REM
480 CALL LINK("CLEAR")
```

```
481 CALL LINK("WINDOW")
485 B=B-1
490 FOR L=0 TO B
500 CALL LINK("PRINT",1+16*L,1
,"FILE NAME?")
510 CALL LINK("INPUT",1+16*L,1
00,NAME$)
520 CALL LINK("PRINT",9+16*L,1
,"LOADING LINE (0-10)?")
525 CALL LINK("INPUT",9+16*L,2
00,LI)
530 IF INT(LI)<>LI OR LI<0 OR
LI>10 THEN 510
540 OPEN #1:NAME$,SEQUENTIAL,I
NTERNAL,INPUT ,FIXED
550 INPUT #1:NEQ,N(L),XIN(L),H
(L)
560 IF N(L)>NMAX THEN NMAX=N(L
)
570 FOR J=0 TO NEQ
580 FOR I=0 TO N(L)
590 IF J=LI THEN INPUT #1:Y(L,
I)ELSE INPUT #1:YDUM
600 IF J=LI AND Y(L,I)>YMAX(L)
THEN YMAX(L)=Y(L,I)
610 IF J=LI AND Y(L,I)<YMIN(L)
THEN YMIN(L)=Y(L,I)
620 NEXT I
630 NEXT J
640 CLOSE #1
650 NEXT L
660 REM
670 REM normalization procedur
e is selected here. normalizin
g a file permits the simultane
ous plotting of quantities wit
h different dimensions.
680 REM
690 CALL LINK("CLEAR")
693 CALL LINK("WINDOW")
695 CALL LINK("PRINT",1,1,"NOR
MALIZATION (Y/N)?")
697 CALL LINK("INPUT",1,200,A$
)
700 IF A$<>"Y" AND A$<>"y" THE
N 820
710 FOR J=0 TO B
720 IF ABS(YMAX(J))>ABS(YMIN(J
))THEN YM=ABS(YMAX(J))ELSE YM=
ABS(YMIN(J))
730 FOR I=0 TO N(J)
740 Y(J,I)=Y(J,I)/YM
750 NEXT I
```

```
760 YMAX(J)=YMAX(J)/YM :: YMIN
(J)=YMIN(J)/YM
770 NEXT J
780 REM
790 REM it is possible to disp
lay the Y array prior to plott
ing. unless the increments H i
n all the rows of Y are the sa
me, there is no direct
800 REM relation between adjac
ent printed data.
810 REM
820 CALL LINK("PRINT",15,1,"DI
SPLAY FILE (Y/N)?")
825 CALL LINK("INPUT",15,200,A
$)
830 IF A$<>"Y" AND A$<>"y" THE
N 1008
832 CALL LINK("CLEAR")
834 CALL LINK("WINDOW")
840 FOR I=0 TO NMAX+(NMAX/4-IN
T(NMAX/4))*4 STEP 4
860 FOR J=0 TO 3
870 FOR JJ=0 TO 2
875 IF JJ=2 THEN IILIM=2 ELSE
IILIM=3
880 FOR II=0 TO IILIM
882 CALL LINK("PRINT",1+16*JJ+
48*J,30+II*60,"Y")
883 CALL LINK("FORMAT",0)
884 CALL LINK("PRINT",1+16*JJ+
48*J,30+II*60+6,JJ*4+II)
885 CALL LINK("FORMAT",12,2,3)
886 CALL LINK("PRINT",9+16*JJ+
48*J,II*60+1,Y(II+JJ*4,I+J))
888 NEXT II
900 NEXT JJ
910 NEXT J
911 CALL KEY(0,KP,SP)
912 IF SP=0 THEN 911
913 IF KP<>67 THEN 911
914 CALL LINK("CLEAR")
915 CALL LINK("WINDOW")
979 NEXT I
998 REM
999 REM plotting procedure sel
ected here
1000 REM
1008 CALL LINK("CLEAR")
1009 CALL LINK("WINDOW")
1010 CALL LINK("PRINT",1,1,"PL
OT DATA (Y/N)?")
```

```
1011 CALL LINK("INPUT",1,200,A
$)
1020 IF A$<>"Y" AND A$<>"y" TH
EN 1610
1030 FOR J=0 TO B
1040 IF H(J)<>0 AND XIN(J)<XMI
 THEN XMI=XIN(J)
1050 IF H(J)<>□□0 AND XIN(J)+N
(J)*H(J)>XMA THEN XMA=XIN(J)+N
(J)*H(J)
1060 IF H(J)<>0 AND YMIN(J)<YM
I THEN YMI=YMIN(J)
1070 IF H(J)<>0 AND YMAX(J)>YM
A THEN YMA=YMAX(J)
1080 NEXT J
1090 CALL LINK("PRINT",9,1,"MI
NIMUM X=")
1091 CALL LINK("PRINT",9,100,X
MI)
1100 CALL LINK("PRINT",17,1,"M
AXIMUM X=")
1101 CALL LINK("PRINT",17,100,
XMA)
1110 CALL LINK("PRINT",25,1,"M
INIMUM Y=")
1111 CALL LINK("PRINT",25,100,
YMI)
1120 CALL LINK("PRINT",33,1,"M
AXIMUM Y=")
1121 CALL LINK("PRINT",33,100,
YMA)
1130 CALL LINK("PRINT",41,1,"X
-RANGE?")
1131 CALL LINK("INPUT",41,100,
XMI)
1132 CALL LINK("INPUT",49,100,
XMA)
1140 CALL LINK("PRINT",57,1,"L
ENGTH OF X-DIV?")
1141 CALL LINK("INPUT",57,150,
XDIV)
1150 NXDIV=INT((XMA-XMI)/XDIV)
1160 CALL LINK("PRINT",65,1,"L
ONGER X-DIV (1,2 etc.)?")
1161 CALL LINK("INPUT",65,180,
LMARX)
1170 CALL LINK("PRINT",73,1,"X
-AXIS LABEL (MAX 30 CHARS)")
1171 CALL LINK("INPUT",81,1,XL
AB$)
1180 CALL LINK("PRINT",100,1,"
Y-RANGE?")
```

Cartesian Plots

```
1181 CALL LINK("INPUT",100,100
,YMI)
1182 CALL LINK("INPUT",108,100
,YMA)
1190 CALL LINK("PRINT",116,1,"
LENGTH OF Y DIV?")
1191 CALL LINK("INPUT",116,160
,YDIV)
1200 NYDIV=INT((YMA-YMI)/YDIV)
1210 CALL LINK("PRINT",124,1,"
LONGER Y DIV (1,2 etc.)")
1211 CALL LINK("INPUT",124,180
,LMARY)
1220 CALL LINK("PRINT",132,1,"
Y-AXIS LABEL(MAX 18 CHARS)?")
1221 CALL LINK("INPUT",140,1,Y
LAB$)
1230 CALL LINK("PRINT",156,1,"
PLOT TITLE (MAX 30 CHARS)?")
1231 CALL LINK("INPUT",164,1,T
IT$)
1240 CALL LINK("PRINT",172,1,"
COORDINATES OF ORIGIN?")
1241 CALL LINK("INPUT",180,1,X
X0)
1242 CALL LINK("INPUT",180,100
,YY0)
1250 CALL LINK("CLEAR")
1251 CALL LINK("WINDOW")
1255 CALL LINK("PRINT",1,1,"FU
LL GRID (Y/N)?")
1256 CALL LINK("INPUT",1,160,A
$)
1260 CALL LINK("PRINT",10,1,"H
ARD COPY (Y/N)?")
1261 CALL LINK("INPUT",10,160,
HC$)
1270 CALL LINK("CLEAR")
1271 CALL LINK("WINDOW")
1280 REM
1290 REM drawing X-axis
1300 REM
1310 YY=YY0 :: GOSUB 1730 :: I
YP0=IYP
1320 XX=XMI :: GOSUB 1680 :: I
XP0=IXP :: XX=XMA :: GOSUB 168
0 :: CALL LINK("LINE",IYP0,IXP
0,IYP0,IXP)
1330 IF A$="Y" OR A$="y" THEN
GOSUB 2500 ELSE GOSUB 2000
1380 REM
1390 REM drawing Y-axis
1400 REM
```

```
1410 XX=XX0 :: GOSUB 1680 :: I
XP0=IXP
1420 YY=YMI :: GOSUB 1730 :: I
YP0=IYP :: YY=YMA :: GOSUB 173
0 :: CALL LINK("LINE",IYP,IXP0
,IYP0,IXP0)
1430 IF A$="Y" OR A$="y" THEN
GOSUB 3500 ELSE GOSUB 3000
1435 REM Printing X-axis label
1440 CALL LINK("PRINT",182,40,
XLAB$)
1445 REM Printing Y-axis label
1450 LY=LEN(YLAB$)
1460 FOR I=1 TO LY
1470 CALL LINK("PRINT",I*10,1,
SEG$(YLAB$,I,1))
1480 NEXT I
1485 REM Printing title of plo
t
1490 CALL LINK("PRINT",1,30,TI
T$)
1497 REM
1498 REM curve plotting routin
e
1499 REM
1500 FOR J=0 TO B
1510 XX=XIN(J):: GOSUB 1680 ::
 IXP0=IXP :: YY=Y(J,0):: GOSUB
 1730 :: IYP0=IYP
1520 FOR I=1 TO N(J)
1530 XX=XIN(J)+I*H(J):: GOSUB
1680 :: YY=Y(J,I):: GOSUB 1730
1540 CALL LINK("LINE",IYP0,IXP
0,IYP,IXP)
1550 IXP0=IXP :: IYP0=IYP
1560 NEXT I
1570 NEXT J
1580 CALL KEY(0,KP,SP)
1590 IF SP=0 THEN 1580
1600 IF KP<>67 THEN 1580
1605 IF HC$="Y" OR HC$="y" THE
N CALL LINK("DUMP")
1610 CALL LINK("CLEAR")
1615 CALL LINK("WINDOW")
1620 CALL LINK("PRINT",1,1,"LO
AD NEW FILES?")
1625 CALL LINK("INPUT",1,160,A
$)
1627 IF A$="Y" OR A$="y" THEN
60
1640 END
1650 REM
```

```
1660 REM calculates the screen
 value of the X coordinate of
a point
1670 REM
1680 IXP=INT(15+225*(XX-XMI)/(
XMA-XMI))
1690 RETURN
1700 REM
1710 REM calculates the screen
 value of the Y coordinate of
a point
1720 REM
1730 IYP=INT(174-156*(YY-YMI)/
(YMA-YMI))
1740 RETURN
1997 REM
1998 REM Draws X-axis division
s
1999 REM
2000 FOR I=0 TO NXDIV
2010 XX=XMI+I*XDIV :: GOSUB 16
80
2020 IF (LMARX-I-1)/10=INT((LM
ARX-I-1)/10)THEN CALL LINK("LI
NE",IYP0-5,IXP,IYP0+4,IXP):: G
OTO 2040
2030 IF (LMARX-I-1)/5=INT((LMA
RX-I-1)/5)THEN CALL LINK("LINE
",IYP0-5,IXP,IYP0+2,IXP)ELSE C
ALL LINK("LINE",IYP0-5,IXP,IYP
0,IXP)
2040 NEXT I
2050 RETURN
2497 REM
2498 REM Draws X-axis grid
2499 REM
2500 YY=YMI :: GOSUB 1730 :: I
YP0=IYP :: YY=YMA :: GOSUB 173
0
2505 FOR I=0 TO NXDIV
2507 XX=XMI+I*XDIV :: GOSUB 16
80
2510 IF (LMARX-I-1)/10=INT((LM
ARX-I-1)/10)THEN CALL LINK("LI
NE",IYP,IXP,IYP0+4,IXP):: GOTO
 2530
2520 IF (LMARX-I-1)/5=INT((LMA
RX-I-1)/5)THEN CALL LINK("LINE
",IYP,IXP,IYP0+2,IXP)ELSE CALL
 LINK("LINE",IYP,IXP,IYP0,IXP)
```

