TI*MES

TI(/4A USER'S GROUP (U.K.) CONTACTS

Chairman: Trevor Stevens. Tel.0623 793077
249 Southwell Rd. East, Rainworth, Notts. NG12 OBN
Vice Chairman & Programming: Mark Wills.
12 "Rosehill" , Betton St., Shrewsbury, Shropshire. SY3 7YN
General Secretary: Richard Twyning. Tel. 0623 27670
24, Peel Rd., Mansfield, Notts. NG19 6HB
Publicity Officer & Module Librarian: Philip Trotter. Tel.0642 817356
80, Martonburn Rd., Grovehill, Middlesborough. TS4 2TH
Membership Secretary: Alasdair Bryce. Tel.0389 65903
51, Dumbuie Ave., Silverton, Dumbarton, Scotland. G82 2JH
Treasurer: Alan Rutherford. Tel.0625 524642
13, The Circuit, Wilmslow, Cheshire. SK9 6DA
TI*MES Editor & Distribution: Alan Bailey. Tel.081 508 1053
14, Shelley Grove, Loughton, Essex. IG10 1BY
Hardware & Projects: Mike Goddard. Tel.0978 843547
"Sarnia", Cemetary Rd., Rhos, Wrexham, Clwd. LL14 2BY
Librarians:Cassette: Nicky Goddard. Tel.etc. as above.
Disk: Stephen Shaw. (journal Exchange)
10, Alstone Rd., Stockport, Cheshire. SK4 5AH
Publicatios : Mike Curtis. Tel. 0209 219051
21, Treliske Rd., Roseland Gdns., Redruth, Cornwall Tr15 1QE

MAGAZINE CONTENTS

ISSUE NO. 37 SUMMER 1992

EDITORIAL

I am glad to be able to report that we have a full 60 pages again, and good augeries for the future! As you will see from the front cover we also have a full crew for you to contact on every aspect of use of our computer. In addition there is , as you will see from the contents of the magazine, every indication of lively hardware developments overseas which we hope will become available to us here.

We welcome Richard Twyning and are grateful to him for being willing to undertake the considerable task of General Secretary. We must all acknowledge with our thanks the years of service by Jim Ballinger in this post. Thank you Jim, and best wishes for a speedy recovery from ill-health.

Our thanks also to Edward Shaw for his years of service as Module Librarian, and to Phil Trotter for his readiness in assuming this load as well as that of Publicity Officer.

It was decided at the AGM to provide me as Editor with a PEB on loan from the Group, so as soon as I can get my old Diablo daisywheel connected, I should be able to accept disks as a source of articles for the magazine. I will let you know how I get on! By the way can anyone help with the pinout of the Miniwriter III + module? Pity if no one actually reads this part of the magazine!

DISCLAIMER

The views expressed by contributors to this magazine are strictly their own and do not necessarily represent those of the Committee. Contrary opinions are welcomed, and will be given equal prominence if at all possible. Attributions not made, or made in error will be corrected on request.

NEXT COPY DATE

Please let me have your copy for the next, Autumn, issue of the magazine by 1st. September. This present issue is made up of very well reproduced copy. Very few sheets required lateral adjustment, most sitting very perfect, and only some sheets needed trimming in the heads to keep their bottoms on the page. Thank you one and all!

FROM THE CHAIRMANS CHAIR

T.Stevens c1992

Well the AGM at DERBY is now over and many things seen and said. We had a very good turnout at the show and a lot was on offer. The minutes for this meeting will appear in the next issue so I will not go into those in this artical.

I showed as promised the two programs that I reviewed in the last issue of TI*MES. These were Sound FX and GIF Mania. The demo's were a great success which showed how the pictures and sounds could be passed from machine to machine. I had my AMIGA in the demonstration. We also had Richard TWYNING with his Geneve. I showed, over the course of the day, how to take a digitised picture, in this case a picture of the TI99/4a, which was in HAM Iff formatt on the AMIGA. Then on that machine converted it to a GIF file. I then with TELCO and JR-Comm joined the two computers together. I then passed the data to the Texas. I then ran GIF Mania. Richard ran YAPP. We then had three pictures of the same subject on all three machines. Everyone was amazed at the quality and colours that the TI produced seeing that it was only in 40 cols and restricted to 16 colours, which cannot be palletted. Also the fact that due to the processor type the blocking was minimal. The demonstration was also photographed by one of our overseas visitors.

We had as you have already noted overseas visitors. They came from Germany and Holland. Both fortunately spoke very good English, so a lot was learned from a new area of the TI community. This was highlighted when I went round the show and found that the Dutchman, (I can't remember his name!) was selling plans on behalf of the Dutch TI User Group, for an 80 Coloumn card. To be quite honest I could not believe the price. For the sum of £1.00 I got a book (In Dutch) and the plans, plus the schematics for the card. He also had with him a made up card which was of wonderfull quality. Into the price was a Disk which included the full DSR Programs for putting onto the EPROM. For this price I could not miss the chance, so I bought one of the packages. Now I am no electrical wizard, I know a small amount about how the thing works. However the only project I have done before is to make a Wiget, which I am using now. Fortunately at the show I met a man called Derek HAYWOOD. We looked at the thing together. We now have in the pipe line two production cards which can be made by any PCB firm and also a contact for the V9938 which is the hart of the thing. Derek has to date managed to sort out a some queries and adjustments, and has arranged for the booklet to be translated into English. He has infact sent me the first half of this. So with a few visits and exchanges of ideas, the card could quite easily be available through the TI User Group (UK) in Kit form for anyone wishing to build one. The cost I think at this time will be about £55.00 all in, but that remains open to change at the present due to demand and item prices. Ie V9938 £27.00.(£13.00 in 50's) However if you know where they are

cheaper let me know.

I also had a rumage around Mike Goddards stall and found some nice cheap disks. He also had on sale some old ROMOX cartridges which were ideal for Super Space convertion. Phil Trotter was also there with his large supply of drinks etc, to keep us all cooled down as the show was on a nice sunny day.

While I was there I discussed various things with other members and it became quite clear that the TI*MES was becoming DISK based and did not cater for the unexpanded member. I would like to try and correct that and would like to see some of you out there writing for your magazine. I do intend doing some articals myself but time is time as you will be aware. So if you have a program be it on tape or disk, or you have written something on a typewriter then send it into our TI*MES Editor. I would also like to encourage a questions and answers coloumn. Do you have any queries? I am sure you have. If you have, put pen to paper and send them to me and I will try to answer them and then send them in for the next issue. Your name of course will be used if you give permission.

To show you what I mean regarding BASIC programing I am going to pass on a few small programs for you to type in. These are one's that go back a few years but even so you can learn by them.

SOUNDS

The TI has a very good Three Channel Sound processor with two extra Noise Channels. These can be put to good use if you use them wisely. I will not explain about CALL SOUND as I feel that it is quite well explained in the TI Manual. However just by using the volume variable in a loop can do some wonderfull things. How about this....

```
100 REM A is the Volume
110 FOR A=0 TO 30 STEP 5
120 CALL SOUND (-99,698,A,1924,A)
130 NEXT A
140 FOR A=0 TO 30 STEP 5
150 CALL SOUND(-99,554,A,1527,A)
160 NEXT A
```

Did that sound like something you knew? Well try this one for size.

```
100 REM Watch your mirror!!
120 N=1
130 FOR F=700 TO 900 STEP 5
140 CALL SOUND(-99,F,0)
150 NEXT F
160 FOR F=900 TO 700 STEP -8
170 CALL SOUND(-99,F,0)
180 NEXT F
190 N=N+1
200 IF N=4 THEN 270 ELSE 190
270 END
```


As you can see the second program uses the notes instead of the volume.