```
2530 NEXT I
2540 RETURN
2997 REM
2998 REM Draws Y-axis division
s
2999 REM
3000 FOR I=0 TO NYDIV
3010 YY=YMI+I*YDIV :: GOSUB 17
30
3020 IF (LMARY-I-1)/10=INT((LM
ARY-I-1)/10)THEN CALL LINK("LI
NE",IYP,IXP0+5,IYP,IXP0-4):: G
OTO 3040
3030 IF (LMARY-I-1)/5=INT((LMA
RY-I-1)/5)THEN CALL LINK("LINE
",IYP,IXP0+5,IYP,IXP0-2)ELSE C
ALL LINK("LINE",IYP,IXP0+5,IYP
,IXP0)
3040 NEXT I
3050 RETURN
3497 REM
3498 REM Draws Y-axis grid + c
ross over origin
3499 REM
3500 XX=XMI :: GOSUB 1680 :: I
XP0=IXP :: XX=XMA :: GOSUB 168
0
3505 FOR I=0 TO NYDIV
3507 YY=YMI+I*YDIV :: GOSUB 17
30
3510 IF (LMARY-I-1)/10=INT((LM
ARY-I-1)/10)THEN CALL LINK("LI
NE",IYP,IXP,IYP,IXP0-4):: GOTO
 3530
3520 IF (LMARY-I-1)/5=INT((LMA
RY-I-1)/5)THEN CALL LINK("LINE
",IYP,IXP,IYP,IXP0-2)ELSE CALL
 LINK("LINE",IYP,IXP,IYP,IXP0)
3530 NEXT I
3532 XX=XX0 :: GOSUB 1680 :: I
XP0=IXP :: YY=YY0 :: GOSUB 173
0 :: IYP0=IYP
3534 CALL LINK("LINE",IYP0+4,I
XP0+4,IYP0-4,IXP0-4)
3536 CALL LINK("LINE",IYP0+4,I
XP0-4,IYP0-4,IXP0+4)
3540 RETURN
```

The TI*MES Enthusiast

Greetings from those in the bad lands, we still have no rain, Northwest Water have no water, and it is COLD. To add insult to injury, my fridge packed up (Its only 20 years old!!) and we both have the sniffles. Moans over, let's wish all TIer's a belated happy and prosperous new year.

The year started really well, lying in our rattan hut on stilts in Thailand, warm and friendly, listening to thousands of fire-crackers and rockets going off. Maybe more about the place later. Last issue I promised to expand on the problem of static and how to avoid its destructive effects, so here goes.

The Satan Bug

Static charges can exist on any material that is not earthed, but mainly will occur on insulators especially plastics and glass. Touch your screen and you will probably hear the discharge from the glass to your finger, thats about 2 to 6 kv! Unless you're particularly sensitive you won't feel a shock as the current is very small indeed. So now you are charged up unless you are in contact with a good earth. Small as this charge is, it is sufficient to cause internal damage to an integrated circuit. The damage results in the layers of different semiconductor being stressed, causing a minute fault in the interface. This will eventually cause a breakdown in the layer, maybe years later, and the chip is duff. So the first and obvious precaution to take, is to earth your body before touching anything with semiconductors in it. No, don't lie on the floor with a circuit board and soldering iron, the carpet will have much more charge than you.

Earth in this context means a metalic conductor all the way to and into the ground. Water pipes used to be well earthed, now they are plastic, so safest is the earth in the electric wiring, providing that this is in good condition. All metal pipes in a house, water, gas, and central heating should be earth bonded to the main electric earth point, but beware, some wiring systems use earth leakage trips (ELCBs or more correctly residual current devices, RCDs) on the incoming supply. These will not have a continuous good earth through them. However any earth is better than none, so if handling chips or PCBs, wear a conductive wrist strap connected to an earthed object, e.g. the expansion box, radiator or pipe. Incidentally, all that ally trim on the TI console is not earthed, instead it forms part of the RF screening.

Chips should be packed in non-static packages or foam, usually black because it is made with conductive carbon, but metal foil and tins are usually fine. Avoid poly bags and boxes, polyethylene is a good insulator. For the same reason I don't use a plastic covered work table. It is also a good precaution not to wear clothing that can build up a charge by friction as you move, the jumper that emits sparks and crackles when you take it off is a definate no-no. So are nylon shirts (Does anyone still wear them?) and plastic shoes on a nylon or acylic carpet make for a really good static generator, almost like the Van de Graff generators in my old physics lab. Yes, I am that old, thank you.

Satanic Rights

Keeping equipment away from static is usually much less involved. Things like expansion boxes, printers, stand-alone units etc are normally in full or part metal cases and hence earthed. However, as the console and tape recorders are not earthed, they should be kept away from static sources such as the monitor/TV. Similarly it is a good idea to keep your disc boxes on an earthed case or at least ensure that the disc is touched against the earthed case of the drive before inserting it. Most drives do have a metal frame inside but some of the newer ones seem to be nearly all plastic.

Reading back through the last paragraphs it seems remarkable that any of our gear has lasted this long. Suffice it to say that the longevity of TI gear is due in the main to the quality of design and construction, and the chips used. Old time 74 series chips are much more resistant to damage than CMOS newcomers. Encasing the expansion PCBs in metal cases protects not only the actual PCB inside but prevents dischares through the connections into the main boards. If all this seems extreme, witness the number of failed Sinclairs, Ataris, etc. that we have heard about or experienced.

The other way that our chips are zapped is by the bits of wire going in and out of all the equipment. Main source is spikes on the mains supply, usually resulting in regulators going off or duff. The expansion box does have a filter in the mains socket at the back but this is an RF filter to prevent radiation out of the box. The console has a capacitor and toroid filter in the power board mainly for the same reason.

Exorcism Of The Demons

The "surge suppresor" type of mains plug or multi socket is the best place to catch the blips before they can get to the gear. A cheap and easy solution is to wire voltage dependant resistors such as the RS Componants metal oxide varistors (part # 2322-592-62712) or similar between all pins on the mains plug i.e. one between line and neutral, one between line and earth and one between neutral and earth. The smaller ones can usually be fitted in the plug top itself, but remember that if these absorb a hefty spike, they blow and need regular checking and replacement. Spikes can also get in through cassette recorder leads, printer leads and particularly modem connections to 'phone lines. Using good quality equipment connected with good leads which are kept away from mains cables will ensure the least likelihood of damage occuring. Ribbon cables for printer connection are cheap, but screened round cables give better protection from picking up unwanted signals and pulses from power cables.

Disconnecting items not in use reduces the number of possible ways unwanted signals can penetrate our venerable chips. TI did a splendid job of buffering practically every line into and out of the console and each and every expansion card. Some third party and home made add-ons are not so well engineered. If a sudden failure occurs it is usually a buffer chip or regulator that goes, failure of the 9900 processor itself is rare. My experience with Amstrads is that because the Z80 processor is directly connected to the expansion port, a simple loose or pulled out printer lead blows the processor itself. And most are not socketed, a lot of tricky soldering is required.

The total destruction of everything is likely to occur in the event of a direct lightning strike, so there is little point in worrying about that. Many people still remove all the mains plugs from the sockets when not in use and especially when thundery weather is likely, but the current opinion of the equipment manufacturers is that leaving the plugs in prevents charges building up by shunting them to ground. My attic ceiling is plastic coated, and in thunder storms the static on the plastic can be heard crackling to ground around the light fittings. Ball lightening, though not proven to exist, is thought most likely to be a ball of dry air so charged up that the molecules ionise and become a plasma ball. Whilst mentioning ionisation, the popular electrostatic air ionisers which work by producing small amounts of ozone are not a good idea near electrical equipment because they cause accelerated oxydation of copper and brass. When TVs became larger screen and hence higher EHT to the tube final anodes became necessary, the ozone produced in the sets was responsible for the tuner switch contacts oxydising at an alarming rate, which is one reason why the solid state tuner became popular so quickly. I used to see TVs where bad soldering on the EHT connections caused excessive corona discharges, and every piece of componant lead and bare wire was completely blackened with copper oxide. Trying to resolder anything in these sets involved sand-papering to remove it first.

Think though, does you house contents include the cost of replacing your "old" computer should it be totally wiped out? When we had a claim, only declared items over $500 were covered, the cost of replacing a full system with anything remotely as good as the TI would be much more than this.

Insurance companies don't usually charge any extra for covering the odd expensive item, but they do need to know before they have a claim to settle.

De-regulation

Thinking back to my remarks about uprating the voltage regs in the console, firehose, and PEB I have had cause to reconsider. Voltage regulators such as 78M05, 7805 and 78S05 have different current ratings. The regulation on the higher current ones is actually better than the smaller ones, but the available short circuit current is obviously higher too. This means that should anything go leaky or short, the excess current could do more damage. Its a toss up whether the longer life due to less heating, or the current limiting acting like a fuse is the important issue. Life is so full of compromises.

At least TI used linear power supplies, repairing the later types of switch mode supplies seems to be a black art, most are just dumped and replaced no matter what small failure occurs. PCs have supplies with amazing current outputs, fans to keep them cool and still fail usually before the rest of the computer. Maybe having to drive processors at 40meg plus and 8meg plus memories has a lot to do with it. Never mind, they will get it right one day, the one natural resource we will never run down is silicon.

Preserving The Species Or Avoiding Extermination

Having got this far with the TI it would be a shame to see users dumping their equipment for want of parts. Pat Trainor from N. Ireland 'phoned Trevor and myself the other day wanting a spare console. I hope we can help him and others in future, though it seems some items are becoming scarce. Perhaps as a group we should try to retain items whether working or not that are no-longer wanted. Even a console that doesn't work will yeild parts to repair several others, I wonder how many have been binned because the cartridge port connector needed cleaning and adjusting [*<Blush> - Ed*], or someone had the alpha lock on and could no longer play their games? I would be willing to stock-pile and repair gear so that we can supply users needs well into the future. If there are any bits not being used or duff that you have no use for, would you send them to me, postage refunded and even reasonable payments made, and I will redistribute them on a no charge basis to whoever can put them to a good use? I am thinking in terms of busted consoles and PEB cards as well as working systems. Disc drives are available at such low cost that they are not worth repairing, but there must be many users who could use a disc system or printer if they could be available for little cost. If you are going to the workshop, and I hope that many are, please bring the bits with you, we can have a swop meet.

Workshop 16th March

I hope to bring my system along, I have a TI 7000 emulator board that may be of interest if we can work out how to program it. I'll bring plenty of discs too if anyone wants any, the price is unbeatable for the quality and it saves-postage if you can pick them up at Sandbach.

If anyone wants a lift from around the Stockport area, ring me on 0161 430 7298 evenings please. I will have only 1 seat spare, and plenty of cargo space, its the works van for me, Christine will need the car that day. Perhaps Steven Shaw can come and we can bring the Disc Library?

In issue 50 I did ask if anyone could tell me where that damn alarm clock is in Return to Pirates Isle. A resounding silence followed, am I the only one that plays Adventures? Has no-one ever found it? Has everyone found it but won't tell? Before they have me in a straight jacket, please help. This article is written again without using the header (.HE text) in TI-Writer 'cos I still can't work that out either [*It makes no odds to me (in fact, the less formatting codes the better) - I just import the text into a DTP - Ed*]. Does anybody read this mag? Is there anybody out there?

That reminds me, the BBS last time I was on had achieved 17 users, there must be many more modem owners in the group than that. If not then lets help users aquire one, cheap, before the people who have put in all the time to set the thing up loose heart and close it down for everybody. Phone costs really do not enter into the argument for not trying it out, using only 2400 baud I can log in, read my and all new mail, see if any interesting new files have been uploaded and disconnect, all inside three minutes, costing at off peak about sod all. If anyone wants any help setting up, just ask, everyone will be glad to help.

This will have to be all the ravings for this article if Editor Richard S is to get it in time to edit it and publish, so see you at Sandbach, Great to be in a Pub all day without having to give an excuse!

End Of File, Ross.

THE TI*MES ENTHUSIAST ABROAD

Thailand - or more precisely Phuket which is an island state joined to the mainland by a bridge - was our New Year destination. Wonderful beaches, good extremely cheap accommodation and food and a serene local population what more could you ask for? We booked a flight with one night's accommodation included and then we were on our own.

Armed only with the information obtained from the Tourist Authority of Thailand and the Rough Guide we left Manchester on 27th. December arriving in Phuket via Gatwick and Bahrain on 28th. December where we were transported to our accommodation in the 4* Phuket Merlin hotel in the centre of Phuket town. As it was already evening and we had had very little sleep en route we went straight to bed. After a very early breakfast the next morning we checked out of the hotel and set off to look for some local transport. Within 2 minutes we were accosted by a tuk-tuk (local taxi, used to be a rickhaw, now a Diahatsu pick-up) driver who after some haggling took us to our chosen destination on the other side of the island. The first bungalows we tried were far too expensive, they wanted £25 per night! In order to get over that shock we walked onto the beach to consider where to try next. A local approached us and asked if we were looking for somewhere to stay and suggested Happy Hut which was up a steep hill just past the first place we tried. The climb was worth it, rooms were only £4 per night complete with balcony, en-suite bathroom, coathangers, mosquito net, resident toad and lizards to catch the insects!

Christine Bennett

Converting TI-Artist instances by Jacques Groslouis

Taking a TI Artist _I file and creating a D/V 80 file that will print graphics, using the Funlweb Editor. This version of TIA2FW5 handles instances with more than 9 columns if your Epson-compatible printer will handle reverse linefeeds and horizontal tabs.

A carriage return and linefeed is placed after every 80 characters in a D/V 80 file. This plays havoc with single-density graphics printing. My solution was to shorten the lines saved to disk and to override the carriage return and linefeed commands with reverse feed and horizontal tabs. Since the horizontal tab function of printer code relates to characters that are six dots wide and the program converts TI-Artist instance columns which are eight dots wide another problem arose.

The program listed below was used to convert the instance to a file that could be printed by the Funnelweb editor. This full picture instance took a bit over an hour to convert. Although the instance is 32 columns wide, it takes up the space of about 42 characters - add about one-third to the number columns - and entering 19 - being (80-42)/2 at the tab prompt in the program will center the instance when printing.

You can change the tab location by editing the third character of the first line of the file produced by my program. At tab 19, the third character is produced by CTRL U, SHIFT S, CTRL U.

```
TIA2FWSG
100 ! SAVE DSK1.TIA2FW5G
110 E$=CHR$(27):: Q,W=8
120 F$=E$&CHR$(3)&CHR$(0)::
HT$=CHR$(9):: RF$=E$&CHR$(10
6)&CHR$(24)
130 DEF ST$=E$&CHR$(68)&CHR$
(W)&CHR$(0)
140 CALL SCREEN(11):: DISPLA
Y ERASE ALL AT(1,4):"TI-ARTI
ST TO FWB v5.0": :"CONVERSIO
N PROGRAM"
150 DISPLAY AT(5,1):"INSTANC
E file name:" :: ACCEPT AT(5
,21)VALIDATE(UALPHA,DIGIT)SI
ZE(8):NAME$
160 DISPLAY AT(7,3):"The fil
e is on drive 1" :: ACCEPT A
T(7,24)SIZE(-1)VALIDATE(DIGI
T):FD
170 DISPLAY AT(8,1):"Which d
rive for new file? 1" :: ACC
EPT AT(8,27)SIZE(-1)VALIDATE
(DIGIT):SD
180 DISPLAY AT(11,1):"Print
at Tab Position? 10" :: ACCE
PT AT(11,24)VALIDATE(DIGIT)S
IZE(-2):T :: IF T>78 THEN 18
0
190 A$="DSK"&STR$(SD)&"."&NA
ME$&STR$(T)
200 NAME$="DSK"&STR$(FD)&"."
&NAME$&"_I"
210 OPEN #1:NAME$,INPUT :: O
PEN #2:A$,OUTPUT
220 INPUT #1:X,Y
230 GOSUB 430
240 IF X*Y>25 THEN DISPLAY A
T(20,4):"This may take awhil
e." :: DISPLAY AT(21,4):"Ple
ase be patient..."

250 PRINT #2:E$&CHR$(108)&CH
R$(T);:: DISPLAY AT(14,1):"R
OM:";Y;"COL:";X
260 PRINT #2:E$&CHR$(65)&CHR
$(8);ST$
270 FOR K=1 TO Y :: PRINT #2
:K$;
280 FOR L=1 TO X :: INPUT #1
:C(7),C(6),C(5),C(4),C(3),C(
2),C(1),C(0)
290 IF L=INT(X/6)*6 THEN K$=
CHR$(27)&CHR$(75)&CHR$(Q*(X-
L))&CHR$(0)
300 FOR I=7 TO 0 STEP -1 ::
A=C(I)
310 FOR J=7 TO 0 STEP -1 ::
IF 2^J>A THEN 340
320 A=A-2^J :: B(J)=B(J)+2^I
 :: IF B(J)=255 THEN B(J)=25
4
330 IF B(J)=32 THEN B(J)=16

340 NEXT J :: NEXT I
350 FOR I=7 TO 0 STEP -1
360 PRINT #2:CHR$(B(I));:: D
ISPLAY AT(16,1):"ROM:";K;" C
OL:";L;"Char:";B(I):: B(I)=0

370 IF ((L=6)OR(L=12)OR(L=18
)OR(L=24)OR(L=30))AND I=0 TH
EN PRINT #2 :: PRINT #2:ST$;
HT$;RF$;K$;:: GOSUB 430 :: M
=M+Q
380 NEXT I :: NEXT L :: IF X
>6 THEN PRINT #2:F$ ELSE PRI
NT #2:CHR$(13)

390 W=Q :: NEXT K :: PRINT #
2:E$&CHR$(108)&CHR$(0); !004
400 PRINT #2:CHR$(27)&CHR$(6
5)&CHR$(12)
410 CLOSE #1 :: CLOSE #2 ::
DISPLAY AT(23,3):"Another (Y
/N)?N" :: ACCEPT AT(23,17)VA
LIDATE("YNyn")SIZE(-1):Q$ ::
 IF Q$="Y" OR Q$="y" THEN 10
0
420 END
430 IF X<6 THEN K$=CHR$(27)&
CHR$(75)&CHR$(X*6)&CHR$(0)EL
SE K$=CHR$(27)&CHR$(75)&CHR$
(48)&CHR$(0)
440 RETURN
```

Tips From The Tigercub

In Tips #55, I showed you some quick and easy ways to create new character sets. Since folks nowadays don't like to key in long programs, let's continue with "tinygram" programming, and at the same time show you how to manipulate strings, and teach you the value of using MERGE format. First, let's make a screen to display our new characters. Some of them will have to be double-spaced horizontally or vertically, so -

```
100 CALL CLEAR :: X=1 :: FOR
CH=48 TO 159 :: PRINT CHR$(CH)
&" ";:: X=X+2 :: IF X<29 THEN
110 ELSE PRINT "":"":"";:: X=1
110 NEXT CH
```

Save it:
SAVE DSK1.100,MERGE

Now, you might like to move the common punctuation marks into the same character sets as the characters, so that you will not have to reidentify so many sets, also so you can color them easier.

```
120 DATA 32,33,34,44,46
130 FOR J=1 TO 5 :: READ CH::
CALL CHARPAT(CH,CH$):: CALL CH
AR(J+90,CH$):: CALL CHAR(J+122
,CH$)
140 NEXT J :: CALL CHARPAT(63,
CH$):: CALL CHAR(64,CH$):: ::
CALL CHAR(96,CH$)
```

If you want to program in Basic, or use BXB with characters all the way up to ASCII 159, add **CALL CHAR(J+154,CH$)** to the end of line 130 **and CALL CHAR(128,CH$)** to the end of line 140.

Save by **SAVE DSK1.120,MERGE**

If you are using that transliteration, you must remember that with upper case characters the ? is @, space is [, ! is \, " is], comma is , period is _. With the lower case they are FCTN keys C, F, A, G, W and V and for the 3rd set (ASCII 129 to 154) they are CTRL comma, period,;,=,* and (.

You can transfer upper case to lower by **CALL CHARPAT(CH,CH$)** and then **CALL CHAR(CH+32,CH$)** or the opposite by CH-32 and if you have BXB merged in you can create a 3rd set by CH+64.

The following are all incompatible with each other, so give them all line number 150 and save them in merge format as 150A, 150B, etc. The numerals and the upper case letters all have the topmost pixel row blank to provide spacing between lines of text. We can make taller letters by deleting the top row and doubling the 7th row -

```
150 FOR CH=48 TO 126 :: CALL C
HARPAT(CH,CH$):: CALL CHAR (CH
,SEG$(CH$,3,12)&SEG$(CH$,13,4)
):: NEXT CH
151 REM
```

Or, you can double the 3rd row -

```
150 FOR CH=48 TO 95 :: CALL CH
ARPAT(CH,CH$):: CALL CHAR(CH,S
EG$(CH$,3,4)&SEG$(CH$,5,12))::
 NEXT CH
151 REM
```

The lower case letters are really small upper case with the upper 3 rows blank. All their vertical bars are in the 4th, 6th and 8th rows, so let's drop the first 3 rows and quadruple the 7th.

```
150 FOR CH=97 TO 127 :: CALL C
HARPAT(CH,CH$):: CALL CHAR(CH,
SEG$(CH$,7,6)&RPT$(SEG$(CH$,13
,2),4)&SEG$(CH$,15,2)):: NEXT
CH
151 REM
```

Or, for topheavy letters, quadruple the 5th row -

```
150 FOR CH=97 TO 127 :: CALL C
HARPAT(CH,CH$):: CALL CHAR(CH,
SEG$(CH$,7,2)&RPT$(SEG$(CH$,9,
2),4)&SEG$(CH$,11,6)):: NEXT C
H
```

```
151 REM
```

Or, if you want line spacing -

```
150 FOR CH=97 TO 122 :: CALL C
HARPAT(CH,CH$):: CH$=SEG$(CH$,
5,8)&RPT$(SEG$(CH$,13,2),3)&SE
G$(CH$,15,2):: CALL CHAR(CH,CH
$):: NEXT CH
151 REM
```

Or, for something silly -

```
150 FOR CH=48 TO 90 :: CALL CH
ARPAT(CH,CH$):: CALL CHAR(CH,S
EG$(CH$,3,2)&RPT$(SEG$(CH$,5,2
),4)&SEG$(CH$,9,4)&SEG$(CH$,15
,2)):: NEXT CH
151 REM
```

For some good blocky characters -

```
150 FOR CH=48 TO 90 :: CALL CH
ARPAT(CH,CH$):: CALL CHAR(CH,R
PT$(SEG$(CH$,3,2),2)&SEG$(CH$,
5,8)&RPT$(SEG$(CH$,15,2),2))::
 NEXT CH
151 REM
```

Or, if you would prefer them shorter for single-line spacing -

```
150 FOR CH=48 TO 90 :: CALL CH
ARPAT(CH,CH$):: CALL CHAR(CH,"
00"&RPT$(SEG$(CH$,3,2),2)&SEG$
(CH$,7,6)&RPT$(SEG$(CH$,15,2),
2)):: NEXT CH
151 REM
```

If you would like numerals the same size as lower case,

```
150 FOR CH=48 TO 57 :: CALL CH
ARPAT(CH,CH$):: CALL CHAR(CH,"
0000"&SEG$(CH$,1,6)&SEG$(CH$,9
,4)&SEG$(CH$,15,2))::NEXT CH
151 REM
```

You can even shrink the lower case to only 4 rows high, although some letters are not very legible -

```
150 FOR CH=97 TO 122 :: CALL C
HARPAT(CH,CH$):: CALL CHAR(CH,
SEG$(CH$,1,6)&SEG$(CH$,5,4)&SE
G$(CH$,11,6)):: NEXT CH
151 REM
```

Something modernistic -

```
150 A$="00" :: FOR CH=48 TO 90
 :: CALL CHARPAT(CH,CH$):: CAL
L CHAR(CH,SEG$(CH$,1,4)&A$&SEG
$(CH$,7,6)&A$&SEG$(CH$,15,2)):
: NEXT CH
151 REM
```

Or perhaps even better -

```
150 A$="00" :: FOR CH=48 TO 90
 :: CALL CHARPAT(CH,CH$)::CH$=
SEG$(CH$,3,10)&RPT$(SEG$(CH$,1
3,2),2)&SEG$(CH$,15,2)
151 CALL CHAR(CH,SEG$(CH$,1,4)
&A$&SEG$(CH$,7,2)&A$&SEG$(CH$,
11,2)&A$&SEG$(CH$,15,2)):: NEX
T CH
```

I call this one "Spooky".

```
150 FOR CH=48 TO 122 :: CALL C
HARPAT(CH,CH$):: CH$=SEG$(CH$,
3,14)&SEG$(CH$,1,2):: X$=SEG$(
CH$,1,1)&"0"
151 FOR J=3 TO 15 STEP 2 :: X$
=X$&SEG$(CH$,J,1)&SEG$(CH$,J-1
,1):: NEXT J :: CALL CHAR(CH,X
$):: X$="" :: NEXT CH
```

And "Spooky" backward -

```
150 FOR CH=48 TO 122 :: CALL C
HARPAT(CH,CH$):: FOR J=1 TO 15
 STEP 2 :: CH2$=CH2$&SEG$(CH$,
J,1)&SEG$(CH$,J+3,1):: NEXT J
:: CALL CHAR(CH,CH2$):: CH2$="
" :: NEXT CH
151 REM
```

Now, clear the memory with NEW, then -

```
MERGE DSK1.100
MERGE DSK1.120
```

Tips From The Tigercub

Add a line 500 GOTO 500. And start MERGEing in your series of "150" routines and running them to see what you have created.

Then, save these next routines in MERGE format as 160A, 160B, etc.

All normal characters have the leftmost column of pixels and the two right most columns blank, for spacing between letters. We can widen the character into the left column -

```
160 FOR CH=48 TO 122 :: CALL C
HARPAT(CH,CH$):: FOR J=1 TO 15
 STEP 2
161 CH2$=CH2$&SEG$("014589CD",
POS("01234567",SEG$(CH$,J,1),1
),1)&SEG$(CH$,J+1,1):: NEXT J
:: CALL CHAR(CH,CH2$):: CH2$="
" :: NEXT CH
162 REM
163 REM
```

Or widen it both left and right -

```
160 FOR CH=48 TO 122 :: CALL C
HARPAT(CH,CH$):: FOR J=1 TO 15
 STEP 2
161 CH2$=CH2$&SEG$("014589CD",
POS("01234567",SEG$(CH$,J,1),1
),1)&SEG$("028A",POS("04
8C",SEG$(CH$,J+1,1),1),1)
162 NEXT J :: CALL CHAR(CH,CH2
$):: CH2$="" :: NEXT CH
163 REM
```

Or even a full 8 columns wide by just changing the "028A" in line 161 to "0129"

For darker characters, we can shade them into the 7th column -

```
160 FOR CH=48 TO 122 :: CALL C
HARPAT(CH,CH$):: FOR J=2 TO 16
 STEP 2 :: IF SEG$(CH$,J-1,1)=
"1" THEN CH2$=CH2$&"18" :: GOT
O 163
161 IF CH=67 OR CH=71 OR CH=99
 OR CH=103 THEN 162 :: IF SEG$
(CH$,J-1,1)="4" AND SEG$(CH$,J
,1)="0" THEN CH2$=CH2$&"60" ::
 GOTO 163
162 CH2$=CH2$&SEG$(CH$,J-1,1)&
SEG$("0367CBEF",POS("02468ACE"
,SEG$(CH$,J,1),1),1)
163 NEXT J :: CALL CHAR(CH,CH2
$):: CH2$="" :: NEXT CH
```

Or shade them both left and right -

```
160 FOR CH=48 TO 122 :: CALL C
HARPAT(CH,CH$):: FOR J=1 TO 15
 STEP 2 :: $=SEG$(CH$,J,1):: P
=POS("0123456789ABCDEF",A$,1)
161 A$=SEG$("0367CDEF89ABCDEF"
,P,1):: B$=SEG$(CH$,J+1,1):: P
=POS("02468ACE",B$,1)::B$=SEG$
("0367CBEF",P,1):: CH2$=CH2$&A
$&B$
162 NEXT J :: CALL CHAR(CH,CH2
$):: CH2$="" :: NEXT CH163 CAL
L CHAR(74,"000C0C0C0C4C38"):
: CALL CHAR(106,"0000000C0C0C4
C38")
```

Or shaded into both of the rightmost columns -

```
160 FOR CH=48 TO 122 :: CALL C
HARPAT(CH,CH$):: FOR J=2 TO 16
 STEP 2 :: CH2$=CH2$&SEG$(CH$,
J-1,1)&SEG$("0377EBFF",POS("02
468ACE",SEG$(CH$,J,1),1),1)::
NEXT J :: CALL CHAR(CH,CH2$)::
 CH2$="" :: NEXT CH
161 REM
162 REM
163 REM
```

Or into all 8 columns -

```
160 FOR CH=48 TO 122 :: CALL C
HARPAT(CH,CH$):: FOR J=1 TO 15
 STEP 2 :: P=POS("0123456789AB
CDEF",SEG$(CH$,J,1),1)
161 A$=SEG$("0367CDEF89ABCDEF"
,P,1):: P=POS("02468ACE",SEG$(
CH$,J+1,1),1):: B$=SEG$("0367E
BFF",P,1):: CH2$=CH2$&A$&B$
162 NEXT J :: CALL CHAR(CH,CH2
$):: CH2$="" :: NEXT CH
163 REM
```

More neatly, shaded inward at right -

```
160 FOR CH=48 TO 122 :: CALL C
HARPAT(CH,CH$)
161 FOR J=1 TO 15 STEP 2 :: CH
2$=CH2$&SEG$(CH$,J,1)&SEG$("0C
8C",POS("048C",SEG$(CH$,J+1,1)
,1),1):: NEXT J
162 CALL CHAR(CH,CH2$):: CH2$=
"" :: NEXT CH
163 REM
```

Or inward at right, outward at left -

```
160 FOR CH=48 TO 122 :: CALL C
HARPAT(CH,CH$):: FOR J=1 TO 15
 STEP 2
161 CH2$=CH2$&SEG$("0367CBEF",
POS("01234567",SEG$(CH$,J,1),1
),1)&SEG$("0C8C",POS("048C",SE
G$(CH$,J+1,1),1),1)::NEXT J
162 CALL CHAR(CH,CH2$):: CH2$=
"" :: NEXT CH
163 REM
```

Here's a weirdo -

```
160 FOR CH=48 TO 122 :: CALL C
HARPAT(CH,CH$):: FOR J=9 TO 15
 STEP 2
161 CH2$=CH2$&SEG$("014589CD",
POS("01234567",SEG$(CH$,J,1),1
),1)&SEG$("028A",POS("048C",SE
G$(CH$,J+1,1),1),1)
162 NEXT J :: CALL CHAR(CH,SEG
$(CH$,1,8)&CH2$):: CH2$="" ::
NEXT CH
163 REM
```

Try changing that to **FOR J=1 TO 7** and **CALL CHAR(CH,CH2$&SEG$(CH$,9,8))**

And one more -

```
160 FOR CH=48 TO 122 :: CALL C
HARPAT(CH,CH$):: FOR J=1 TO 7
STEP 2
161 A$=SEG$("02468ACE",POS("01
234567",SEG$(CH$,J,1),1),1)::
B$=SEG$("0808",POS("048C",SEG$
(CH$,J+1,1),1),1):: CH2$=CH2$&
A$&B$ :: NEXT J
162 CALL CHAR(CH,CH2$&SEG$(CH$
,9,8)):: CH2$="" :: NEXT CH
163 REM
```

Now, clear the memory, MERGE in 100 and 120, put in a holding line 500 GOTO 500 and start MERGEing in all of the different combinations of the 150 and 160 lines and see how many different character sets you can make!

I like little programs that load quickly and do just what I want to do at the moment. And one of the things I wanted to do quickly was to find phone numbers. So, I used FUNLWEB to create a little file -

```
SMITH, JOHN (999) 111-2222
BUSH, GEO. (000) 123-1234
GHADDAFI, O. (666) 66-6666
```

and all my other frequently called numbers. I SAVEd it as DSK1.PHONELIST and wrote this little routine to use it.

```
100 CALL CLEAR
110 OPEN #1:"DSK1.PHONELIST",I
NPUT
120 DISPLAY AT(12,1):"LAST NAM
E?" :: ACCEPT AT(14,1):N$
130 LINPUT #1:M$ :: IF POS(M$,
N$,1)<>0 THEN DISPLAY AT(16,1)
:M$ :: RESTORE #1 :: GOTO 120
140 IF EOF(1)<>1 THEN 130
150 DISPLAY AT(16,1):"NAME NOT
 FOUND" :: RESTORE #1 :: GOTO
120
```

Now actually, that was all I needed,(even though it did take several seconds to find a name at the end of the file), and it was easy enough to load the file into FUNLWEB when it needed updating. But, programmers are never satisfied, so I decided to write a self-contained program -

```
100 CALL CLEAR
200 DATA "ALDA, ALAN 888-9999"
201 !@P-
300 DATA "BUSH, GEORGE 111-111
1"
```

```
400 DATA "PRESLEY, ELVIS 000-0
000"
499 !@P+
500 DISPLAY AT(12,1):"LAST NAM
E?" :: ACCEPT AT(14,1):N$
600 READ M$ :: IF POS(M$,N$,1)
<>0 THEN DISPLAY AT(16,1):M$ :
: RESTORE 200 :: GOTO 500
700 ON ERROR 800 :: GOTO 600
800 DISPLAY AT(16,1):"NAME NOT
 FOUND" :: RESTORE 200 ::GOTO
500
```

That funny thing in line 201 turns off the prescan and speeds up initialization. This routine is no faster than the last, but can be updated by editing the program itself. It is limited to about 500 records due to the least-known and greatest weakness of the TI, that string storage is limited to console memory.

But, computer users are paranoid about speed, so I decided to put my data into a pre-loaded array with self incrementing subscript numbers, and find the data by a binary search.

```
100 !QUICKFINDER by Jim Peters
on
200 DIM D$(50):: GOTO 300 :: D
$(),X :: !@P-
300 X=X+1 :: D$(X)="ALDA, ALAN
 (999) 666-1234"
400 X=X+1 :: D$(X)="BUSH, GEOR
GE (111) 111-1111"
500 X=X+1 :: D$(X)="GHADDAFI,
OMAR (999) 456-1234567"
600 X=X+1 :: D$(X)="KHOMEINI,
AYATOLLAH  (666) 666-6666"
700 !@P+
800 INPUT "NAME? ":M$
900 IF M$>D$(X)THEN PRINT "NOT
 FOUND":"CLOSEST IS":D$(X):: G
OTO 800
1000 IF M$<D$(1)THEN PRINT "NO
T FOUND":"CLOSEST IS":D$(1)::
GOTO 800
1100 H=X :: S=INT(X/2)
1200 S$=D$(S):: IF POS(S$,M$,1
)=1 THEN 1700
1300 S$=D$(S+1):: IF POS(S$,M$
,1)=1 THEN S=S+1 :: GOTO 1700
1400 IF S$>M$ THEN H=S :: S=IN
T(H/2):: GOTO 1600
1500 S=S+INT((H-S)/2)
1600 IF S=S2 THEN 1800 ELSE S2
=S :: GOTO 1200
1700 PRINT D$(S):: GOTO 800
1800 PRINT "NOT FOUND":"CLOSES
T ARE"
1900 IF D$(S2)>M$ THEN PRINT D
$(S2-1):D$(S2+1):: GOTO 800
2000 PRINT D$(S2+1):D$(S2+2)::
 GOTO 800
```

Note that in this case the records must be in alphabetical sequence. New records can be inserted in intermediate line numbers, in alphabetic sequence, always preceded by **X=X+1 :: D$(X)=**.

Obsolete records can be deleted, and records can be corrected in place if the correction does not change the alphabetic sequence.

This idea did not work out as well as I hoped. The maximum number of records is less than 300, for the reason mentioned above, and this leaves so little free memory that even a binary search is slow. However, for a smaller file this is perhaps the best method. For a large file, the best method is certainly a fixed sequential disk file, accessed by a binary search routine. But, that requires other routines to delete, add or change records, and had best be the subject of another Tips.

There is apparently a mistaken belief that sprites cannot be used together with my BXB routine. Not so - you can use all 28 of them!

However, you cannot change their color with **CALL COLOR(#,N)**. The only other limitations of BXB that I can think of, are that a single **CALL COLOR** cannot be used for multiple character sets and a single **CALL CHAR** can only reidentify one character. **CALL CHARPAT** cannot return the hex code of an ASCII above 143 because those ASCII's were not supposed to be available in Extended Basic. I have used BXB on hundreds of Basic-only programs and have had only two rare problems. If the program contains multiple line feed colons

::::::, the computer may rearrange them into pairs of double colons :: :: and lock up. Or, if the colons are before the text, as in PRINT:"something" you may get a puzzling error message.

Also on rare occasions you might get an error message indicating the subprogram was called from a line containing a **CALL CHAR**, if the programmer had inadvertently put more than 16 characters in the hex code. Basic just ignores any extra characters, and Xbasic uses them to reidentify the following ASCII, but BXB crashes.

From the TI*MES newsletter from England, here is an extremely useful bit of assembly which should be assembled as ALPHA/O and placed on the disk of every joystick program, or mbedded in it with ALSAVE.