** MINI COMPETITION **

Well I hope you have been inspired into writing for us and we hope to hear from you soon. Just to start the ball rolling I have below a listing in EXB which is called a one liner. This is an idea that was first started by Home Computer Magazine, way back when the TI99/4a started in 1979. What you have to do is write a game a demo or whatever, but it must not exceed one line. You can how ever use the extended line trick which gives you more on one line. You do this when the computer beeps the end of line, by pressing enter. Then re-enter your line number as this the case below (1) then use FCTN E then enter, then FCTN 8 and then you will have the extra room. You will have to do this on the example below which was sent into HCW and won a prize by Mr Hamilton of Canada. Also when you enter the program, to condense don't add any spaces between the seperators. When you press enter the computer will sort this out for you. If you come up with anything then send it to me (My address is one the front.) and I will GIVE AWAY some XB Programs to the winner. This will be on tape only. You will then get yourself into the magazine.

This is a game called Alphabet Attack. You the "@" symbol are being attacked by the Alpha letters. Points are awarded against you for hits. Low score and high game length wins.

```
1 N=28::FOR X=4 TO N::CALL SPRITE(£X,60 +X,X/2,N.N,X,M)::FOR
Y=5 TO X::CALL COINC(£Y,£4,N+M,C)::M=M-::DISPLAY
AT(4,9):M::CALL JOYST(1,E,F)::CALL MOTION(£4,-2*F,2*E)::NEXT
Y::NEXT X
```

Going back to the AGM I thank all the members that supported us on the day, and to those who took on the task of being a committee member once again or for the first time. Without your support this group would not survive. I also thank all those that could not attend, but support us by your yearly membership.

TIUG(UK) MEMBERSHIP NEWS

by Alasdair Bryce

This has been a busy few months on the membership front with renewal forms coming in thick and fast. Since issue 36 we have been joined by four new members. A warm welcome goes to David McCann, Sumaela Yassin, Robert Wilson and Shabtai Affias all the way from Nova Scotia. It is also my sad duty to report the death of Mr. Stan Preston of St. Helens who had been a member of the group for some years. He will be missed by all in those in the TI world who knew him.

Several of you raised the question of TI*MES back-issues in your renewal forms. For those who may be interested I can exclusively reveal that the following issues are presently in print and are available for purchase, viz: 4,8,9,10,11,13,14,16,18,19,20,22,24,25 27,29,34,35 and 36. All issues are priced at £1 each plus p&p and I would be happy to hear from any of you who are looking to fill in gaps in your TI*MES collection.

A number of you also mentioned the possibilty of printing area lists of members in TI*MES as Peter Walker had done a few years back. This has been duly noted and a full list of all members will appear in issue 38 in the hope that some of you can get together and form your own local group's or organise self-help workshops.

Many thanks to those of you who turned out for the A.G.M. in Derby. Despite the 8 hour bus journey home I thoroughly enjoyed the day and came away with the feeling that the TI was going to be around for many years to come. While the machine's U.K. following may not be huge, what it lacks in numbers it makes up for in enthusiasm and determination.

Those of you who will be receiving renewal forms with this issue will notice that the form has changed slightly. It was decided at the A.G.M. not to increase membership fees for this year but in order to provide funds for the groups libraries to obtain new software for the benefit of all members it was suggested that we would instead ask members to make a donation to group funds of 50p or so in addition to their £12.50 renewal fee. This is entirely voluntary and a matter for each individual to consider for him or herself but everyone stands to benefit from it and even at £13.00 the group still offers excellent value for money so please don't be shy.

That's all for now. I'll see you again in October once you've recovered from your summer holidays.

STILL MESSING ABOUT.

SHOPPING LIST by Jim Ballenger

This programme is based on one included in Jim Peterson's Tips from the Tigercub that Jim circulated to club newsletters, and I used it slightly messed about because I'm queer like that. If you read this, you will know that Jim has O.K'd the idea.

It is a bit cheeky to alter programmes by such a respected and energetic a programmer, but he has a sense of humour too it seems.

The programme should be run with the printer fired up, and will print it headings automatically. It is so designed that if you alter the data lines 400 et hoc as that they read in the sequence the items are met the appropriate line in your supermarket, the printed list will follow this.

The goods listed can be altered simply by amending the data list suitably. This list is, or rather was, based on my local store, but a new management has altered the lay out(!)

A small listing, but it is, as usual, available through the club library if required.

```
100 CALL CLEAR :: CALL SCREE
N(12)
110 FOR C=1 TO 9 :: CALL COL
OR(C,7,16):: NEXT C
120 OPEN £2:"PIO"
130 PRINT £2:CHR$(27);"W";CH
R$(1);"SHOPPING LIST" :: PRI
NT £2:"*************": : :
140 DISPLAY AT(6,1):"    S
HOPPING LIST"
150 DISPLAY AT(7,1):"    *
************"
160 DISPLAY AT(16,1):"  GROU
PS OF GOODS WILL BE    SHOW
N. ENTER A QUANTITY    FOR
EACH ITEM REQUIRED."
170 DISPLAY AT(20,1):"  IF N
OT NEEDED PRESS THE
   ENTER KEY           PRES
S ANY KEY TO START"
180 CALL KEY(0,K,S):: IF S=0
 THEN 180
190 RESTORE :: FOR F=1 TO 57
 :: READ A$
200 DISPLAY AT(12,1):A$
210 DISPLAY AT(12,LEN(A$)+2)
:"0"
220 ACCEPT AT(12,LEN(A$)+2)S
IZE(-4):Q
230 IF Q=0 THEN GOTO 250
240 PRINT £2:"[ ] ";A$;RPT$(
" ",(17-LEN(A$)));Q
250 NEXT F
260 FOR DEL=1 TO 250 :: NEXT
 DEL
270 DISPLAY AT(12,1):"ADDITI
ONAL ITEMS? Y/N"
280 ACCEPT AT(14,13)VALIDATE
("YN")SIZE(-1):Q$
290 IF Q$="N" THEN 370
300 DISPLAY AT(12,1):"ITEM?"
310 ACCEPT AT(12,7):A$
320 DISPLAY AT(14,1):"QUANTI
TY?"
330 ACCEPT AT(14,11):Q
340 PRINT £2:"[ ] ";A$;RPT$(
" ",(17-LEN(A$)));Q
350 DISPLAY AT(13,1):"
360 GOTO 270
370 PRINT £2:CHR$(27);"W";CH
R$(0)
380 CLOSE £2
390 CALL CLEAR :: END
400 DATA BUTTER,MARGARINE,CH
EESE,EGGS,CREAM,COOKING FATS
,COOKING OILS,TEA,COFFEE,COC
OA
410 DATA SUGAR,MILK,TINNED F
OODS,FROZEN FOODS,SOUPS,BREA
KFAST CEREALS,BISCUITS,JAM-M
ARMALADE,FLOUR,CAKE MIXES
420 DATA CUSTARD,JELLIES ETC
.,RICE/SAGO ETC.,DRIED FRUIT
,SPAGHETTI,PEAS/BEANS,PICKLE
S/SAUCES,SALAD CREAM,SALT/PE
PPER,VINEGAR
430 DATA SPICES/MUSTARD,FLAV
OURINGS,MEAT,POULTRY,SAUSAGE
S,BACON,HAM/PIES ETC.,SUET,F
ISH,FRUIT
440 DATA VEGETABLES,SALADS,B
READ,CAKES,PET FOODS,ICE CRE
AM,SWEETS,WINES,SOFT DRINKS
ETC.
450 DATA SOAPS,DETERGENTS,PO
LISHES,CLEANERS,SODA/STARCH,
MATCHES,TOILETRIES,STATIONER
Y
```

MODULE LIBRARY-MODULE LIBRARY-MODULE LIBRARY-MODULE LIBRARY

Do you have any modules that you would consider selling or donating to the module library. Resonable prices paid. for more information contact me at the address below.

The latest list of modules available for purchase follows
*please note that cheques should be made payable to P.TROTTER
Also members should contact me about the modules that they are seeking as stocks are constantly changing.