```
        DEF    ALPHA
* save old R12
ALPHA   MOV    R12,@>FFFC
* 9900 CRU base=0
        CLR    R12
* signal alphalock key line
        SBZ    21
* check alphalock other side
        TB     7
* jump if state=on
        JNE    STATE
*  state=off
        SETO   @>FFFE
* as off skip next line
        JMP    JUMPA
*  state=on
STATE   CLR    @>FFFE
* stop sending to alpha key
JUMPA   SBO    21
* restore R12
        MOV    @>FFFC,R12
* standard XB return now
* clear error for basic
        SB     @>837C,@>837C
* return to calling program
        B      @>0070
        END    ALPHA
```

Now, put this in the first lines of the joystick program -

```
1 ! by M. Gikow, Andover  MA
August 1988
2 ! used with ALPHA/O, will de
tect whether Alpha Lock is up
(A=255) or down (A=0)
3 CALL CLEAR :: CALL INIT :: C
ALL LOAD("DSK1.ALPHA/O")
4 CALL LINK("ALPHA"):: CALLPEE
K(-1,A):: IF A=0 THEN DISPLAY
AT(12,1):"RELEASE ALPHALOCK" :
: GOTO 4 ELSE CALL CLEAR
```

I published this one in the C.O.N.N.I. newsletter. Barry Traver picked it up and put it in the TI Forum in Computer Shopper, but their typesetter garbled it, so here is how it was supposed to be -

According to the TI-Writer Reference Guide, page 77, when you select the PrintF command, then type C and space once and then the device name, any control characters with ASCII less than 32 are removed before the file is printed. With Funlweb, at least, this is not quite true. A carriage return character, ASCII 13, or a line feed character, ASCII 10, at the end of a line is actually not deleted but is changed to the space bar character, ASCII 32. This can be proved by running this little routine -

```
100 OPEN #1:"DSK1.(filename)",
INPUT
110 LINPUT #1:M$ :: PRINT M$:L
EN(M$):: IF LEN(M$)>0 THEN PRI
NT ASC(SEG$(M$,LEN(M$),1))
120 CALL KEY(0,K,S):: IF S=0 T
HEN 120 ELSE 110
```

Therefore, when a file is Filled/Adjusted and the line feed characters are stripped with the C option, the lines are one character longer than they appear to be. An apparently blank line also contains ASCII 32. Since these characters are blank, they normally do no harm. However, they can create problems when records are read into programs for multiple column printing or concatenation of strings. In these cases, this routine can be used to strip out any ASCII below 33 at the ends of records.

```
100 DATA INPUT,OUTPUT
110 FOR J=1 TO 2 :: READ J$::
DISPLAY AT(12,1)ERASE ALL:J$&"
 FILENAME?":"DSK" :: ACCEPT AT
(13,4):F$(J):: OPEN #J:"DSK"&F
$(J),UPDATE :: NEXT J
120 LINPUT #1:M$
130 IF ASC(SEG$(M$,LEN(M$),1))
<33 THEN M$=SEG$(M$,1,LEN(M$)-
1):: IF LEN(M$)>0 THEN 130
140 PRINT #2:M$ :: IF EOF(1)<>
1 THEN 120 :: CLOSE #1 ::CLOSE
 #2
```

Attention all newsletter editors! If you are going to print my Tips (or anything else that contains program listings!) through the Formatter, PLEASE first replace and transliterate the ampersand, asterisk, period, carat and "@" sign! Print this one through the Formatter and see why -

```
100 A=A*264 :: @=1
110 PRINT "1 . . . 2 . . . 3.
. .4 . . . 5 . . . 6 . . .7 .
. . 8 . . . 9 . . . 0"
120 M$=M$&A$&B$&C$ :: K=K^3
```

Here's how you do it. Load the above in the Editor, position the cursor at the beginning of the 1st line, hit FCTN 9, type RS and Enter, then /&/}/ and Enter. At the prompt, type A. Now get the cursor back to the beginning, repeat the above with /*/|/, and then /./\/ and /^/~/ and /@/{/
and the file should now look like this -

```
100 A=A|264 :: {=1
110 PRINT "1 \ \ \ 2 \ \ \ 3 \
 \ \ 4 \ \ \ 5 \ \ \ 6 \ \ \ 7
 \ \ \ 8 \ \ \ 9 \ \ \ 0"
120 M$=M$}A$}B$}C$ :: K=K~3
```

Now use FCTN 8 to open 5 lines at the top and add this transliteration -

```
.TL 92:46
.TL 123:64
.TL 124:42
.TL 125:38
.TL 126:94
```

Save the result, go to the Formatter and print it.

If my multi-column Printall program (Tips from the Tigercub #45) won't run on your Epson-compatible printer, try changing line 250 to -

```
250 ACCEPT AT(12,3)VALIDATE("1
23")SIZE(1):P :: IF P=2 THEN P
RINT #1:CHR$(27);CHR$(77)ELSE
IF P=3 THEN PRINT #1:CHR$(15)
```

You might also need to change the 136 in line 280 to 132.

If your printer offers the elite condensed option, you might want to add - `:" (4) ELITE CONDENSED"` to line 240, change the VALIDATE string in 250 to "1234", add `ELSE IF P=4 THEN PRINT #1:CHR$(27);CHR$(77);CHR$(15)` to the revised line 250 and add `+(P=4)*160` to the first statement in line 280.

Memory almost full,

Jim Peterson

The Art Of Assembly Part 9

Last time we spent most of our space dealing with file errors. This month we'll get into ways to deal with the normal situation of files that do open and get read or written.

Since space is limited, we'll concentrate on commonly used file types, like Display/Variable 80, Display/Fixed 80, and so-called Program files, also known as Memory Image files. After you've read this article, you should know how to change the source code to handle other file types.

One of the primary requirements for doing anything with files on the TI is to set aside some space in the VDP Ram for your Peripheral Access Blocks (PABs) and the Buffers you'll need to send or receive data from your files. To some extent, you are free to use many different locations in VDP, but must take care not to overlap areas important to your program's execution. If you're operating in Graphics Mode on the VDP (this is the normal mode when you enter from E/A Option 3) you'll find that any address above >1000 and below >37D7 can be used. In many cases we've put our PAB at >1000 and our buffer at >1050. It's important to insure that the buffer won't overlap either the PAB or >37D7. For most devices, the entire PAB, including the file descriptor, will not exceed 25 bytes in length. If, however, you're dealing with Hard Drive files, the descriptor may occupy many more bytes, including directory and sub-directory names.

If you are operating in TEXT mode on the VDP, another area in VDP Ram is open for your use, between >400 and >800. In our Word Processor, which operates in TEXT mode, we use that area for four separate PABs and their associated buffers.

If only one file at a time is opened, you can "recycle" and use the same PAB and Buffer area for any and all files you use. Otherwise, you'll need a separate PAB and Buffer for each of the files that are open simultaneously.

The practice we use for establishing PABs is fairly simple. The fixed data for the first ten bytes is placed in the data area of our source code, with a small area (usually 15 bytes) reserved beyond that for the file descriptor, which comes from user-entered data. An example is shown in the sidebar at label PAB1DT. This PAB data is preset for opening a D/V 80 file type. The code at label START shows how one might use our subroutines CRSIN and MOVSTR (given in previous articles of this series) to get the name of the file in place as a user input. In this case, the full version of CRSIN should be used, including the lower case to upper case conversion lines, so the user won't have to worry about having Alpha Lock engaged to enter a file name with all upper-case letters. One note of caution - after the call to CRSIN, it's wise to see whether the user has left the input blank, and issue him an error message if he has. CRSIN leaves the length of the input string in R2, so a simple MOV R2,R2 followed by a JEQ to jump somewhere and report an error will do the trick.

The code at label OPNF1 shows how to load that data into the PAB area in VDP, and then open the file. First, the file mode must be placed in the PAB1DT area. We've shown in the example a file to be opened for INPUT by moving a byte called INMD to PAB1DT+1. Incidentally, we don't recommend using UPDATE mode for Variable record length files. It is too easy to create an unusable file by trying to write to such a file in UPDATE mode. For Fixed record length files, UPDATE will allow you to modify records in the file at random without messing up the rest of the file.

It's important at this stage that the first byte in this PAB be 00, so that the file will OPEN. The next step is to write this data into VDP at the correct place, and to insure that the right number of bytes are written to include the complete file descriptor. The first example in the sidebar shows the right way to do this, so that the actual descriptor length is used to write all the necessary bytes into the VDP. If you're not dealing with the longer descriptors needed for Hard Drives or RS232, you can use the second, or shortcut, method, which writes 25 bytes regardless of descriptor length. (25 bytes will include the 10 of the PAB, plus 5 for the device name and period, plus 10 for the maximum acceptable file name length.)

Once the PAB has been written to the VDP, one needs to place the address PAB1+9 at location >8356, which we call PABPNT. The DSRLNK must have this address in that location so it can find the file descriptor in VDP Ram. We normally include a CLR STATUS before calling DSRLNK, but we're not sure it's necessary. In some cases, we've forgotten to do it with no ill effects. Be brave, and leave that line out.

In all cases, the BLWP DSRLNK line must be followed by a line reading DATA 8. Maybe there was supposed to be another use for DSRLNK in which some number other than 8 would be used, but we don't know about that. In any case, forgetting the DATA 8 will cause DSRLNK to fail. We've shown here the steps necessary to detect an error on opening the file, and that branches to the code we included in last month's article.

Given that the file opens, we can read records from the file using the code at label RDF1. The bytes READF, WRITEF, and CLOSEF will work for any kind of file except Memory Image type, in which they're not needed.

For illustrative purposes only, we've parked the contents of each record we read at location TEMSTR, which in this case has been set to a block of 81 bytes, 1 for the length of the record, plus 80 for the maximum possible record length. In a real application, you'd want to move that string to somewhere else before reading the next one from the file.

Here we've also included the error detection needed for read operations, but made the exception for error code 5, End Of File. If we've reached the end of file, we simply jump ahead to the close file operation at CLSF1. We should mention for those skilled in Extended Basic programming that this End of File error does not work exactly like the EOF function in XB. XB reports EOF when the last record in the file is read, while this error does not report until you try to read a record beyond the last record. Let's say for example there were forty records in the file opened as #1, and we're reading with XB. As soon as we've read the fortieth, XB will report EOF(1). In our Assembly case, Error 5 will not be reported until we try reading the forty-first record.

Incidentally, the XB Manual states that the EOF function will not work for Fixed Record length files. That's wrong. EOF works just the same for Fixed or Variable in XB. Error 5 in Assembly also works for both Fixed and Variable record lengths.

While we're at comparisons to XB, we should say that the method we've shown for reading into a string (TEMSTR) is essentially equal to a LINPUT function in XB, in that it places the entire contents of the record at location TEMSTR. This may be important if you've created the file with more than one variable stored in the same record, as you'll have to sort out the contents of the record for yourself after they're dumped into TEMSTR. In a later article, we'll try to give some pointers on how to separate different variables in such a case. We've done that when reading the Catalog file of a disk for our Word Processor, and it's not really difficult.

The final step in file operations is to close the file. The code to accomplish that is shown at CLSF1. We normally recommend that files be closed as soon as you've finished reading or writing them. There are possible exceptions, such as the case of a Fixed record length file in which you want to skip around and read random records. In that case, you can with reasonable safety leave the file open while other functions are performed, then close it when exiting your program.

Trying to close a file that hasn't opened will in general do no harm, but will result in an error. We haven't bothered to show detection or handling of that error. What we recommend is that the error trapping that shows a file has not opened should also cause the program not to try closing it. The adept student will no doubt invent a simple way to do this.

Now that you know how to open a file and read it, you'll easily determine how to open one for output and write to it. The sample code shown at OPNF2 will serve well. Studying that annotated code should give you all you need, so we won't dwell on it here. Also in today's sidebar are PAB setups and mode bytes for D/F 80 type files.

Our final topic for today is that of the special kind of files called Memory Image files, or, as they list in a disk catalog, Program files. The first thing you should know about them is that the name PROGRAM is often a misnomer. In our Word Processor, we use a file called WPCHARACT, which will list on a disk catalog as PROGRAM, but is in reality a character set which we read into VDP directly at >808 to set up character definitions for characters from 1 through 144.

True PROGRAM files, such as those made by the XB SAVE operation or by the TI SAVE utility, have file headers so the computer can detect what kind of files they are, and where they belong in memory. Thus if you try to load in and run an XB program under E/A Option 5, you'll get an error once the E/A loader reads the file header information from the file.

We very often use the dangerous practice of creating memory image files without bothering to place headers on them, since they're used in ways that don't need headers. It isn't really dangerous unless some thoughtless user tries to load them as XB or Option 5. Disaster may then ensue.

We just made a little experiment along that line, and it turns out that, while XB is forgiving, and reports I/O ERROR 50 for our "headerless" files, E/A Option 5 simply goes bonkers given a "program" file that isn't one. Maybe not always, but it just did that in two out of two tries with memory image but non-type 5 files. We must include that subject at more length in a future article, which we plan to subtitle "Off the End of the World".

In Part 7, we showed some source code used in our Word Processor's E/A Option 5 loader, which should serve as a good example of how we can bring our headerless memory image files into memory, then branch into the program placed in memory that way.

In today's source, there's the inverse case, showing how the "saver" part of our Golf Score Analyzer works to save that program into two memory image files. Like the Word Processor, the GSA's object file can only be loaded initially by the CALL LOAD process under Extended Basic. (We suffer the tiresome delay, so our users won't have to.) Once we've done that, we can CALL LINK ("SAVIT"), and thus exercise the code shown at that label in the sidebar. SAVIT first saves our INSTALL program for the GSA, which is part of the code when loaded from the object file by XB. That part gets used as an overlay into the memory normally reserved for user data, and performs steps necessary to install GSA on a Ramdisk. Next, SAVIT takes the part of GSA that resides in Low memory and saves that in a file called DSK1.GOLFCODE, and then takes the part residing in High memory into a file called DSK1.GOLFCODE1. It also tells us on-screen the length of each part as it's being saved. We use that information to update the two loaders that we supply with GSA, one for XB, the other for E/A or TIW. In GSA's normal use, all High memory from >B000 through >FFE6 is set aside for user data.

The setup in the PAB for these memory image operations is very simple. There's no need for record size or file type data. The only bytes that count in the PAB data are the first one, which is 06 for a save and 05 for a load, the third and fourth, which point to the buffer in VDP Ram, and bytes six and seven, which must contain the number of bytes to be saved in the file. From byte 9 onward, things are the same as for any other file type, with the length of the descriptor in byte 9, followed by the descriptor itself.

It is important to transfer the stuff you're saving from its memory location into the buffer in VDP Ram, and to make sure there's not more than the buffer area can handle. If our buffer were at >1020, for example, we could save no more than 10,167 bytes in a file, since any more would overwrite data at VDP Ram address >37D7, which is needed by the computer.

Once that's been done, and the PAB data written to VDP at some lower location than >1020, and the PAB+9 address passed to >8356, just a DSRLNK call makes the file on the disk. No OPEN operation or CLOSE operation is necessary.

The code section we've shown for the Golf Score Analyzer's SAVIT is somewhat convoluted. It also calls a subroutine called INTDI1, which has not been in any of our articles so far. We'll pass that along very soon. The saving code is not saved as part of the main program, but as part of the INSTALL program. This is done so that INSTALL can make changes in the main program itself, then save the two main files to a RAMDISK drive with the modifications in place. The part called GETINS is saved as part of the main program, but is only used when the user performs an installation process. Otherwise, the part at GETINS is overwritten by user data. Okay, so that was clear as mud, but it all works the way it was intended to, and that's what really counts.

Information for dealing with other file types, such as Internal, is available in the E/A Manual in pages 291 through 304. The setup information for byte 1 of the PAB, on page 293, is given bit-by-bit, so one must do some tedious work to figure out what the correct HEX code for that byte should be. We use a hand-held calculator from Radio Shack that converts binary to octal or hex or decimal at the press of a key, and that comes in handy. (Ours is model EC-4030, which is probably out of production by now.)

None of the code in today's source is complete, nor will the whole thing even assemble correctly, but it's a series of pieces that you can use in your own programs. All of what's shown is taken from real programs we've written, and it worked when integrated into those programs.

[These articles are available on disk from the User Group Disk Library, occupying a total of 7 sssd disks. Or ask for a single article by quoting the number at the top.]

The Art Of Assembly Part 9

```
* SOME CODE FRAGMENTS FOR FILE HANDLING OPERATIONS THESE ARE BITS AND PIECES
* BE INTEGRATED INTO PROGRAMS TO PERFORM FILE OPENINGS, READING, WRITING,
* AND CLOSINGS ALL PUBLIC DOMAIN SOURCE CODE
* REQUIRED REFERENCES
       REF  VMBW,VMBR,VSBW,VSBR
       REF  DSRLNK
* REQUIRED EQUATES
STATUS EQU  >837C
WS     EQU  >20BA
GPLWS  EQU  >83E0
PAB1   EQU  >1000
BUF    EQU  >1050
PABPNT EQU  >8356
* CODE AT LABEL START COULD BE USED TO GET A FILE NAME INPUT FROM THE
* USER.  IT USES SUBROUTINES WE'VE SUPPLIED PREVIOUSLY
START  LI   R15,RTNSTK   SET STACK FOR HIGH LEVEL SUBROUTINE
       LI   R0,3         ROW 1, COLUMN 4
       LI   R4,15        15 CHARACTERS WILL BE ACCEPTED
       BL   CRSIN       USE CRSIN SUBROUTINE
       LI   R9,TEMSTR    POINT AT TEMPORARY STRING
       LI   R10,PAB1DT+9 POINT R10 AT FILE DESCRIPTOR LENGTH BYTE
       BL   MOVSTR      MOVE STRING FROM TEMSTR TO PAB DATA
*  TWO WAYS TO OPEN ARE SHOWN
*  USE ONLY ONE OF THESE, DEPENDING ON YOUR NEED
*  FIRST IS THE LONG AND ACCURATE METHOD SECOND IS A SHORTCUT
OPNF1  MOVB INMD,PAB1DT+1 OPEN WILL BE INPUT MODE
       LI   R0,PAB1      SET WRITE ADDRESS IN R0
       MOVB PAB1DT+9,R2 GET DESCRIPTOR LENGTH BYTE INTO LEFT BYTE R2
       SRL  R2,8         RIGHT JUSTIFY SO R2 IS A WORD OF LENGTH
       AI   R2,10        ADD 10 TO INCLUDE THE PAB1DT LINE PLUS DESCRIPTOR
       LI   R1,PAB1DT    POINT R1 AT PAB DATA
       BLWP VMBW        WRITE BYTES TO PAB LOCATION IN VDP RAM
       AI   R0,9         ADD NINE TO ADDRESS IN R0
       MOV  R0,PABPNT   PLACE THAT ADDRESS AT >8356
       CLR  STATUS      CLEAR GPL STATUS
       BLWP DSRLNK      USE DSRLNK UTILITY
       DATA 8            REQUIRED DATA
       STST R14          STORE STATUS REGISTER IN R14
       ANDI R14,>2000    MASK ALL BUT BIT #2 IN R14
       JEQ  RDF1         IF ZERO, GO AHEAD TO READ FILE
       B    OPNERR      ELSE TO OPNERR (SHOWN IN LAST ARTICLE)
* SHORTCUT METHOD FOR FILES OTHER THAN HARD DISK OR RS232 TYPE
OPNF1  MOVB INMD,PAB1DT+1 OPEN WILL BE INPUT MODE
       LI   R0,PAB1      SET WRITE LOCATION
       LI   R1,PAB1DT    SET SOURCE FOR PAB DATA
       LI   R2,25        25 BYTES - MAX FOR MOST PURPOSES
       BLWP VMBW        WRITE DATA TO VDP
       AI   R0,9         ADD NINE
       MOV  R0,PABPNT   MOVE TO >8356
       CLR  STATUS      CLEAR STATUS
       BLWP DSRLNK      USE LINKAGE VECTOR
       DATA 8            REQUIRED DATA
       STST R14          STORE STAUS REGISTER IN R14
       ANDI R14,>2000    MASK ALL EXCEPT BIT 2
       JEQ  READF1       IF ZERO, PROCEED TO READ
       B    OPNERR      OPNERR SHOWN LAST ARTICLE
RDF1   MOVB READF,R1    MOVE READ OPCODE INTO LEFT BYTE R1
       LI   R0,PAB1      PAB ADDRESS IN VDP
       BLWP VSBW        WRITE ONE BYTE INTO PAB
       AI   R0,9         ADD NINE
       MOV  R0,PABPNT   MOVE TO >8356
       CLR  STATUS      CLEAR GPL STATUS
       BLWP DSRLNK      USE DSRLNK
       DATA 8            REQUIRED DATA
       LI   R0,PAB1+1    SET TO SECOND BYTE OF PAB IN VDP
```

The Art Of Assembly Part 9

```
       BLWP VSBR        READ INTO LEFT BYTE R1
       SRL  R1,13        SHIFT R1 RIGHT BY 13 BITS
       JEQ  READON       IF ZERO, NO ERROR IN DSR OPERATION
       CI   R1,5         IF ERROR = 5, END OF FILE HAS BEEN REACHED
       JEQ  CLSF1        IF SO, CLOSE THE FILE
       B    FILERR      ELSE SOME OTHER ERROR, REPORT THAT TO USER
READON LI   R0,PAB1+5    POINT AT PAB+5 IN VDP RAM
       BLWP VSBR        READ THAT BYTE INTO LEFT BYTE R1
* FOR D/V FILES, THE BYTE AT PAB+5 IS THE LENGTH OF THE RECORD JUST READ
       MOVB R1,R2        MOVE BYTE INTO R2
       SRL  R2,8         RIGHT JUSTIFY LENGTH IN R2
       MOVB R1,TEMSTR   MOVE BYTE TO TEMSTR
       LI   R0,BUF       POINT TO BUFFER LOCATION IN VDP
       LI   R1,TEMSTR+1  CONTENT GOES TO TEMSTR+1
       BLWP VMBR        READ CONTENT OF RECORD FROM VDP BUFFER
* CODE WOULD BE INSERTED HERE TO MOVE THE RECORD FROM TEMSTR, OR DISPLAY
* THE RECORD ON THE SCREEN, OR ANY OTHER OPERATION YOU DESIRE
       JMP  RDF1         JUMP BACK TO READ NEXT RECORD
CLSF1  LI   R0,PAB1      POINT TO PAB ADDRESS
       MOVB CLOSEF,R1   GET CLOSE OPCODE IN LEFT BYTE R1
       BLWP VSBW        WRITE OPCODE TO PAB
       AI   R0,9         ADD NINE
       MOV  R0,PABPNT   PLACE AT >8356
       CLR  STATUS      CLEAR STATUS
       BLWP DSRLNK      CALL DSRLNK
       DATA 8            REQUIRED DATA
* FROM HERE, GO ON TO NEXT PROGRAM OPERATION
* CODE BELOW OPENS A D/V 80 FILE FOR WRITING, THEN WRITES THE RECORD TASHED
* AT TEMSTR TO THE FILE
OPNF2  MOVB OUTMD,PAB1DT+1 OPEN WILL BE OUTPUT MODE
       LI   R0,PAB1      SET WRITE ADDRESS IN R0
       MOVB PAB1DT+9,R2 GET DESCRIPTOR LENGTH BYTE INTO LEFT BYTE R2
       SRL  R2,8         RIGHT JUSTIFY SO R2 IS A WORD OF LENGTH
       AI   R2,10        ADD 10 TO INCLUDE THE PAB1DT LINE PLUS DESCRIPTOR
       LI   R1,PAB1DT    POINT R1 AT PAB DATA
       BLWP VMBW        WRITE BYTES TO PAB LOCATION IN VDP RAM
       AI   R0,9         ADD NINE TO ADDRESS IN R0
       MOV  R0,PABPNT   PLACE THAT ADDRESS AT >8356
       CLR  STATUS      CLEAR GPL STATUS
       BLWP DSRLNK      USE DSRLNK UTILITY
       DATA 8            REQUIRED DATA
       STST R14          STORE STATUS REGISTER IN R14
       ANDI R14,>2000    MASK ALL BUT BIT #2 IN R14
       JEQ  WRTF2        IF ZERO, GO AHEAD TO WRITE FILE
       B    OPNERR      ELSE TO OPNERR (SHOWN IN LAST ARTICLE)
WRTF2  MOVB TEMSTR,R1   GET LENGTH OF RECORD IN LEFT BYTE R1
       LI   R0,PAB1+5    POINT TO RECORD LENGTH BYTE OF PAB
       BLWP VSBW        WRITE LENGTH TO PAB
       MOVB R1,R2        PLACE LENGTH IN LEFT BYTE R2
       SRL  R2,8         RIGHT JUSTIFY LENGTH IN R2
       LI   R1,TEMSTR+1  POINT TO STRING CONTENT
       LI   R0,BUF       POINT AT BUFFER IN VDP
       BLWP VMBW        WRITE RECORD CONTENTS TO VDP
       MOVB WRITEF,R1   GET WRITE OPCODE IN R1
       LI   R0,PAB1      POINT TO START OF PAB
       BLWP VSBW        WRITE THE OPCODE BYTE TO VDP
       AI   R0,9         ADD 9
       MOV  R0,PABPNT   MOVE TO >8356
       CLR  STATUS      CLEAR GPL STATUS BYTE
       BLWP DSRLNK      CALL DSR LINKAGE
       DATA 8            REQUIRED DATA
       LI   R0,PAB1+1    POINT TO SECOND BYTE OF PAB
       BLWP VSBR        READ THAT BYTE INTO R1
       SRL  R1,13        SHIFT R1 RIGHT 13 BITS
       JEQ  WRTON        IF ZERO, NO ERROR, SO GO ON
       B    FILERR      ELSE BRANCH TO ERROR HANDLING
```

The Art Of Assembly Part 9

```
WRTON
* THIS LABEL WOULD GET ANOTHER RECORD READY AT TEMSTR, THEN BRANCH BACK TO
* WRTF2, OR ELSE IF FINISHED WOULD BRANCH BACK TO CLSF1 TO CLOSE THE FILE
* THIS FILE, SINCE IT'S USING THE SAME PAB AND BUFFER, CAN'T BE OPEN WHILE
* FILE 1 IS ALSO OPEN
* REQUIRED DATA SECTION
* THE FOLLOWING DATA SOURCE LINES ARE REQUIRED
* THIS PAB DATA AND MODE BYTES APPLY TO D/V 80 FILES
PAB1DT DATA >0014,BUF,>5000,>0000,>000F
       BSS  15
INMD   BYTE >14          BYTE FOR INPUT OF DISPLAY/VARIABLE FILE
OUTMD  BYTE >12          BYTE FOR OUTPUT OF DISPLAY/VARIABLE FILE
APPMD  BYTE >16          BYTE FOR APPEND OF DISPLAY/VARIABLE FILE
UPDAMD BYTE >10          BYTE FOR UPDATE MODE OF D/V FILE -NOT RECOMMENDED
WRITEF BYTE 3            OPCODE FOR WRITE OPERATION
READF  BYTE 2            OPCODE FOR READ OPERATION
CLOSEF BYTE 1            OPCODE FOR CLOSE OPERATION
* THE DATA BELOW IS A PAB SETUP PLUS THE MODE BYTES
* FOR A D/F 80 FILE
PAB2DT DATA >0001,BUF,>5050,>0000,>000F
       BSS  15
FINMD  BYTE >05          INPUT MODE BYTE FOR DISPLAY/FIXED FILE TYPE
FOUTMD BYTE >03          OUTPUT MODE BYTE FOR D/F FILES
FAPPMD BYTE >07          APPEND MODE BYTE FOR D/F FILES
FUPDMD BYTE >01          UPDATE MODE BYTE FOR D/F FILES
* BELOW IS CODE AND DATA SAMPLE FOR MAKING MEMORY IMAGE FILES
* DERIVED FROM OUR GSA PROGRAM
       AORG >B000        THIS CODE STARTS AT >B000
INSTDT DATA >0500,>1020,0,ENINST-SAVDT,>000C
       TEXT 'DSK1.INSTALL'
SAVBYT BYTE 6
LDBYTE BYTE 5
* THIS SECTION IS USED TO LOAD THE INSTALL CODE WHEN NECESSARY
GETINS
       MOVB LDBYTE,INSTDT PUT LOAD OPCODE IN FIRST BYTE OF PAB DATA BLOCK
       LI   R0,PAB1      POINT TO PAB LOCATION
       LI   R1,INSTDT    POINT R1 AT DATA BLOCK
       LI   R2,22        22 BYTES IN PAB
       BLWP VMBW        WRITE TO VDP
       AI   R0,9         ADD 9
       MOV  R0,PABPNT   TO >8356
       CLR  STATUS      CLEAR STATUS
       BLWP DSRLNK      LOAD FILE INTO VDP BUFFER
       DATA 8            REQD DATA
       LI   R0,>1020     POINT AT BUFFER
       MOV  INSTDT+6,R2 LENGTH OF FILE INTO R2
       LI   R1,SAVDT     FILE STARTS AT LOCATION SAVDT
       BLWP VMBR        GET CODE INTO MEMORY
       B    INSTAL      THEN BRANCH TO NOW-INSTALLED INSTALL CODE
ENDAUX EQU  $
*  SAVER - STASHES PROGRAM AS MEMORY IMAGE FILE 15 JUN 89
       DEF  SAVIT        DEFINED ENTRY POINT
SAVDT  DATA >0600,>1020,0,ENMAIN-GPLLNK,>000D
       TEXT 'DSK1.GOLFCODE'
SAVDT1 DATA >0600,>1020,0,ENDAUX->A000,>000E
       TEXT 'DSK1.GOLFCODE1'
SAVIT
       MOV  R11,>8300   STASH REGISTER 11
       LWPI WS           LOAD OUR WS
       LI   R15,RTNSTK   SET R15 TO RETURN ADDRESS
       BL   CLS         CLEAR SCREEN
       LI   R9,INSTDT+9  DISPLAY FILE DESCRIPTOR
       LI   R0,SCRWID*5+4 AT ROW 6, COLUMN 5
       BL   DISLI       USING SUBROUTINE
       MOV  INSTDT+6,R5 BRING LENGTH OF INSTALL SECTION INTO R5
       LI   R0,SCRWID*7+15 SCREEN LOCATION ROW 8, COLUMN 16
       BL   INTDI1       DISPLAY INTEGER NUMBER ON SCREEN
       MOVB SAVDT,INSTDT SET FOR SAVING INSTALL PART
       MOV  INSTDT+6,R2 GET LENGTH IN R2
       LI   R1,SAVDT      POINT TO START OF CODE TO BE SAVED
       LI   R0,>1020      BUFFER ADDRESS
       BLWP VMBW         WRITE TO BUFFER
       LI   R0,PAB1       SET TO WRITE PAB
       LI   R2,22         22 BYTES
       LI   R1,INSTDT     PAB DATA FOR INSTALL SECTION
       BLWP VMBW         WRITE TO PAB IN VDP RAM
       AI   R0,9          ADD 9
       MOV  R0,PABPNT    AT >8356
       CLR  STATUS       CLEAR
       BLWP DSRLNK       PERFORM WRITING OF FILE INSTALL TO DSK1
       DATA 8             DATA
SAVPT2
       LI   R9,SAVDT+9    GET DESCRIPTOR FOR GOLFCODE FILE
       LI   R0,SCRWID*9+4 ROW 10, COLUMN 5
       BL   DISLI        DISPLAY FILE DESCRIPTOR
       MOV  SAVDT+6,R5   GET LENGTH OF LOW-MEMORY CODE SECTION IN R5
       LI   R0,SCRWID+15  ROW 12, COLUMN 16
       BL   INTDI1       DISPLAY INTEGER
       LI   R0,>1020      POINT TO BUFFER
       LI   R1,GPLLNK     GPLLNK IS AT START OF PROGRAM'S LOW MEM PORTION
       LI   R2,ENMAIN-GPLLNK ENMAIN IS END OF LOW MEM PART OF CODE
       BLWP VMBW         WRITE TO BUFFER
       LI   R0,PAB1       SET FOR PAB
       LI   R1,SAVDT      POINT TO DATA
       LI   R2,23         23 BYTES
       BLWP VMBW         WRITE PAB TO VDP
       AI   R0,9          ADD 9
       MOV  R0,PABPNT     TO >8356
       CLR  STATUS
       BLWP DSRLNK       WRITE FIRST SECTION OF CODE TO FILE
       DATA 8
       LI   R9,SAVDT1+9   GET DESCRIPTOR FOR SECOND FILE
       LI   R0,SCRWID+4  SCREEN ROW 16, COLUMN 5
       BL   DISLI        DISPLAY THE DESCRIPTOR
       MOV  SAVDT1+6,R5 GET LENGTH OF HIGH MEMORY SECTION IN R5
       LI   R0,SCRWID+15   ROW 18, COLUMN 16
       BL   INTDI1       DISPLAY LENGTH (AS DECIMAL NUMBER)
       LI   R0,>1020      POINT TO BUFFER
       LI   R1,>A000      START OF HIGH MEMORY
       LI   R2,ENDAUX->A000 LENGTH OF HIGH MEM PORTION
       BLWP VMBW         WRITE INTO BUFFER
       LI   R0,PAB1       SET FOR PAB
       LI   R1,SAVDT1     POINT TO PAB DATA
       LI   R2,24         24 BYTES TO WRITE
       BLWP VMBW         WRITE PAB TO VDP
       AI   R0,9          ADD 9
       MOV  R0,PABPNT
       CLR  STATUS
       BLWP DSRLNK       WRITE FILE GOLFCODE1 TO DISKDSRLNK
       DATA 8
GEXIT  LWPI GPLWS         LOAD GPL WORKSPACE
       MOV  >8300,R11    REPLACE R11
       RT                 RETURN
INSTAL
* THE CODE FOR THE INSTALLATION PROCESS FOLLOWS HERE (NOT SHOWN)
* IT ENDS AT A LABEL CALLED ENINST
TEMSTR BSS  81            TEMPORARY STORAGE LOCATION FOR RECORD
* THE NUMBER IN THIS BSS MUST BE ONE MORE THAN THE LARGEST STRING LENGTH
* EXPECTED IN THE PROGRAM'S EXECUTION
RTNSTK DATA 0             RETURN ADDRESS STACK NEEDED BY CRSIN
```

John Phillips

I would venture a guess that most people who have owned a TI-99 for more than a couple of years have run across the name John Phillips before. He is a near legend in the TI-99/4A cartridge and assembly language programming community and can claim authorship, co-authorship or significant involvement in over a dozen cartridge programs produced for the 99/4A, not to mention numerous articles written about the inner workings of the 4A's architecture.

Phillips will be 32 years old this year (1993) but he was only 21 when he was hired by Texas Instruments in 1982 right after graduating from Illinois State University. He started his career with TI in Dallas doing COBAL programming for business applications but it took him only 6 months to get a requested transfer to Lubbock where the "real" action was. John had purchased a 99/4 during his senior year in college and was already familiar with the Home Computer's architecture and he had wanted to program video games since purchasing his first cartridge, which was Munchman. Phillips didn't know TMS9900 assembly language but it didn't take him long to learn it.

His first project at Lubbock was Moonmine, followed by Hopper, which he co-authored with Michael Archulata. Hopper was followed by Word Radar, which he wrote in 2 weeks, for Developmental Learning Materials (DLM), the firm started by Bill Maxwell and Jerry Chaffin.

After completing Word Radar TI sent Phillips to Japan where he met with several companies who were being recruited to write software for the 99/4A. Following his return from Japan he became involved in almost every piece of software that was slated for production or that was actually produced for the 99/4A. When TI announced the end of the Home Computer Division Phillips was offered several incentives to stay at TI but turned them all down because none involved work with the 99/4A. Instead, he and fellow employee Michael Archuleta went to work for DLM, which had continued to work on products for the TI-99/4A even though it was no longer being produced.

In December 1983 John Phillips announced to the TI Community that he was available to any User Group for seminars, demonstrations and question and answer sessions related to the TI-99/4A. He would travel to virtually any location if the User Group would pay round trip airfare from Dallas, Texas plus lodging? While he could only make himself available on weekends, it was a pretty generous offer.

Both Phillips and Archuleta eventually left DLM (probably because the work there dried up too) and started their own firm in February 1984 called Video Magic. Video Magic also came to an end in too short a time, I suspect because it was becoming painfully obvious that one could not make a living trying to write software for the 99/4A.

At Texas Instruments Michael Archuleta was responsible for the 99/4A Technical Hotline and for 99/4A software quality assurance. Phillips was a third-party software development consultant and programmer in the education/entertainment section of the Consumer Products Division. Both men would get together again in 1986 to collaborate on the 4A FLYER game cartridge that was commissioned by Triton Products. To date, that is the last time we've heard from the John Phillips/Michael Archulata team.

Archuleta and Phillips were involved in, or responsible for such TI-99 favorites as:

Angler Dangler

Phillips worked on this project as the debugger of the final code, but the project never reached completion before the bailout so Angler Dangler was never officially released. It does exist in GRAM file format however, so it probably was not too far from being a real product when someone at TI made the decision to pull the plug. If you look at the October 23, 1983 IUG price list you will see Angler Dangler listed as being available.

Beyond Parsec

This cartridge, which Bill Moseid's DataBiotics firm released for the 99/4A during the third quarter of 1988, started life in early 1984 as one of two game cartridges John Phillips was writing for CorComp's new CCI-99/64 (aka Phoenix) computer. The other game was Star Wars. Both efforts came to a screaching halt however, when TI objected to the use of the Parsec name, and George Lucas' company apparently objected to the use of the trademarked Star Wars name. The Star Wars code must have actually been finished at the time though, because I have the game on disk as a GPL file. It was ultimately renamed Star Trap and released in cartridge form by Exceltec in 1985 and then by DataBiotics during the third quarter of 1988.

Beyond Space

This is a John Phillips creation that was completed in May 1984, but not released until the first quarter of 1985 when Exceltec/Sunware marketed it. It was picked up by Unisource Electronics for their catalog/encyclopedia but pretty much floundered and then just disappeared.

Burgertime

Phillips provided the final debugging for Burgertime.

D Station

This John Phillips creation has the distinction of being the only program ever released by the International 99/4 User Group on the Romox ECPC cartridge. When the IUG ECPC library failed Exceltec (aka Sunware) picked up the program and marketed it for a short time in 1985. Triton finally introduced D Station in their Fall 1988 catalog along with a brand new D Station II game, also written by John Phillips.

Facemaker

Phillips collaborated with Intersoft's Jerry Spacek on this project. Spacek you may recall wrote Defend the Cities, which was the first commercial Mini-Memory assembly language game ever written. In the Facemaker project Spacek translated Spinnaker's source code to TMS9900 assembly language and Phillips ported it to cartridge format.

Hopper

Michael Archuleta and John Phillips co-wrote Hopper, which was the only cartridge developed entirely on the TI-99/4A Home Computer, using the Editor/Assembler cartridge for all of the programming. All of the other TI-99 cartridge software programs were developed on a TI Mini, not the 99/4 or 4A.

Jawbreaker II

Phillips converted the original Sierra On-Line source code to TI-99/4A code.

Moonmine

Programmed by John Phillips from a design by Bob Hendren. You may remember that Hendren was also the project engineer behind Parsec and the person who recruited Aubree Anderson to do the voice for the Parsec game.

Peter Pan's Space Odyssey

Phillips and Archuleta collaborated on this program while employed at DLM. It was never officially released but is available as a GRAM file.

Slymoids

Slymoids was written by James R. Von Ehr II. The cartridge conversion was accomplished by John Phillips.

Super Demon Attack

Phillips worked on this project, but I have no information on the specific contributions he made to its completion other than possible debugging of the final code.

The Great Word Race & Word Radar

John Phillips authored.

Treasure Island

Phillips provided the final debugging for this game cartridge, which had apparently become stalled by a bug that no one could find.

More Tips...

Thanks to Steve Chapman and Bill Wallbank of Stone & Webster Engineering Corp. TIUG for this one. If V=21 you are in Extended Basic, otherwise you are in Basic. I am not sure it will work with all consoles and modules.