ADDITION AND SUBTRACTION 1	3.00	YAHTZEE	3.00
ADVENTURE WITH PIRATE TAPE	5.00	NUMBER MAGIC	3.50
BLACKJACK AND POKER	3.00	PROTECTOR	4.50
HUNT THE WUMPUS	3.50	A-MAZING	3.50
BEGINNING GRAMMAR	3.00	HANGMAN	3.50
CONECT FOUR	3.50	TI INVADERS	3.50
DISK MANAGER*	2.00	TOMBSTONE CITY	3.00
EXTENDED BASIC INC MANUAL	22.50	OTHELLO	3.00
EXTENDED BASIC NO MANUAL	15.00	HUSTLE	3.00
PERSONAL RECORD KEEPING	3.50	ALIEN ADDITION	3.00
PERSONAL REPORT GENERATOR	3.50	THE ATTACK	3.00
TERMINAL EMULATOR II	5.00	EARLY READING	3.00
EDITOR ASSEMBLER MANUAL*	22.50	TI WRITER NO MAN*	3.00

MODULES MARKED * NEED DISKS,32K RAM OR BOTH ALSO NOTE EARLY READING NEEDS A SPEECH SYNTHESIZER TO RUN.

PURCHASING MODULES FROM THE LIBRARY

You may return any module purchased within four weeks and be refunded the purchase price less postage which will be charged at 40 pence per module.

Application to purchase/loan modules. modules required

Name _____ _____

Address _____ _____

_____ _____

_____ _____

_____ _____

_____ _____

I enclose cheque/postal order for £ (as indicated on the list) PLEASE MAKE CHEQUES PAYABLE TO P.TROTTER.
Foreign orders can only be accepted if a bankers draft is enclosed drawn in sterling on a london bank. it also helps if a little extra is added on for postage overseas.

80 MARTONBURN RD GROVEHILL MIDDLESBROUGH CLEVELAND TS4 2TH

Dear TI'ers,

this is an unusually short article for me, but I felt that I couldn't let everyone down by not putting in some sort of article.

The reason it is so short is because I'm in the middle of revising for my exams, and I haven't got time to carry on with one of the important things in life, which is writing TI software.

For those who are interested, I am studying BSc. Computing Systems at Nottingham Polytechnic.
My subjects include:

Computer Technology (A sad excuse for sad 8-bit Rotimola 6809 programming),

Systems Programming (Compiler writing),

Analytical Methods (Set theory, matrices etc.),

Statistics (An O.U. style lecturer with no life),

Software Engineering (Data abstraction, Data Flow Diagrams, Filing Systems, and the complete guide to life, the universe, and the number 42 with Dr. Frank Newman!!)

Computer Systems (Rotimola 68000 programming, but the lecturer dislikes Intel and Motorola, and has done 9900 and admits its advantages),

Procedural Programming (Sad Borland Turbo Pascal with no standard power function!!),

and finally, Physics (which sad fool had the idea of putting physics on a computing course!!)

It was probably the same fool that had the idea of putting 8-bit assembly programming on a degree course.

People in the 23rd century will look back in the archives of time, and wonder how stupid the world was to use such machines.

As I am sat in lectures, and even in the computer room, I feel as though I'm superior to everyone on the course.

Even when I see fourth year students walking around, I think to myself, they're only educated in 8-bit programming and have had four years of the IBM brainwashing process, whereas I've had an ultra-reliable 16-bit stackless RISC machine for the last ten years.

It taught me programming, and produced my diagrams for a 5th form 'O' level physics project in 1985 for which I got 20 out of 20, not to mention all my work in the sixth form, and through my year of business studies, and a year of first diploma in engineering.

My 4A did have a 14 month (well earned) break when I got my first Geneve, but in August 1990 when the Geneve overheated in the heatwave, out came the Cray 1 again, and it supported me

again until February this year, when I got my second Geneve.

I recently took my console into the Poly so that I could refer to it in a talk which I presented to the group on decent computing.

I was sat there waiting while people walked into the class, and some sad fool walked by me to sit down, and he thought he would try and attempt a joke.
He pointed to the Cray 1 and said, "I did my Pascal project on one of those!!"
Well, I did do my Pascal project on my 4A, and typically my Multi-Workspaced mini earned me an A for that project.
If only that brainless fool, and the rest of the world knew the power contained in our machines.
The assignment was to produce a lexical analyser. In Turbo Pascal it took probably 300 lines or more!! In poor old 1980 TI Extended BASIC with a Virtual Memory Array (Dis/Fix 254 Relative file to you), it took about 15 lines. I actually used Extended BASIC to write the Turbo Pascal program. I read the Relative file from disk with a little 10 line XB program, and then generated the Turbo Pascal Source code that would replicate the same operations as my mainframe computer version, and then outputted the source code to a Dis/Var 80 file.

Oh yes!! I almost forgot to mention a subject I left off of my list. This is the saddest subject we have had to study. I can't beleive the amount of brainwashing that this lecturer (Ted Ashworth if you have the misfortune to ever go to Nottingham and bump into him!) must have suffered when he was a student. In one of his lecture notes, he gave some dates on the development of computing:

1978-82: 8-bit

1982-86: 16-bit

He also said that the best machine from 1978 to 1982 was the Apple II!! It's a crime that biased IBM propaganda merchants like this should be allowed to carry on teaching.

It's a sad world out there!!! I blame it on Intel!!!

Well, sorry about getting carried away, but I just can't stop when I get going, especially when talking about my 4A (or music!!) Music and computing have alot in common. Today's music is nothing but endless noise aimed at brainwashing sad fools wearing back to front baseball caps. It's got alot of flashy lights, but no underlying power (IBM fits that sentence perfectly), but go back 41 years to 1951 and you've got John Lee Hooker's MAD MAN BLUES, or back to 1936-1937 and you've got the recordings of the great Robert Johnson.

I have been trying to get some blues recordings converted to Sound FX files, and have got a file of Leadbelly's June 15th 1940 recording of Midnight Special.

I had it digitized on a P.C. using a sound blaster, but

the file format was slightly incompatible with sound FX (and Mr. Chairman's Amiga). It's possible to hear the song, but there's loads of noise in the foreground that needs tidying up!!

Well, I did say that this would be a short article didn't I!!!

Now, down to the important stuff. Firstly, I must thank Mark Wills for the nice writeup he gave Gary Smith and I in the last newsletter. However, we have had some problems with the graphics card he mentioned. The YAMAHA V9990 is a surface mounting chip, and would have had to be sent to a proper company and mounted, which would cost quite a bit, unless we were talking quite a few cards, and without knowing the response to the card, it would have been risky. The card would work (hopefully) on an expanded 4A, or GENEVE, in the expansion box, and, console only owners, YES, it would work on a console, with NOTHING but TI-BASIC. In both configurations, however, it would need an RGB monitor, seperate from your normal display.
The reason that it would work on the console was that we had not used any signals in the box, that are not available on the console. It would however, mean paying a little bit extra for a small circuit which swapped a few pins around, and sorted out the power to the card. The card would then be usable, and ALL functions would be available from CALL's from TI-BASIC that we would implement in the spare space of the card's DSR ROM.

What the card would not do, is replace the 80-column card. People wishing to give their systems 80-columns, would still have to get an 80-column card. What it would do however, is provide the additional mindblowing power of 512 * 424 in 32768 colours, and 640 * 480 pixels non-interlaced, and 768 * 480 pixels, interlaced. If that's not enough, there's always the V9990's 125 sprites to play with!!! It may not be too late. Anyone who is interested could let me or Gary know, and if there's enough of a response, then we will have to see what we could do. Another chip under consideration however, is the Texas Instruments TMS34010 which would give an acceptable resolution of 1024 * 1280 with 256 different colours per line, selected from a palette of 16777216. * Also controllable from TI-BASIC *

Another project that Gary has in mind is a Hard Disk Controller card using the same chip as Louigi, which would be 100% hardware, and software compatible with the Myarc HFDCC, but it would lack the floppy function. This isn't a great loss, as I'm still using my TI-Controller for floppy control!!

For those people who attended the AGM at Derby, may know that I had a few problems with the GENEVE. This was due to me putting my 12 to 16MHz Turbo button upside down, when I set the machine up at the show, and for some reason my GPL mode doesn't like 16MHz. This is because this GENEVE as got slightly slower RAM's. Gary's GENEVE is fine at 16MHz in GPL!! When I get my first GENEVE back from Lou, the problem should be cured.
My joysticks don't work on this GENEVE either, but they did on my first one. Not even my Quickshot, and adapter, but Trevor Stephens' joystick worked on it perfectly.