```
100 RANDOMIZE (0)
110 V=INT(RND*100)
```

How can you input a blank (CHR$ 32) with ACCEPT AT? As far as I know, you can't. With LINPUT, just hit the space bar, and with INPUT, type " ". But with ACCEPT AT the space bar gives a null string and " " gives " " ! However, you can code around it -

```
X$=CHR$(34)&CHR$(32)&CHR$(32):
: ACCEPT AT(1,1):T$ :: IF T$=
X$ THEN T$=CHR$(32)
```

And, to clear up the puzzling behavior of the "quote marks" -

```
100 CALL CHARPAT(34,CH$):: CAL
L CHAR(35,CH$)!written by Jim
Peterson
110 DISPLAY AT(1,7)ERASE ALL:"
THE # PUZZLE":" You can't ente
r PRINT # or PRINT ### - the c
omputer demands an even number
 of #."
120 DISPLAY AT(5,1):"1 PRINT
## !prints a null string (noth
ing)":"2 PRINT #*# !prints *"
130 DISPLAY AT(8,1):"3 PRINT #
### !prints #":"4 PRINT ##*##
!crashes as  STRING-NUMBER MI
SMATCH"
140 DISPLAY AT(11,1):"5 PRINT
##**## !crashes as  SYNTAX ERR
OR"
150 DISPLAY AT(13,1):"6 PRINT
###### !prints ##":"7 PRINT ##
#*### !prints #*#":"8 PRINT ##
#**### !print #**#"
160 DISPLAY AT(16,1):"9 PRINT
######## !prints ###":"10 PRIN
T ####*#### !crashes as STRING
-NUMBER MISMATCH"
170 DISPLAY AT(19,1):"11 PRINT
 ####**#### !crashes as SYNTAX
 ERROR":"12 PRINT ########## !
####"
180 DISPLAY AT(22,1):"13 PRINT
 #####*##### !##*##":"14 PRINT
 #####**#####!##**##"
190 DISPLAY AT(24,1):"TRY IT!
LINE NO.(1-14)?" :: ACCEPT AT(
24,25)VALIDATE(DIGIT)SIZE(2)BE
EP:LN :: IF LN<1 OR LN>14 THEN
 190
200 CALL CLEAR :: ON LN GOSUB
230,240,250,260,280,290,300,31
0,320,330,340,350,360,370
210 PRINT :;:;:"Press any key"
220 CALL KEY(0,K,S):: IF S=0 T
HEN 220 ELSE 110
230 PRINT "" :: RETURN
240 PRINT "*" :: RETURN
250 PRINT """" :: RETURN
260 PRINT ""*"" !crashes as ST
RING-NUMBER MISMATCH - the * i
s misinterpreted as a multipli
er!Same with +,-,/
270 !with anything else, inclu
ding numerals, crashes as SYNT
AX ERROR - but inserts a space
 before the character!
280 PRINT ""**"" :: !crashes
290 PRINT """""" :: RETURN
300 PRINT """*""" :: RETURN
310 PRINT """**""" :: RETURN
320 PRINT """""""" :: RETURN
330 PRINT """"*"""" !crash
340 PRINT """"**"""" !crash
350 PRINT """""""""" :: RETURN
360 PRINT """""*""""" :: RETUR
N
370 PRINT """""**""""" :: RETU
RN
```

This one is just for the fun of it - it uses the contents of computer memory to create designs -

```
100 DISPLAY AT(3,10)ERASE ALL:
"COLORPEEK": :TAB(7);"by Jim P
eterson": : :" Watch the compu
ter's memory": :"displayed in
color."
110 DISPLAY AT(12,1):"Choose":
 :"(1) plain colors": :"(2) ba
rs & checks": :"(3) patterns"
:: ACCEPT AT(12,8)VALIDATE("12
3")SIZE(1):Q :: CALL CLEAR ::
IF Q=1 THEN 170
120 DISPLAY AT(12,5):"wait,ple
ase" :: IF Q=3 THEN 140
130 FOR CH=32 TO 143 :: CALL C
HAR(CH,RPT$("F0",8)):: NEXT CH
 :: GOTO 160
140 RANDOMIZE :: FOR CH=32 TO
88 :: FOR J=1 TO 4 :: X$=SEG$(
"0018243C425A667E8199A5BDC3DBE
7FF",INT(16*RND+1)*2-1,2):: B$
=B$&X$ :: C$=X$&C$ :: NEXT J :
: CALL CHAR(CH,B$&C$)
150 CALL CHAR(CH+55,B$&C$):: B
$,C$="" :: NEXT CH
160 FOR SET=0 TO 14 :: CALL CO
LOR(SET,SET+1,16-SET):: NEXT S
ET :: CALL SCREEN(2):: GOTO 18
0
170 FOR SET=0 TO 14 :: CALL CO
LOR(SET,SET+2,SET+2):: NEXT SE
T :: CALL SCREEN(16)
180 FOR J=-1 TO -2000 STEP -1
:: CALL PEEK(J,A):: A=A-(A<33)
*(A+32):: A=A+(A>143)*(A
/2):: R=R+1+(R=24)*24 :: CALL
HCHAR(R,1,A,32)
190 C=C+1+(C=32)*32 :: CALLVCH
AR(1,C,A,24):: NEXT J :: GOTO
100
```

Unlike most of the number games played against the computer, you can win this one -

```
100 CALL CLEAR :: CALL SCREEN(
16):: DISPLAY AT(3,8):"THE `37
' GAME" !by Jim Peterson
110 DISPLAY AT(5,1):" We will
take turns picking":"a number
from 1 to 5, but":"not the num
ber that was just":"picked."
120 DISPLAY AT(10,1):" The num
bers we pick will be":"added t
o the total count."
130 DISPLAY AT(13,1):" Whoever
 reaches 37 is the":"winner, b
ut if you go over":"37 you los
e."
140 CALL SHOW(20,1,"Press any
key to start")
150 CALL KEY(0,K,S):: IF S=0 T
HEN 150
160 DATA 4,11,17,24,30,37
170 DATA 262,330,392,523,523
180 DATA 1047,784,659,523,523
190 C,P=0 :: CALL CLEAR :: CAL
L MAGNIFY(2):: R=10 :: FOR J=1
 TO 5 :: CALL SPRITE(#J,48+J,5
,R,10):: R=R+30 :: NEXT J
200 CALL SHOW(24,1,"(Y)ou or (
C)omputer first?"):: ACCEPT AT
(24,28)VALIDATE("YC")SIZE(1):Q
$ :: DISPLAY AT(24,1):""
210 IF Q$="C" THEN CALL SHOW(2
2,8,"I pick 4"):: CALL COLOR(#
4,1):: P=4 :: C=4 :: CALL SHOW
(3,10,"COUNT=4")
220 CALL SHOW(20,8,"Pick your
number"):: ACCEPT AT(20,26)VAL
IDATE("12345"):N :: IF N=P THE
N 220
230 IF P>0 THEN CALL COLOR(#P,
5)
240 CALL COLOR(#N,1):: P=N ::
C=C+N :: CALL SHOW(3,10,"COUNT
= "&STR$(C)):: IF C=37 THEN 32
0 ELSE IF C>37 THEN 340
250 RESTORE 160
260 READ X :: IF C<X THEN B=X-
C ELSE IF X<37 THEN 260
270 CALL SHOW(22,8,"I'm thinki
ng..."):: FOR Y=1 TO 700 :: NE
XT Y
280 IF B>5 AND B/2=INT(B/2)THE
N B=B/2
290 IF B>5 OR B=P THEN B=1-(P=
1)
300 CALL SHOW(22,8,"I pick "&S
TR$(B)):: CALL COLOR(#P,5):: C
ALL COLOR(#B,1):: P=B :: C=C+B
 :: CALL SHOW(3,10,"COUNT= "&S
TR$(C))
310 IF C=37 THEN 340 ELSE IF C
>37 THEN 320 ELSE 220
320 RESTORE 170 :: FOR J=1 TO
5 :: READ F :: CALL SOUND(100,
F,5,F*1.03,5):: NEXT J :: CALL
 SHOW(12,8,"YOU WIN!")
330 CALL SHOW(15,8,"Play again
? (Y/N)"):: ACCEPT AT(15,26)VA
LIDATE("YN"):Q$ :: IF Q$
="N" THEN STOP ELSE 190
340 RESTORE 180 :: FOR J=1 TO
5 :: READ F :: CALL SOUND(300,
30000,30,30000,30,F,30,-4,5)::
 NEXT J :: CALL SHOW(12,8,"YOU
 LOSE!"):: GOTO 330
350 SUB SHOW(R,C,T$):: FOR J=1
 TO 10 :: DISPLAY AT(R,C):" "
:: DISPLAY AT(R,C):T$ :: NEXT
J :: SUBEND
```

A couple more peculiarities of the computer

```
100 DISPLAY AT(3,8)ERASE ALL:"
POS PUZZLE #1": :"        from
Tigercub"
110 DISPLAY AT(9,1):"Why does
the computer say":"that X=1 if
 you answer the":"prompt with
the Enter key":"(null-string)
?"
120 DISPLAY AT(14,1):"110 INPU
T M$"
130 DISPLAY AT(15,1):"120 X=PO
S(""TESTING"",M$,1)::":"PRINT
X :: GOTO 100"
140 !POS PUZZLE #1 - why does
the computer say that X=1
if you answer the prompt with
Enter (null-string) ? - Jim Pe
terson
150 INPUT M$
160 X=POS("TESTING",M$,1)::PRI
NT X :: GOTO 140
```

```
100 DISPLAY AT(3,8)ERASE ALL:"
POS PUZZLE #2": :"        from
Tigercub"
110 DISPLAY AT(7,1):"Why does
the computer say":"that the fi
rst position of":"null-string
is at whatever":"position it i
s told to start":"search at?"
120 DISPLAY AT(13,1):"100 M$="
""""
130 DISPLAY AT(14,1):"110 DISP
LAY AT(20,1):""POS?"" :: ACCEP
T AT(20,6):P"
140 DISPLAY AT(16,1):"120 X=PO
S(""TESTING"",M$,P):: DISPLAY
AT(22,1):""X="";X :: GOTO 110"
150 M$=""
160 DISPLAY AT(21,1):"POS?"::
ACCEPT AT(21,6):P
170 X=POS("TESTING",M$,P)::DIS
PLAY AT(23,1):"X=";X :: GOTO 1
60
```

Irwin Hott informs me that assembly routines which have been imbedded into Xbasic programs, using ALSAVE or SYSTEX, can be saved to cassette and reloaded. This could be very useful for those who have a stand-alone or "matchbox" 32k.

And, a mini-game for you to have fun with or improve on -

```
1 !      2-LINE GAME by Jim Pe
terson- use S&D keys to paint
thewhite line on the highway
2 !if it is too easy, change t
he 6 in A$=RPT$(CHR$(143),6) t
o 5 and the 5 in C>T+5 to 4
100 CALL CLEAR :: A$=RPT$(CHR$
(143),6):: CALL COLOR(14,2,2,2
,16,16):: CALL SCREEN(4):: T=1
1 :: C=14 :: CALL HCHAR(22,C+2
,42):: RANDOMIZE
110 T=T+INT(3*RND-1)+(T=21)-(T
=1):: PRINT TAB(T);A$ :: CALL
KEY(3,K,S):: C=C+(K=83)-(K=68)
:: CALL HCHAR(22,C+2,42):: IF
C<T OR C>T+5 THEN STOP ELSE 11
0
```

And finally, one of the best examples of compact programming I have ever seen -