At the AGM, I met a very helpful German, who also owns a

Myarc HFDCC. I got talking to him, and he said that he had got a Seagate 80MByte hard disk, fully formatted, and he had got Myarc Disk Manager 5 version 1.3. He has also modified his controller so that he's got an extra head select line which allows up to 15 heads to be addressed. He didn't have the information with him, but he said he would send it over (It hasn't arrived yet, and I'm worried!!). With the modification, and the new disk manager, I'll be able to get my pair of Seagate St-4096's formatted up to 80 Megs each. At the moment I'm struggling along with a full Seagate ST-251-1 40Meg drive, and piles of floppies with the new software. When I get it all set up, I will have 200Megabytes of TI software, all online and instantly available!!! I will then donate the odd 10 megabytes of memory to the bulletin board.

The use of the bulletin board has now ceased, and Trevor has stopped even setting it up. BUT, DONT DESPAIR!!!! When my exams finish on the 22nd of June, I'm going to get around to completing my TI projects. The first will be to get After Hours to answer the phone properly. The next is a 128 column word processor that will support colour printing etc.

I have also got ideas for a macro assembler that's icon driven, and will run with missing link, and let anyone do what they want with machine code, even if the only thing they know is the number of our microprocessor.

Another project is a CAD program that will introduce a new (Not another one!!) file format called Scaled Vector Format which will be loadable with little routines, into Missing Link, XHI, Advanced BASIC, or convertable to Picasso etc. I am also going to work on a decent printing program, now that I've got a 24 pin colour printer (excuse this poor quality which is actually printed with my Sept 7th 1984 Epson RX80!!).

I am going to write a program that will take Myart images and print them in a close simulation to 256 colours. I am also hoping to write a monochrome 360 * 360 dots per inch Hex Density Graphics dumper, which will print whatever I can translate. I might make it work with Picasso, or I might make it re-print in 24-pin mode, the graphics for 9 pin which have already been printed to disk with programs such as Picasso, TI-Artist, MacFlix, PagePro etc.

Something else I've just started working on is a very little program that takes less than 20 lines at the moment and allows Extended BASIC to plot points or draw lines on a small grid of 80 * 88 pixels. I t achieves it by re-defining the characters, so text is not displayable, unless you draw it using the line commands, but it will give console only people the chance of trying out graphics, if they haven't got 32K, and therefore can't run The Missing Link. I think I'll call my program The Little Link!!!

Well I have exceeded 200 lines (and I said it would be short!!! Oh Well!!!) In future articles, I will probably cover some of the things I've done at Polytechnic, such as Finite State Machines (which is what I got the A for!) Finite State Machines are a damn good way of analysing text and extracting individual words, or tokens etc.

Just to keep everyone going, here's a listing of a useless but interesting little demo!!! It needs some way to freeze and unfreeze the sprite motion, which is what the CALL LOAD(-32806,64) and CALL LOAD(-32806,16) is for. People without 32K could probably use Triton Super XB, and use the built in commands in that to do it.

Hopefully, there'll be more from me next issue!!!

All for now, from Richard Twyning.

P.S. I did the 28 column listing with a LIST "P1O":28: from Triton Super XB!!!

```
1 CALL MAGNIFY(3):: IS=40 ::
 TS=127 :: CALL INIT :: CALL
 LOAD(-31806,64)
2 CALL SPRITE(£18,76,2,96,67
,0,IS,£19,72,2,96,101,0,IS,£
20,68,2,96,85,0,IS,£21,64,2,
99,69,0,IS)
3 CALL SPRITE(£22,60,2,111,5
7,0,IS,£23,56,2,114,41,0,IS,
£24,48,2,130,38,0,IS,£25,52,
2,144,54,0,IS)
4 CALL SPRITE(£26,36,2,146,2
7,0,IS,£27,40,2,146,43,0,IS,
£28,44,2,146,59,0,IS)
5 CALL LOAD(-31806,16):: FOR
 D=65 TO 90 :: CALL CHARPAT(
D,A$):: CALL CHAR(D+32,A$)::
 NEXT D
10 CALL CHAR(36,"00000000010
20404091120404081807F2040808
0000000E0081EC82828C600FF")
11 CALL CHAR(40,RPT$("0",20)
&"7192926900FF08040201010000
001012D41814D200FF")
12 CALL CHAR(44,RPT$("0",10)
&"80402020100804040202FC"&RP
T$("0",18))
13 CALL CHAR(48,"28282814160
906010708102020408080222261A1
434C830C0F00804020201")
14 CALL CHAR(52,"8080"&RPT$(
"0",47))
15 CALL CHAR(56,RPT$("0",22)
&"0E314EB1A0"&RPT$("0",18)&"
03040A916BC4A8")
16 CALL CHAR(60,RPT$("0",10)
&"0102041A314AA418A04080040A
314AA418B0408000")
17 CALL CHAR(64,RPT$("0D",6)
&"0F1D2F25122946A418B07CFEC7
838383E3D39320C080")
18 CALL CHAR(68,"303030FC303
0303031391F0F01010000"&RPT$(
"0",12)&"30FCCE86FFFF80C6FE3
8")
19 CALL CHAR(72,RPT$("60",10
)&"E0E0"&RPT$("0",27))
20 CALL CHAR(76,"0303"&RPT$(
"0",64))
21 CALL CHAR(80,"01030303030
30303FF80C04060381C0EFC04050
50406020C1921213D050505C4")
22 CALL CHAR(84,"00804040800
00C12BA8ABAA2A1A1A1220000000
00000000000000")
23 CALL CHAR(88,"F84C0406030
1000000000000000000009364081
010C8F800000000000")
24 CALL LOAD(-31806,64):: CA
LL SPRITE(£15,80,2,30,114,0,
TS,£16,84,2,30,130,0,TS,£17,
88,2,46,120,0,TS):: CALL LOA
D(-31806,16)
30 PRINT "":" a demo by rich
ard twyning ":"":":"":":"":""
:"":":"":":"":":"":":"":":":
"":":":":":":":""
999 CALL LOAD(-31806,16):: P
RINT "     i blame it on int
el"
1000 CALL KEY(0,K,S):: IF S=
0 THEN 1000
```

P.S. Sorry about the £ signs!!

CASSETTE LIBRARY REPORT..... NICKY GODDARD

Well to start off with I would like to say thank you for voting me in again at the AGM in Derby, it is a great pleasure to once again serve the TI-99/4A community.

The cassette library has been quiet over the past 3 months but I hope it will perk up soon. There is a sale of surplus stock cassettes this time due to me being up to my eyeballs in cassettes. I hope by the next TI*MES I will have more to write about but for now I will leave you, see you next time.....

==========

PLUS STOCK CASSETTES SURPLUS STOCK CASSETTES SURPLUS STOCK CASS

==========

PRICES - 65p For the first cassette, 50p thereafter postage included.

EDUCATIONAL

Happy Math.

GAMES

Lionel and the Ladders, Battlestar Attack, Adventuremania, Shuttle Attack, Kat Traxx, Turn of card + 3 Card brag, Fun Pac 2, Sengoku Jedai, Pilot, Mania Demonstration, The count, Pharoes curse + 3D Noughts and crosses, Kong, Devils Island + Russian Roulette, 99 Vaders, Pirate Adventure, Quasimodo, Chalice + Penguin, TI Trek, Pentathlon, Guess + Snooker, Winging it + instructions.

GAMES COLLECTION. SPECIAL PRICE = 65p

Carols, O - Level maths, Joystick drawer, Biorythyms, Number scrabble, Tic Tac Toe, Word Scrabble, Xmas.
Hurry, Hurry, Hurry only 1 games collection in stock.

UTILITIES

Games Writers Pack 1 + manual, Games Writers pack 2 + manual, Teach Yourself Basic + manual, Beginners Basic Tutor + manual, Starter Pack 1 + manual, Starter Pack 2 + manual, TI Logo Sampler Programming Aids I, Graphics Creator, Toad Graphics, Graphics Package.

MORE OFFERS.....

Buy any 6 cassettes for £2.50 rather than £3.15.
or Buy any 10 cassettes for £4.50 rather than £5.15
This offer only applies to cassettes bought in lots of 6 or 10 any other cassettes are charged at 50p each.

ON THE HARDWARE FRONT..... MIKE GODDARD

The AGM at Derby brought one very welcome visitor in the form of Berry Harmsen of TI GEBRAIKERS the Dutch TI user group. Although it is always a pleasure to welcome Berry to our meetings this time was all the more memorable as he brought details of the long awaited 80 column display card developed by his group although it is in DIY form there was enough enthusiastic members there to get the project in motion here (please call me for the latest developments). Thank you Berry for sharing your groups projects with us.

Things have been fairly quiet regarding the exchange and upgrade of TI equipment for the last few months with not many members upgrading although several wanting to sell up. Although what they will move on to beats me as there are very few machines if any on the market today which are as easy to use as the TI.

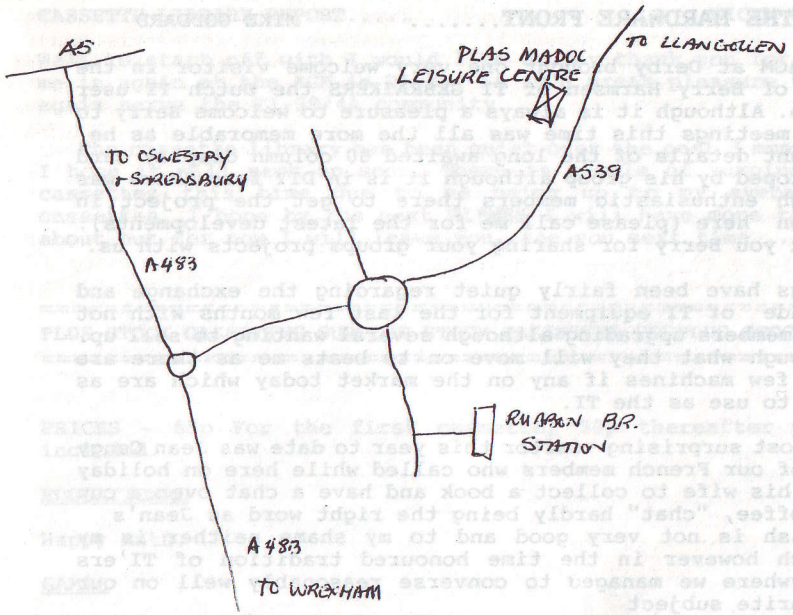
The most surprising visitor this year to date was Jean Cangy one of our French members who called while here on holiday with his wife to collect a book and have a chat over a cup of coffee, "chat" hardly being the right word as Jean's English is not very good and to my shame neither is my French however in the time honoured tradition of TI'ers everywhere we managed to converse reasonably well on our favourite subject

The latest on the 80 column card is that I have managed to obtain some prices for the printed circuit board which are obviously cheaper the more we buy so it would be usefull if those members interested could get in touch as soon as possible so that we can obtain the best price possible. I do hope to produce a kit of parts for the board within the near future.

See also elsewhere details of the third Wrexham workshop to be arranged for August.....

THE THIRD WREXHAM WORKSHOP

The third Wrexham workshop will take place on Saturday 15th AUGUST in the AQUA LOUNGE at the PLAS MADOC LEISURE CENTRE, ACREFAIR, WREXHAM. For those who have attended the previous workshops it is exactly the same venue and for those who haven't details below. We hope to have all the latest details of the 80 column card and anything else which is of current interest and of course we won't discourage anybody from bringing anything along that they want to demonstrate etc. The nearest Rail Station is RUABON and is two minutes walk from the venue. Buses run at fairly frequent intervals from Oswestry and Chester, please consult your local bus company for details.....



THE MATHEMATICAL JOUST.

Walter Allum

To keep his job, the mathematician at the court of San Ferian had to solve at least as many problems set by his rivals at other courts as they did his. This was in ye olden tymes, of course, when jousting about anything was fashionable. One problem gave him a fright. It was: find the smallest integer that becomes five times bigger when its last digit is transferred to the front. Or prove there isn't one. No leading zeros; not to end with 7 (this rules out the familiar 142857). Those who want to solve the puzzle on their own, read no further.

.....Ah, you've come back to see what we've done! Let us seek an n-digit integer X made up of an integer M with n-1 digits and a single digit D appended. Clearly, X= 10*M+D. After the digit transfer, 5*X=D*10^(n-1)+M. Eliminate M between these two equations to get X=(F/49)*D where F=10^n-1 which is a number composed of n nines. As D can't be 7, nothing cancels. So, for X to be an integer, 49 must divide F exactly.

A beginner in programming will find it a modest challenge to try larger and larger F until (hopefully) one does the trick. But, with no TI to help him, our hero felt his heart sink at the prospect. If you know of the earliest printed arithmetic (1478) and its involved treatment of (for us) simple multiplication and division, you will sympathize. Worse, at the time, the famous Fermat/Euler theorem was still in the future so that there was no assurance that the search would ever succeed. We know it will and can even set an upper limit on the n needed. Plain sailing now but if you want to check your result, see at the end of this article.

Now you want to know what happened at the San Ferian contest. Nothing! It was called off because their lordships decided to have a nice little war instead.

As a kind of "Son of San Ferian", we may set ourselves the task of finding integers in other number systems (e.g. binary, hexadecimal) that increase by factor k upon digit transfer. F is now not of such convenient form for simple working with the TI's denary facilities. Still, you may identify 10842(hex) and 104(hex) as the smallest no-leading-zero integers to increase 2x and 4x respectively. But, there's a surprise when you look at 3x and 5x

Here is a program to help your search. It does rather more than is needed for the general digit-switch problem. It determines the +ve values of N to ensure that B^N leaves a remainder C when divided by M (all integers, of course). Essentially, it is a simple search but there has to be provision for stopping the action in the insoluble cases and for recognising multiple solutions. It is an example of when it is preferable to sidestep the subtler math, useful at the desk but awkward to have in a program, by lengthier simple arithmetic well suited to a computer. In XB for compactness but all statements are simple Basic.

```
100 CALL CHAR(128,"007E007E007E0000") :: LB=LOG(10^12)
110 PRINT :: PRINT :: PRINT "   SOLUTION OF B^N"&CHR$(128)&"C(MOD M)  FOR N.":" I.
    E. N SUCH THAT B^N LEAVES REMAINDER C WHEN DIVIDED BY M."
120 PRINT :: INPUT " M,B,C.(B & C BOTH +VE <M )          ":M,B,C :: PRINT :: IF (B
    >=M)+(C>=M) THEN 120
130 CNT=0 ::  Z1$=" " :: Z2$=" " :: Z3$=" "
140 IF C<>1 THEN 150 :: Z1$="0"
150 N=0
160 BETA=B^N :: MU=M :: GOSUB 330
170 MU=MU/D :: BETA=B :: GOSUB 330 :: IF D=1 THEN 190 :: N=N+1 :: IF (N<2)+((N>=2
    )*(N*LOG(B)<LB)) THEN 160
180 PRINT :: PRINT  "ARITHMETIC TROUBLE" :: GOTO 300
190 N1=N :: IF N1<>0 THEN 200 :: CNT=1 :: V1=1
200 IF (C=1)*(N1>0) THEN 250 :: V=1
210 FOR N=1 TO N1+M-1 :: V=V*B :: V=V-M*INT(V/M) :: IF V=V1 THEN 240 :: IF V<>C
    THEN 230 :: Z1$=STR$(N)
220 IF N<N1 THEN 250
230 IF N<>N1 THEN 270 :: V1=V :: CNT=1 :: GOTO 280
240 IF Z1$=" " THEN 290 :: Z2$="+I*STR$(CNT)" :: Z3$="WHERE I IS
    ANY NON-NEGATIVE INTEGER"
250 PRINT :: PRINT :: PRINT :: PRINT " SOLUTION OF ":"  "&STR$(B);
    "^N"&CHR$(128)&STR$(C);"(MOD )";STR$(M);")":" ":" N=";Z1$;Z2$:"
     ":" "&Z3$
260 GOTO 300
270 IF CNT=0 THEN 280 :: CNT=CNT+1
280 NEXT N :: GOTO 300
290 PRINT :: PRINT " NO SOLUTION"
300 STOP
310 REM
320 REM  SUB FOR D, THE GCD OF BETA & MU
330 MN=BETA :: MX=MU :: IF MN<=MX THEN 340 :: A=MX :: MX=MN :: MN=A
340 RM=MX-MN*INT(MX/MN) :: IF RM<>0 THEN 350 :: D=MN :: GOTO 360
350 MX=MN :: MN=RM :: IF MN<MX THEN 340 :: D=MX
360 RETURN
370 REM
380 END
```

Completion of San Ferian problem.

n=42 (10^42 -1)/49 is 204081632653061224489795918367346938775 51

You will see it comprises only 41 digits and will remain so when multiplied by any D smaller than 5.