```
1 !JOHN WITTE'S 3-LINE VERSION
 OF JOHN WILLFORTH'S WAVEPOWER
 - PUBLISHED IN GREATER OMAHA
UG NEWSLETTER
100 CALL CLEAR :: A$(1)="ABCDE
FGFEDCBA" :: FOR I=1 TO 7:: CA
LL CHAR(72-,RPT$("0",2*I-2)&"F
FFF",47,"30303EFF7F3E1E04")::
A$(I+1)=SEG$(A$(I),2,12)&SEG$(
A$(I),2,1):: NEXT I
110 CALL SPRITE(#5,47,2,180,18
0,-23,0,#6,47,2,80,100,-23,0):
: CALL MAGNIFY(2)
120 FOR I=1 TO 12 :: PRINT A$(
I+(I>7)*2*(I-7))&A$(1+I+(I>6)*
2*(I-6)):: NEXT I :: GOTO 120
```

Memory full
Jim Peterson

Command Module Emulator

The following notice appears in the "99ER DIGEST" portion of the September 1983 issue of 99ER HOME COMPUTER MAGAZINE (p. 53):
"FAIRS AND AMUSEMENT PARKS GO COMPUTER. TIs Consumer Group will be represented nationwide at 14 state fairs this summer and fall.

Exhibits featuring the 99/4A will reach an expanded market with TI's Product Service Representatives demonstrating educational, entertainment, and information management applications in Arizona, California, indiana, Kentucky, Louisiana, Minnesota, New Mexico, New york, North Carolina, Ohio, Oklahoma, Texas, washington, and Wisconsin.

In addition, Six Flags Magic Mountain amusement park in Valencia CA will feature a 3000 square foot Computer Discovery Center sponsored by TI. Forty 99/4A systems and a short Cosby-narrated film will be exhibited."

It turns out that "Service Representatives" at all these events were supplied by TI with a COMMAND MODULE SIMULATOR.

This unique hardware was different than the PE box. It had a couple of disk drives and came equipped with 22 disks. The disks were intitialized DSDD using the 8 sector per track format found on TI's never officially released DSDD disk controller card for the PE box. EVERY COMMAND MODULE ever released or anticipated for future release was on these disks, encoded in a format unique to the COMMAND MODULE SIMULATOR.

The disks could only be run using the sumulator. TI "Service Representatives" could call up ANY module for members of the public to play with.

Frank Bubenik, Lima UG member and secretary/editor of the Long Island TI UG, reports that one of the LITI members purchased one of these COMMAND MODULE SIMULATORs recently at a flea market along with all 22 disks. Software on the disks apparently includes ALL cartridge software officially released or under development directly by TI or under a TI license (software copyrighted to others but manufactured and distributed by TI).

All of the "never released official TI modules" I have described in the past (such as Wing Wars, Von Drake, Music SDA, Germ Patrol, etc.) are included on the COMMAND MODULE SIMULATOR disks, as well as additional unreleased education titles, and some officially released in very limited quantities educational software by Scott Foresman and Addison-Wesley containing a 1983 copyright.

This rare or never released educational software includes these titles:

Computer Math Games 1 3 and 4
Star Maze; Numeration 1 and 2
Addition & Subtraction 3
Fantastic Fractions 1
Decimal Deli 2
Reading
Cheers
Wonders
Adventures
Rainbows
Power
Flight
Trail
Pyramid Puzzler
Picture Parts
Space Journey

This is really high quality educational software by some of this country's best names in public school publishing. Most cartridges make good use of the "bells and whistles" available on the 99/4A (sound, graphics, speech), much more so than PLATO software designed to run from the /4A.

Rare TI Modules

The word RARE in this series of articles refers to command module cartirdges that were produced in limited quantities and, for the most part, only sold to the public after TI announced they were getting out of the home computer market. This education software all bears 1983 copyrights by Addison Wesley and Scott Foresman. TI was to handle the manufacturing and distribution, but very few of these cartridges ever made it to retail store shelves, and you won't find many of them commonly available at any price from today's dealers. Only the NUMERATION 1 module is commonly available.

The early published history of these titles is interesting. Remember, these rare cartirdges are all c1983, and most only became available in limited quantities in 1984. A full page Scott Foresman ad on page 53 of the November 1982 99ER MAGAZINE (which was published at least a month before the cover date) illustrates instruction books for STAR MAZE, SPACE JOURNEY, and PICTURE PARTS. The same ad shows two TI titles I never heard of, A DOG ON A LOG and A GHOST IN THE HOUSE. Another full page Scott Foresman ad in the December 1982 issue of 99ER shows the FRACTIONS 1 package. ALL of the rare and not so rare Scott Foresman titles (all 12 READING cartridges, the 5 MATH ACTION GAMES, and all of the MATH COURSEWORK series cartridges) are listed in the Sept 1983 99ER MAGAZINE as being available from Soft-Tex of 3 Walnut Lane, Berwyn PA at prices between $40 and $59 each.

Some of these command module titles were sold directly to the public by TI and by Scott Foresman after TI supposidly left the home computer market. A dealer ad in the September '84 issue Micropendium lists some of the Scott Foresman titles for $25 Some titles appear in TRITON catalogs listed as "NEW" starting about 1987.

In the math game modules the emphasis is on the game action or logic rather than the ability to solve specific math problems. Examples include COMPUTER MATH GAMES I, III, and IV, as well as Scott Foresman games such as PICTURE PARTS and PYRAMID BUILDER. The player is asked to solve math problems, but sometimes if the player's answer to the problem is incorrect the computer displays the correct answer and the game proceeds more or less as if the problem had been correctly answered. Many math game modules are timed. In my case this sometimes means that even though I can solve the problems, if I don't solve enough of them fast enough because of the complexity of the game action, then I lose.

Math coursework drill cartridges simulate classroom learning, and in fact many are probably best used in a classroom rather than home. The software tries to teach specific math concepts by solving sample problems for the student and then giving the student a series of problems to solve for himself. Examples are the Milliken Math Sequence series and the Scott Foresman Math Coursework series. Color graphics, music, and voice are often used to make a potentially dull drill more interesting and to reward correct student responses.

Language arts drills include rare titles in the Scott Foresman READING series. Grammar, sentence structure, proper interpretation of text, and research skills such as dictionary and encyclopedia skills are among the topics taught.

The Scott Foresman Reading Series

These modules resemble PLATO software in that they present specific language arts concepts in a text formt and then ask a series of questions to text the student's knowledge of the concept. Unlike PLATO software, the Scott Foresman modules make good use of music and color graphics. The rare modules DO NOT make use of speech synthesis, unlike some of the more common cartridges in the Scott Foresman READING series. These cartridges seem to be designed for in classroom use, which may be why they were not made commonly available to the public. The "suitable age" designations below are taken from the Fall 1987 TRITON catalog which lists most of these modules.

Reading Trail

Suitable for ages 8-12, this cartridge teaches about the characters, setting, and points of view in stories. Famous characters from the Wizard of Oz and a separate story about fishing are used to illustrate specific points.

Reading Power

Suitable for ages 8-12, this module teaches research skills involving the dictionary, encyclopedia, and library card catalog. Specifically the student learns how to find information that is organized alphabetically in these kinds of reference materials. A detective story called "The Lion's Charm" with color anamation and music is used in some of these activities.

Reading Rainbows

This is one of the rarer of the "rare" READING modules. It has been listed in very few catalogs over the years. It teaches how things are alike, parts and wholes, and sizes. Speech synthesis is used effectively. My first grade daughter whipped through this in a very short time, so I assume it is designed for first grade (age 6).

Reading Wonders

Another of the more rare modules, READING WONDERS teaches the student to distingush between various types of fiction and non fiction. Several colorful stories are used to illustrate what is and is not historical fiction, modern realistic fiction, science fiction, biography, autobiography, and information articles. I suspect that this is probably for ages 11-13

Reading Adventures

This uses a variety of stories to teach, within a paragraph, main and supporting details, drawing conclusions, and sequential relationships. I have seen this one mentioned, but not described in catalogs. It looks like about ages 8-10, but I am not sure.

Reading Cheers

I would have guessed this was for 2nd grade, but my 1987 TRITON catalog says ages 8-12 (2nd grade is age 7-8). The module teaches root words with endings (lazy and lazily), contractions, and compound words.

All of the above "rare" modules are c1983. To complete the record I will briefly describe below the more commonly available 1982 Scott Foresman titles in the READING series.

Reading On

Some nicely illustrated science fiction stories illustrate the use of maps, schedules, graphs, and why and how people use them. For ages 8-9

Reading Fun

There is minimal use of speech synthesis in this 2nd grade level module. Four colorful stories illustrate problems and how people solve them, why things happen, and how characters feel.

Reading Round Up

Four stories based on an "American Wild West" theme are used. Comcepts taught are figures of speech, word meaning, and idioms. The module is designed for ages 9-10.

Reading Flight

For ages 11-12. A neat story about an archaeological dig on a south seas island called Bolo Island teaches classifying, summarizing, and outlining information.

With the availability of this "new" 1983 math education software I think it is time for the many adult TI users to "return to our beginnings" and make this "new" software available to our children and/or grandchildren. Dust off a console and set it up for the kids.

Let us return now to those glory days in the 4th quarter of 1983 when the TV was full of commercials promoting the 99/4A as a tool in elementary grade education and when lots of new education software was about to become available. *That software is now here!!*

Monitor Exploits

Diary starts Thursday January 4th.AM. Went and bought a computer sales magazine out every Thursday, to see if they had published my Wants advert for a VGA mono monitor. Yes it had gone in, sit and wait for response.

Evening of the 4th.

Chap from up north of Manchester rings to say, he has what I want Hi-res mono-monitor VGA. He tells me of condition and strikes a deal on postage etc, so we exchange addresses and I promise to send money plus postage the next day. Which I did.

Friday 5th January AM.

10.AM in the morning I went to the P.Office got Postal Orders and Recorded Delivery envelope and sent money.

Wednesday 10th January AM.

Thought I should have received it by now, sent by special delivery I thought. But no chance....

Thursday 11th January PM.

Still no parcel, so I phone the chap in the evening to ask where my parcel is. He said, he posted it by carrier on Wednesday 10th at 10-30 am and it was on a 48 hour special delivery. So I say ok...and he gave me the docket number and consignment number, so I could check up with the carrier company if their were any problems...

Friday 12th January AM.

It is now morning and still no parcel, so I phone the carrier's again and they reply. Have you got the consignment number, yes I have and gave it to her. They check on the computer and answer yes, it arrived in Bristol yesterday morning 11th January and go and check warehouse, they come back and say yes it's here and we will get it to you as soon as possible..

Saturday 13th January

Wait in nearly all day, then phone carrier, they answer and I ask where is my parcel. He replies sorry yard is closed at noon, your parcel must be on the van which is out delivering...

Monday 15th January AM

I telephone carrier company and ask where is my parcel, they answer please can I have your consignment number and checks the computer, then answers yes it left the north depot at 10-30 am Wednesday 10th and arrive in Bristol on Thursday 11th ... Yes I say but where is it now, answer it must be on the van...

That afternoon.

I phone the carrier again, please can you tell me where my parcel is, they answer, can I have your consignment number (me) " SCREAM" look here I said angrily, the parcel was sent on Wednesday 10th January at 10-30am by 48 hour delivery, it is now five days later. Please can you send my parcel, answer...Please wait and I will find out for you (I hear in the background... no idea, what shall we do) they come back to the phone and say can you leave it with them and they will try to find out for me, and call back..

5.O,Clock in the evening.

I telephone the carrier again, and aske where is my parcel,they answer, can we have your consignment number........(my answer) * * ! ! $ * $ or words to that effect. So I give it, please wait and they come back in a couple of minutes later, sorry we seem to have mislaid it. What are you going to do about it, I ask, answer, we will do an investigation and send you a claim form. Yes but, I said, you recieved it in your depot in Bristol on Thursday 11th, your warehouseman saw it there and told me it will be out on the next delivery, so it must be there. Sorry was the answer, it's gone !!

6.O,Clock same evening.

It has arrived mysteriously, the deliveryman says nothing, no excuse, nothing. On questioning him further, he say's, I know nothing about it, I just deliver parcels (not at that time of the day he doesn't)

6.20 PM same evening.

Monitor Exploits

Unpack parcel, check for damage, Test on computer. "OH NO" I've been sold a LEMON. Re-pack, take to my mates house and check it on his computer, same outcome.

TEST

No picture (fuzzy)
Reset with controls
Picture (still fuzzy)
Contrast (set to full)
Tube faulty (picture only fills 3/4 of screen)
No adjustment availiable.

Telephone chap up north and explain problem, he answers, works perfectly on his computer, my answer (as test above) what is he going to do about it. His answer, it must have been damaged in the post. Please return and he'll test it, and if damaged in the post he will make a claim. If not damaged, he will refund my money and re-sale it, I answered thankyou, also for him to take five pounds for his troubles and I will send it back by a different carrier. (16th January)

Epilogue

I was lucky there with this episode, but my moral to this story is never buy second hand by post, unless you are insured or you make a written policy agreed by both parties. Also I will be making a claim against the carrier's for breach of contract on my parcel, by taking 128 hours for delivery instead of 48 hours as agreed and paid for.

Computing a Health Hazard ?

Users of VDU's have been complaining for years about sore eyes and headaches, blurred vision whilst using their computers. The glare effect of bad lighting can result in mistakes and can make you feel uncomfortable whilst using your computer. Sunlight and overhead lighting in the office or home etc can make a strain to your eyes which reflects off your screens. Also static electricity builds up on the surface of your monitor screens in dust particles and is propelled back to the user to be discharged. Monitors and VDU's discharge minute levels of VLF and ELF Frequences (low radiation) using a Filter screen reduces this problem of glare, static and exposure of the low radiation whilst improving contrast etc to your VDU's.

There are many types of Filters and Screens on the market to fit all sizes of VDU's and monitors from 13" to 21" and Laptops. Here's a brief discription of some of the Filters etc.

Polaroid Filter

This Filter is Optical Triacetate, lightweight and unbreakable, eleminates 99% of reflection. It has a double layer of circular polariser's and also eleminates 99% of VLF and ELF radiation. It also reduces Ultra Violet light as well.

Glass Filter

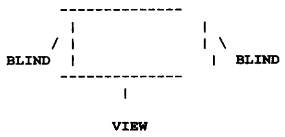
This Filter has multi-layer anti-reflective coating and absorbs 95% of glare, it also has tinted glass that takes the rest of the glare. Resulting in much better resolution (no fuzzy halo's) contrast is much improved, between image and background.

Glare Filter

This Filter is made of High Quality Mesh chemically treated to be non-reflective, it also contains conductive fibres to absorb static (VLF + ELF radiation). The threads are louvred at angles so light going through can't be reflected back as glare. The light from the VDU travels straight through the Filter for glare free viewing.

Privacy Filter

Special Micro-louvre Technology, which prevents bystanders viewing on screen data from either side of your monitor or screen. To view your screen or monitor, you must be sat directly in front to see the data on screen. Also it blocks out 99% of VLF + ELF radiation, also static charge build-up. Fits all notebooks and monitors...



Dear TI'ers,

I hope you have all had a good Christmas and new year. At the time of writing, I'm on the first day of my extra two days off of work after new year. Despite what Edwina Curry said about northern eating habits, and despite my favourite food being chips, I've just made myself some pasta. In case you're interested, a Cross & Blackwell "Romana" Pasta Premiere choice with mushrooms and red & green peppers. I just stick it in my army mess tin, add half a pint of water, and an estimated four or five teaspoons of butter or margarine. Then I add my extra ingredients which could be a dash of barbeque sauce and loads of extra pepper. Keep stirring it while you boil it, and then turn off your electricity or gas and keep stirring until the mixture thickens. When you're happy with it, that's it! I just eat it out of the mess tin to save washing up, but you could be civilised and put it on a plate! It says on the packet it should serve four, but somehow I don't think so, unless that's just me being greedy!

I thought I would make this while I contemplated the work ahead! I've made my new years resolution to try an get up to date on all my TI jobs. The main one has got to be trying to get the SCSI card in the box, and more importantly to get it working. My immediate work is to make all my necessary hardware modifications the SCSI card will need. The most important thing is the 32K that the SCSI version of MDOS needs to load the extra SCSI DSR stuff. Well, I thought that it's loading the SCSI DSR. I think it was Bud that said it's being used as a cache (data buffer). The thing I don't understand is that he said it's because the GENEVE can't handle the speed of the data!

It obviously can if the buffer is part of the GENEVE's memory, but the 32K chips are ZERO WAIT STATE which means literally that they are fast enough so that the processor doesn't have to wait to give the chips time to catch up! However, the GENEVE comes with 32K of zero wait state RAM which I thought would be good enough, but I've just thought that possibly the SCSI DSR might load into the original 32K, and it will use the second 32K as the cache.

The other job I've got to do is modify the address decoding on the 248K Horizon RAM Disk so that it doesn't clash with the GENEVE's extra memory. The address lines in question are AMA, AMB, and AMC which are the extra lines that allow you access extra memory in the box. I think the Asgard Memory card also uses these lines.

TI included them for future expansion, and they made the sensible move of mapping them out on all their cards so that nothing would clash with them. The early Horizon RAM Disks didn't map them out, and neither do any Corcomp cards.

Well, here I am, today's date is the 18th of February, and I can see what people mean when they say it's difficult finding the time to do things. I've been working late and bringing work home! Our company produces PC phones, which means you have a card in your PC instead of a phone on your desk, and you plug a wire into the switchboard, and a handset into it.

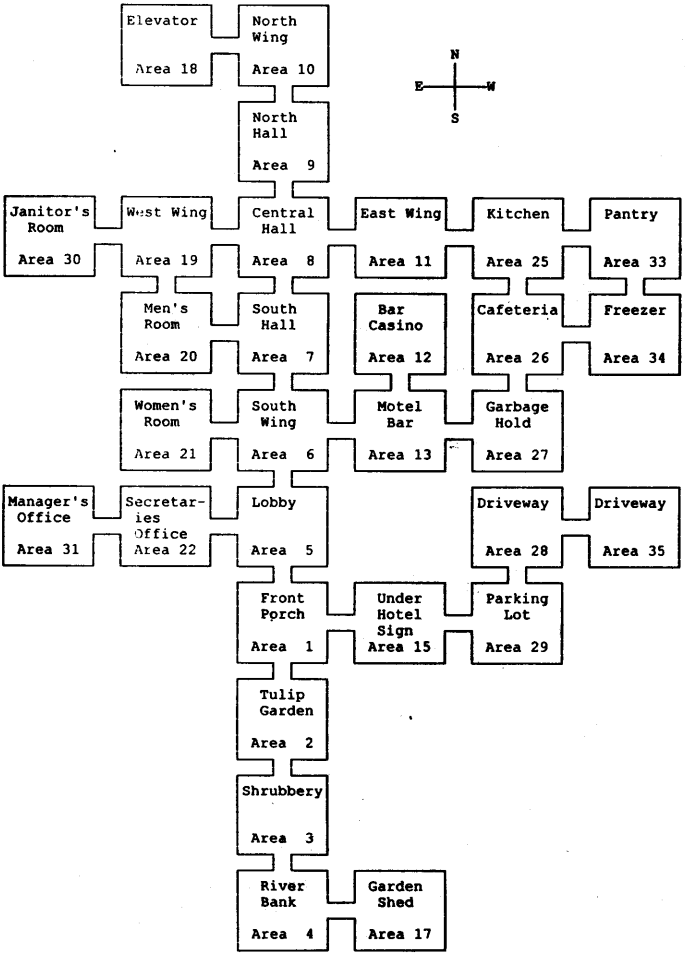
It seems strange that big switchboard manufacturers such as Toshiba, WIN, Goldstar, TIE, and Philips should trust the development of the PC version of their phone to a small company in Edwalton in Nottingham, but that's the case, and the Toshiba Windows Screenphone even made it into a networking magazine.

I'm writing the PC phone for Philips at the moment using visual C for windows, and it's good experience when I finally get around to writing my CAD program. I will write it. When I've organized my disk system onto my optical drive.

I've not made much progress with the SCSI card. I copied my SCSI operating system disk which was kindly sent to me by Jeff Kuhlman in Germany. I hope to meet Jeff at the workshop in Sandbach. My drive wouldn't read the disk, so I had to visit Trevor to copy it. As yet, I haven't had time to try loading this operating system, but my SCSI card is still plugged in my box, and has not caused any complaints with other cards while I've been booting up with old 1.14 MDOS. I really need to get myself in gear and get some more up-to-date software from the States. The BBS is beginning to acquire loads of new files, particularly from Kenneth Hughes who has sent loads of new text files and pictures for the Star Trek area. Quite a bit of time has been taken up by visiting my dad in hospital after he was rushed in on the 15th of February, but he's up and about and wandering about. I will apologize to Pat Trainor in Northern Ireland now if I delay sending him a 4A console, but time is short for the next couple of weeks, and I have struggled to get anything ready for the magazine. I hope users of the BBS will find my flowcharts useful which Richard should have included somewhere. Also to fill up some space, here's a map of the Murder Motel game on the BBS.

All for now from Richard T.

See you in Sandbach on the 16th of March.

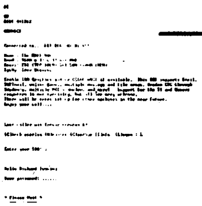


Texas Instruments TI-99/4A User Group U.K.

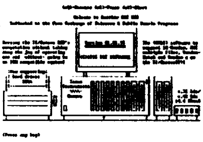
Mansfield On-Line Bulletin Board (The MOBB)

Flow Chart Part 1.

Step 1.
Loggin in....

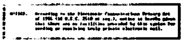


Step 2.
Main Title Screen

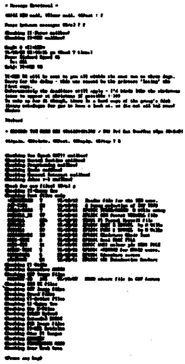


Step 3. Your Details.

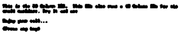
Step 4. Data Protection Act Screen



Step 5. Message Retrieval and checking for new files.

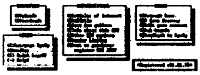


Step 6. Welcome Message
Indicates your column width.

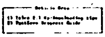


Main Menu

Then.....
Step 7.

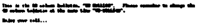


A



Article Area
Contains two areas
of text information
for TELCO and FastTerm

B



Bulletin Screen
General Bulletin.
Still needs editing!!!

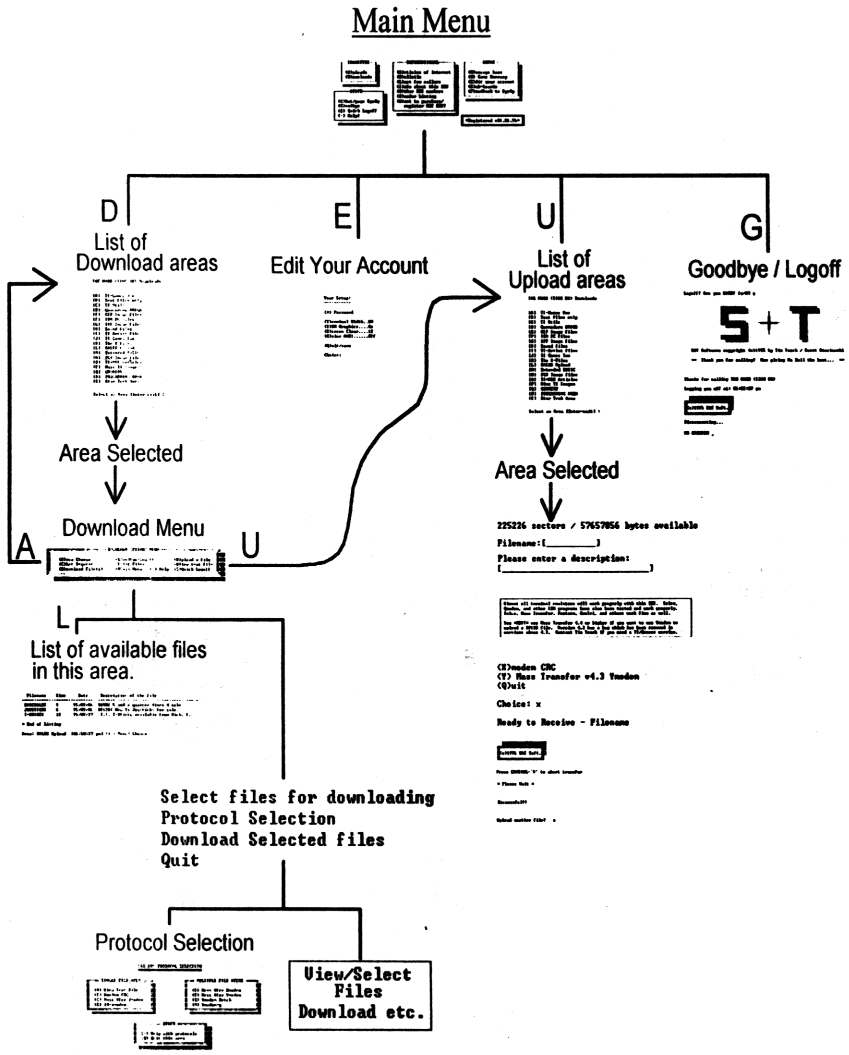
C

Chat with Sysop
If Trevor hears the alarm
he'll break in for a chat!

Texas Instruments TI-99/4A User Group U.K.

Mansfield On-Line Bulletin Board (The MOBB)

Flow Chart Part 2.



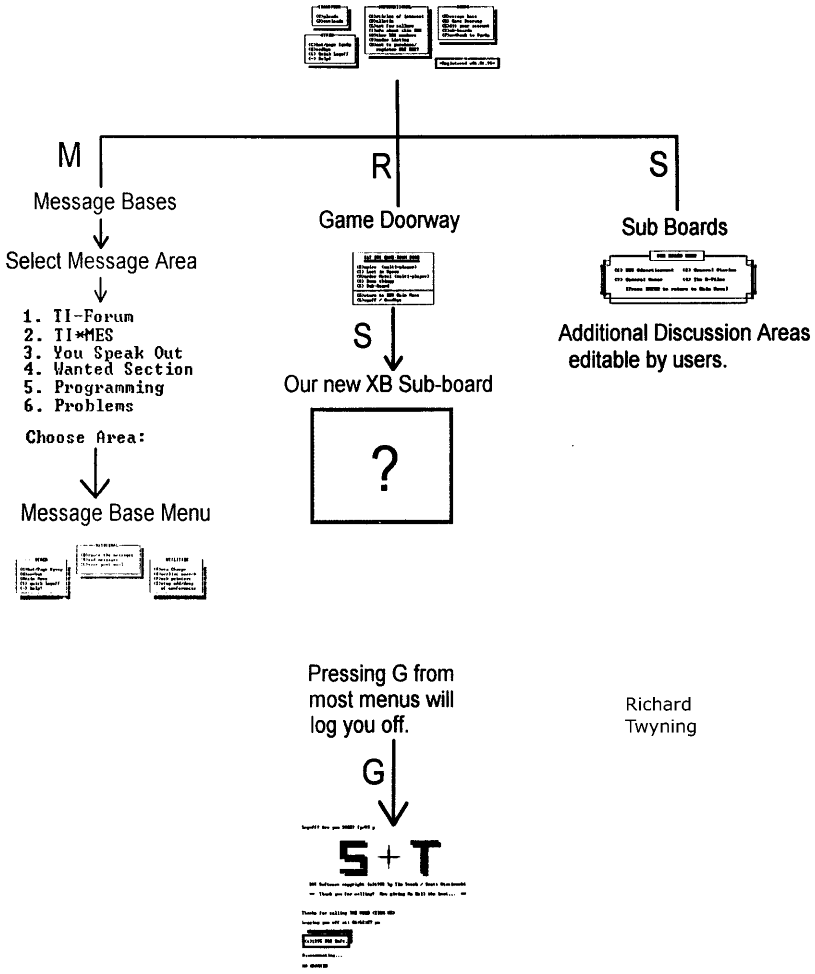
```
225226 sectors / 57657856 bytes available
Filename:[__________]
Please enter a description:
[______________________________]