Then, to be consistent with the postulate that X has n=42 digits, and to give the right outcome upon transferring the final digit, a leading zero would be needed but this is ruled out. So,D=5 gives the smallest acceptable X. Others come from using 6, 8 and 9 (7 not allowed).

Thus, smallest X is 10204081632653061224489795918367346938775 5

With systems other than denary, evaluating X when n is large can be tedious unless you write yourself a program to do it.

```
100 CALL CLEAR :: RANDOMIZE
110 GOTO 120 :: A :: B :: C
:: I :: J :: V :: W :: CALL
SOUND
120 A1=110 :: AS=117 :: B1=1
23 :: C1=131 :: CS=139 :: D1
=147 :: DS=156 :: E1=165 ::
F1=175 :: FS=185 :: G1=196 :
: GS=208 :: A2=220
130 BF=233 :: B2=247 :: C2=2
62 :: CT=277 :: D2=294 :: DT
=311 :: E2=330 :: F2=349 ::
FT=370 :: G2=392 :: GT=415 :
: A3=440 :: AU=466
140 B3=494 :: C3=523 :: CU=5
54 :: D3=587 :: DU=622 :: E3
=659 :: F3=698 :: FU=740 ::
G3=784 :: GU=831 :: A4=880 :
: AV=932
150 !@P-
160 DISPLAY AT(7,4):"THE BAC
H MINI-CONCERT" :: DISPLAY A
T(10,10):"PRESENTS"
170 A=90 :: GOSUB 4340 :: DI
SPLAY AT(15,4):"  MINUET IN
G MINOR "
180 V=6 :: W=4 :: FOR I=1 TO
 2 :: GOSUB 1240
190 CALL SOUND(B,D3,V,BF,W)
200 CALL SOUND(B,D3,V,A2,W)
210 CALL SOUND(C,DU,V,BF,W,G
1,W)
220 CALL SOUND(B,F3,V,BF,W,G
1,W)
230 CALL SOUND(B,DU,V,BF,W,G
1,W)
240 CALL SOUND(B,D3,V,A2,W)
250 CALL SOUND(B,C3,V,A2,W)
260 CALL SOUND(C,D3,V,BF,W)
270 CALL SOUND(B,DU,V,BF,W)
280 CALL SOUND(B,D3,V,BF,W)
290 CALL SOUND(B,C3,V,G1,W)
300 CALL SOUND(B,AU,V,G1,W)
310 CALL SOUND(C,C3,V,A2,W)
320 CALL SOUND(B,D3,V,FS,W)
330 CALL SOUND(B,C3,V,FS,W)
340 CALL SOUND(B,AU,V,G1,W)
350 CALL SOUND(B,C3,V,G1,W)
360 CALL SOUND(C,A3,V,D1,W)
370 CALL SOUND(B,A3,V,D2,W)
380 CALL SOUND(B,A3,V,C2,W)
390 CALL SOUND(B,A3,V,BF,W)
400 CALL SOUND(B,A3,V,A2,W):
: GOSUB 1240
410 CALL SOUND(B,D3,V,B2,W)
420 CALL SOUND(B,D3,V,A2,W)
430 CALL SOUND(C,F3,V,D2,W,D
1,W)
440 CALL SOUND(B,G3,V,D2,W,D
1,W)
450 CALL SOUND(B,F3,V,D2,W,D
1,W)
460 CALL SOUND(B,DU,V,D2,W,G
1,W)
470 CALL SOUND(B,D3,V,D2,W,G
1,W)
480 CALL SOUND(C,DU,V,C2,W)
490 CALL SOUND(B,F3,V,A2,W)
500 CALL SOUND(B,DU,V,A2,W)
510 CALL SOUND(B,D3,V,F1,W)
520 CALL SOUND(B,C3,V,F1,W)
530 CALL SOUND(C,D3,V,BF,W)
540 CALL SOUND(C,G3,V,DS,W)
550 CALL SOUND(C,C3,V,A2,W,F
1,W)
560 CALL SOUND(3*C,AU,V,F2,W
,D2,W):: NEXT I
570 V=4 :: W=2 :: FOR I=1 TO
 2
580 CALL SOUND(C,D3,V,BF,W)
590 CALL SOUND(B,AU,V,BF,W)
600 CALL SOUND(B,C3,V,BF,W)
610 CALL SOUND(B,D3,V,BF,W)
620 CALL SOUND(B,E3,V,BF,W)
630 CALL SOUND(C,F3,V,A2,W)
640 CALL SOUND(C,G3,V,G1,W)
650 CALL SOUND(C,A4,V,F1,W)
660 CALL SOUND(C,AV,V,G1,W)
670 CALL SOUND(B,G3,V,E1,W)
680 CALL SOUND(B,A4,V,E1,W)
690 CALL SOUND(B,AV,V,C1,W)
700 CALL SOUND(B,G3,V,C1,W)
710 CALL SOUND(C,A4,V,F1,W)
720 CALL SOUND(B,G3,V,F1,W)
730 CALL SOUND(B,A4,V,F1,W)
740 CALL SOUND(C,F3,V)
750 CALL SOUND(B,F2,V,A2,W)
760 CALL SOUND(B,G2,V,A2,W)
770 CALL SOUND(B,A3,V,G1,W)
780 CALL SOUND(B,AU,V,G1,W)
790 CALL SOUND(B,C3,V,F1,W)
800 CALL SOUND(B,D3,V,F1,W)
810 CALL SOUND(C,DU,V,G1,W)
820 CALL SOUND(C,D3,V,F1,W)
```



```
830 CALL SOUND(C,C3,V,DS,W)
840 CALL SOUND(C,F3,V,D1,W)
850 CALL SOUND(C,AU,V,DS,W)
860 CALL SOUND(C,A3,V,F1,W)
870 CALL SOUND(C,AU,V,AS,W)
880 CALL SOUND(C,AU,V,D2,W)
890 CALL SOUND(C,AU,V,C2,W)
900 CALL SOUND(C,G2,V,D2,W,B
2,W)
910 CALL SOUND(B,D3,V,D2,W,B
2,W)
920 CALL SOUND(B,C3,V,D2,W,B
2,W)
930 CALL SOUND(C,D3,V,D2,W,B
2,W)
940 CALL SOUND(C,G2,V,C2,W)
950 CALL SOUND(B,DU,V,C2,W)
960 CALL SOUND(B,D3,V,C2,W)
970 CALL SOUND(C,622,V,262,W
)
980 CALL SOUND(B,392,V,233,W
)
990 CALL SOUND(B,587,V,233,W
)
1000 CALL SOUND(B,370,V,220,
W)
1010 CALL SOUND(B,523,V,220,
W)
1020 CALL SOUND(B,392,V,196,
W)
1030 CALL SOUND(B,466,V,196,
W)
1040 CALL SOUND(C,440,V,294,
W)
1050 CALL SOUND(B,440,V,220,
W)
1060 CALL SOUND(B,440,V,196,
W)
1070 CALL SOUND(B,440,V,185,
W)
1080 CALL SOUND(B,440,V,165,
W)
1090 CALL SOUND(B,294,V,147,
W)
1100 CALL SOUND(B,330,V,147,
W)
1110 CALL SOUND(B,370,V,147,
W)
1120 CALL SOUND(B,392,V,147,
W)
1130 CALL SOUND(B,440,V)
1140 CALL SOUND(B,466,V)
1150 CALL SOUND(C,523,V,156,
W)
1160 CALL SOUND(C,466,V,147,
W)
1170 CALL SOUND(C,440,V,131,
W)
1180 CALL SOUND(B,466,V,117,
W)
1190 CALL SOUND(A,523,V,117,
W)
1200 CALL SOUND(A,587,V,117,
W)
1210 CALL SOUND(C,392,V,131,
W)
1220 CALL SOUND(C,370,V,147,
W)
1230 CALL SOUND(3*C,392,V,29
4,W,233,W):: NEXT I :: GOTO
1380
1240 CALL SOUND(C,932,V,196,
W)
1250 CALL SOUND(C,880,V,196,
W)
1260 CALL SOUND(C,784,V,196,
W)
1270 CALL SOUND(C,880,V,175,
W)
1280 CALL SOUND(C,587,V,175,
W)
1290 CALL SOUND(C,587,V,175,
W)
1300 CALL SOUND(C,784,V,156,
W)
1310 CALL SOUND(B,392,V,156,
W)
1320 CALL SOUND(B,440,V,156,
W)
1330 CALL SOUND(B,466,V,156,
W)
1340 CALL SOUND(B,523,V,156,
W)
1350 CALL SOUND(C,587,V,147,
W)
1360 CALL SOUND(B,587,V,294,
W)
1370 CALL SOUND(B,587,V,262,
W):: RETURN
1380 W=1 :: A=100 :: DISPLAY
 AT(15,1):"       MUSETTE IN
 D    " :: GOSUB 4330 :: FOR
I=1 TO 2 :: V=6 :: GOSUB 157
0
```

```
1390 GOSUB 1570 :: V=3 :: GO
SUB 1620 :: GOSUB 1670 :: V=
6 :: GOSUB 1570
1400 GOSUB 1570 :: V=3 :: GO
SUB 1620 :: GOSUB 1710 :: NE
XT I
1410 FOR I=1 TO 2 :: V=3 ::
W=3 :: GOSUB 1740
1420 GOSUB 1740 :: GOSUB 177
0 :: GOSUB 1790
1430 GOSUB 1770
1440 GOSUB 1770 :: GOSUB 181
0 :: V=8 :: W=7 :: FOR J=1 T
O 2 :: V=V-1 :: W=W-1
1450 CALL SOUND(B,E3,V,E1,W)
1460 CALL SOUND(B,DU,V,E1,W)
1470 CALL SOUND(B,E2,V,E1,W)
1480 CALL SOUND(C,D3,V,E1,W)
1490 CALL SOUND(B,CU,V,E1,W)
1500 CALL SOUND(B,A4,V,E1,W)
1510 CALL SOUND(B,GU,V,E1,W)
1520 NEXT J :: V=5 :: W=2 ::
 GOSUB 1870 :: V=4 :: W=1
1530 GOSUB 1870 :: GOSUB 191
0 :: V=2 :: GOSUB 1950 :: GO
SUB 2000 :: V=10 :: W=6 :: G
OSUB 1570
1540 GOSUB 1570 :: V=5 :: W=
3 :: GOSUB 1620 :: GOSUB 167
0 :: V=10 :: W=6
1550 GOSUB 1570
1560 GOSUB 1570 :: V=7 :: W=
5 :: GOSUB 1620 :: V=5 :: W=
3 :: GOSUB 1710 :: NEXT I ::
 GOTO 2040
1570 CALL SOUND(C,A4,V,D1,W)
1580 CALL SOUND(A,G3,V,D1,W)
1590 CALL SOUND(A,FU,V,D1,W)
1600 CALL SOUND(A,E3,V,D1,W)
1610 CALL SOUND(A,D3,V,D1,W)
:: RETURN
1620 CALL SOUND(A,FT,V,FS,W)
1630 CALL SOUND(A,G2,V,G1,W)
1640 CALL SOUND(B,A3,V,A2,W)
1650 CALL SOUND(B,G2,V,31,W)
1660 CALL SOUND(B,FT,V,FS,W)
:: RETURN
1670 CALL SOUND(B,E2,V,E1,W)
1680 CALL SOUND(B,A3,V,A2,W)
1690 CALL SOUND(B,FT,V,FS,W)
1700 CALL SOUND(B,D2,V,D1,W)
:: RETURN
```

```
1710 CALL SOUND(B,E2,V,E1,W)
1720 CALL SOUND(B,A3,V,A2,W)
1730 CALL SOUND(C,D2,V,D1,W)
:: RETURN
1740 CALL SOUND(A,CU,V,A1,W)
1750 CALL SOUND(A,D3,V,A1,W)
1760 CALL SOUND(B,E3,V,A2,W)
:: RETURN
1770 CALL SOUND(B,A4,V,A1,W)
1780 CALL SOUND(B,E3,V,A2,W)
:: RETURN
1790 CALL SOUND(B,E3,V,A1,W)
1800 CALL SOUND(B,E3,V,A2,W)
:: RETURN
1810 CALL SOUND(A,D3,V,A1,W)
1820 CALL SOUND(A,CU,V,A1,W)
1830 CALL SOUND(A,B3,V,A2,W)
1840 CALL SOUND(A,A3,V,A2,W)
1850 CALL SOUND(B,B3,V,E1,W)
1860 CALL SOUND(B,E2,V,E1,W)
:: RETURN
1870 CALL SOUND(A,E3,V,E1,W)
1880 CALL SOUND(A,DU,V,E1,W)
1890 CALL SOUND(A,CU,V,E1,W)
1900 CALL SOUND(A,DU,V,E1,W)
:: RETURN
1910 CALL SOUND(B,E3,V,E1,W)
1920 CALL SOUND(B,GT,V,E1,W)
1930 CALL SOUND(B,A3,V,CS,W)
1940 CALL SOUND(B,D3,V,D1,W)
:: RETURN
1950 CALL SOUND(A,CU,V,E1,W)
1960 CALL SOUND(A,D3,V,E1,W)
1970 CALL SOUND(B,E3,V,E1,W)
1980 CALL SOUND(B,A3,V,A1,W)
1990 CALL SOUND(B,D2,V,D1,W)
:: RETURN
2000 CALL SOUND(A,CT,V,CS,W)
2010 CALL SOUND(A,D2,V,D1,W)
2020 CALL SOUND(B,E2,V,E1,W)
2030 CALL SOUND(C,A2,V,A1,W)
:: RETURN
2040 A=110 :: DISPLAY AT(15,
1):"    LITTLE PRELUDE IN F"
 :: GOSUB 4330 :: V=4 :: W=6
2050 CALL SOUND(A,A2,W,F1,W)
2060 CALL SOUND(A,C3,V,A2,W,
F1,W)
2070 CALL SOUND(A,A3,V,C2,W,
F1,W)
2080 CALL SOUND(A,C3,V,C2,W,
F1,W)
```



```
2090 CALL SOUND(A,F2,V,A2,W,
F1,W)
2100 CALL SOUND(A,C3,V,A2,W,
F1,W)
2110 CALL SOUND(A,A3,V,C2,W,
F1,W)
2120 CALL SOUND(A,C3,V,C2,W,
F1,W)
2130 CALL SOUND(A,F2,V)
2140 CALL SOUND(A,D3,V)
2150 CALL SOUND(A,AU,V,D2,W,
F1,W)
2160 CALL SOUND(A,D3,V,D2,W,
F1,W)
2170 CALL SOUND(A,F2,V,BF,W,
F1,W)
2180 CALL SOUND(A,D3,V,BF,W,
F1,W)
2190 CALL SOUND(A,AU,V,D2,W,
F1,W)
2200 CALL SOUND(A,D3,V,D2,W,
F1,W)
2210 CALL SOUND(A,G2,V)
2220 CALL SOUND(A,E3,V)
2230 CALL SOUND(A,AU,V,G1,W,