(X)modem CRC
(Y) Mass Transfer v4.3 Ymodem
(Q)uit

Choice: x

Ready to Receive - Filename
```

Texas Instruments TI-99/4A User Group U.K.

Mansfield On-Line Bulletin Board (The MOBB)

Flow Chart Part 3.

Main Menu



M

Message Bases

Select Message Area

1. TI-Forum
2. TI*MES
3. You Speak Out
4. Wanted Section
5. Programming
6. Problems

Choose Area:

Message Base Menu

R

Game Doorway

S

Our new XB Sub-board

?

S

Sub Boards

Additional Discussion Areas editable by users.

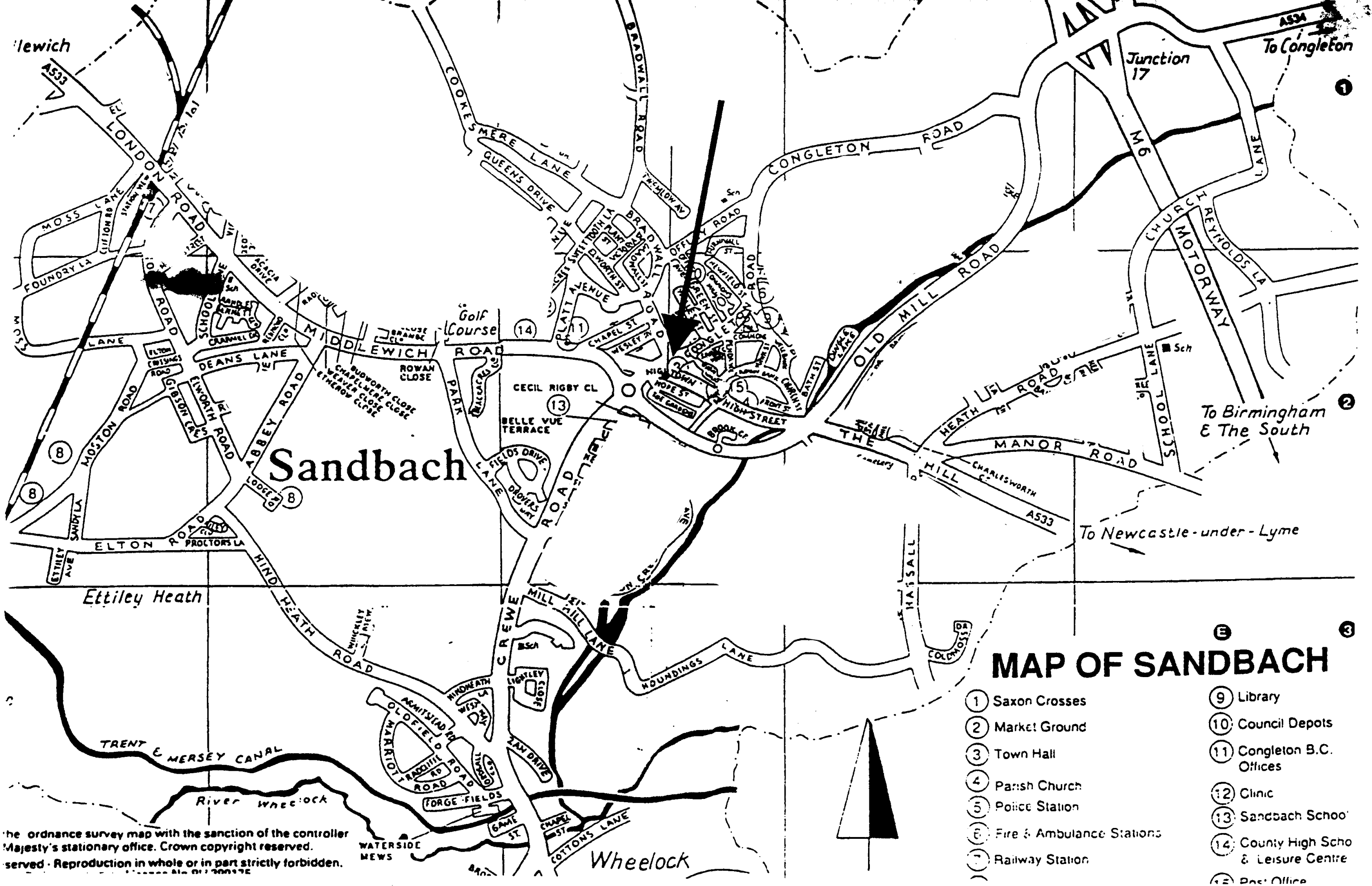
Pressing G from most menus will log you off.

G

S+T

Richard Twyning

The Back Page



MAP OF SANDBACH

1 Saxon Crosses

2 Market Ground

3 Town Hall

4 Parish Church

5 Police Station

6 Fire & Ambulance Stations

7 Railway Station

9 Library

10 Council Depots

11 Congleton B.C. Offices

12 Clinic

13 Sandbach Schoo

14 County High Scho & Leisure Centre

15 Post Office

he ordnance survey map with the sanction of the controller
Majesty's stationary office. Crown copyright reserved.
served · Reproduction in whole or in part strictly forbidden.