F1,W)
2240 CALL SOUND(A,E3,V,G1,W,
F1,W)
2250 CALL SOUND(A,G2,V,BF,W,
F1,W)
2260 CALL SOUND(A,E3,V,BF,W,
F1,W)
2270 CALL SOUND(A,AU,V,G1,W,
F1,W)
2280 CALL SOUND(A,E3,V,G1,W,
F1,W)
2290 CALL SOUND(A,F3,V,A2,W,
F1,W)
2300 CALL SOUND(A,C3,V,A2,W,
F1,W)
2310 CALL SOUND(A,A3,V,A2,W,
F1,W)
2320 CALL SOUND(A,C3,V,A2,W,
F1,W)
2330 CALL SOUND(A,F2,V)
2340 CALL SOUND(A,C2,V)
2350 CALL SOUND(A,A2,V)
2360 CALL SOUND(A,C2,V)
2370 CALL SOUND(A,F1,V)
2380 CALL SOUND(A,C2,V)
2390 CALL SOUND(A,F3,W,C3,W,
A2,V)
```

```
2400 CALL SOUND(A,F3,W,C3,W,
C2,V)
2410 CALL SOUND(A,F3,W,A3,W,
F1,V)
2420 CALL SOUND(A,F3,W,A3,W,
C2,V)
2430 CALL SOUND(A,F3,W,C3,W,
A2,V)
2440 CALL SOUND(A,F3,W,C3,W,
C2,V)
2450 CALL SOUND(A,F1,V)
2460 CALL SOUND(A,D2,V)
2470 CALL SOUND(A,F3,W,D3,W,
BF,V)
2480 CALL SOUND(A,F3,W,D3,W,
D2,V)
2490 CALL SOUND(A,F3,W,AU,W,
F1,V)
2500 CALL SOUND(A,F3,W,AU,W,
D2,V)
2510 CALL SOUND(A,F3,W,D3,W,
BF,V)
2520 CALL SOUND(A,F3,W,D3,W,
D2,V)
2530 CALL SOUND(A,G1,V)
2540 CALL SOUND(A,E2,V)
2550 CALL SOUND(A,G3,W,E3,W,
BF,V)
2560 CALL SOUND(A,G3,W,E3,W,
E2,V)
2570 CALL SOUND(A,AV,W,E3,W,
G1,V)
2580 CALL SOUND(A,AV,W,E3,W,
E2,V)
2590 CALL SOUND(A,G3,W,E3,W,
BF,V)
2600 CALL SOUND(A,G3,W,E3,W,
E2,V)
2610 CALL SOUND(A,A4,W,C3,W,
F2,V):: V=4 :: W=2
2620 CALL SOUND(A,C2,V)
2630 CALL SOUND(A,A2,V)
2640 CALL SOUND(A,C2,V)
2650 CALL SOUND(A,F1,V)
2660 CALL SOUND(A,C1,V)
2670 CALL SOUND(A,A1,V)
2680 CALL SOUND(A,C1,V)
2690 CALL SOUND(A,F1,V)
2700 CALL SOUND(A,A4,V,F1,W)
2710 CALL SOUND(A,F3,V,G1,W)
2720 CALL SOUND(A,A4,V,G1,W)
2730 CALL SOUND(A,C3,V,A1,W)
```

```
2740 CALL SOUND(A,A4,V,A1,W)
2750 CALL SOUND(A,F3,V,AS,W)
2760 CALL SOUND(A,A4,V,AS,W)
2770 CALL SOUND(A,C3,V,C1,W)
2780 CALL SOUND(A,G3,V,C1,W)
2790 CALL SOUND(A,F3,V,D1,W)
2800 CALL SOUND(A,G3,V,D1,W)
2810 CALL SOUND(A,C3,V,E1,W)
2820 CALL SOUND(A,G3,V,E1,W)
2830 CALL SOUND(A,E3,V,C1,W)
2840 CALL SOUND(A,G3,V,C1,W)
2850 CALL SOUND(A,A3,V,D1,W)
2860 CALL SOUND(A,F3,V,D1,W)
2870 CALL SOUND(A,E3,V,E1,W)
2880 CALL SOUND(A,F3,V,E1,W)
2890 CALL SOUND(A,A3,V,F1,W)
2900 CALL SOUND(A,F3,V,F1,W)
2910 CALL SOUND(A,D3,V,G1,W)
2920 CALL SOUND(A,F3,V,G1,W)
2930 CALL SOUND(A,A3,V,A2,W)
2940 CALL SOUND(A,E3,V,A2,W)
2950 CALL SOUND(A,D3,V,BF,W)
2960 CALL SOUND(A,E3,V,BF,W)
2970 CALL SOUND(A,A3,V,C2,W)
2980 CALL SOUND(A,E3,V,C2,W)
2990 CALL SOUND(A,C3,V,A2,W)
3000 CALL SOUND(A,E3,V,A2,W)
3010 CALL SOUND(A,F2,V,BF,W)
3020 CALL SOUND(A,D3,V,BF,W)
3030 CALL SOUND(A,C3,V,A2,W)
3040 CALL SOUND(A,D3,V,A2,W)
3050 CALL SOUND(A,G2,V,G1,W)
3060 CALL SOUND(A,D3,V,G1,W)
3070 CALL SOUND(A,AU,V,F1,W)
3080 CALL SOUND(A,D3,V,F1,W)
3090 CALL SOUND(A,G2,V,E1,W)
3100 CALL SOUND(A,C3,V,E1,W)
3110 CALL SOUND(A,AU,V,C1,W)
3120 CALL SOUND(A,C3,V,C1,W)
3130 CALL SOUND(A,F2,V,F1,W)
3140 CALL SOUND(A,C3,V,F1,W)
3150 CALL SOUND(A,A3,V,E1,W)
3160 CALL SOUND(A,C3,V,E1,W)
3170 CALL SOUND(A,F2,V,D1,W)
3180 CALL SOUND(A,AU,V,D1,W)
3190 CALL SOUND(A,A3,V,G1,W)
3200 CALL SOUND(A,AU,V,G1,W)
3210 CALL SOUND(A,C2,V,E1,W)
3220 CALL SOUND(A,AU,V,E1,W)
3230 CALL SOUND(A,G2,V,C1,W)
3240 CALL SOUND(A,AU,V,C1,W)
3250 CALL SOUND(A,C2,V,F1,W)
3260 CALL SOUND(A,A3,V,C1,W)
3270 CALL SOUND(A,F2,V,A1,W)
3280 CALL SOUND(A,A3,V,C1,W)
3290 CALL SOUND(A,C2,V,F1,W)
3300 CALL SOUND(A,A3,V,A1,W)
3310 CALL SOUND(A,F2,V,C1,W)
3320 CALL SOUND(A,A3,V,F1,W)
3330 CALL SOUND(A,D2,V,AS,W)
3340 CALL SOUND(A,A3,V,AS,W)
3350 CALL SOUND(A,F2,V,BF,W)
3360 CALL SOUND(A,A3,V,BF,W)
3370 CALL SOUND(A,D2,V)
3380 CALL SOUND(A,F2,V)
3390 CALL SOUND(A,A3,V,A2,W)
3400 CALL SOUND(A,C3,V,A2,W)
3410 CALL SOUND(A,D2,V,G1,W)
3420 CALL SOUND(A,AU,V,D1,W)
3430 CALL SOUND(A,G2,V,AS,W)
3440 CALL SOUND(A,AU,V,D1,W)
3450 CALL SOUND(A,D2,V,G1,W)
3460 CALL SOUND(A,AU,V,AS,W)
3470 CALL SOUND(A,G2,V,D1,W)
3480 CALL SOUND(A,AU,V,G1,W)
3490 CALL SOUND(A,E2,V,C1,W)
3500 CALL SOUND(A,AU,V,C1,W)
3510 CALL SOUND(A,G2,V,C2,W)
3520 CALL SOUND(A,AU,V,C2,W)
3530 CALL SOUND(A,E2,V)
3540 CALL SOUND(A,G2,V)
3550 CALL SOUND(A,AU,V,BF,W)
3560 CALL SOUND(A,D3,V,BF,W)
3570 CALL SOUND(A,F2,V,A2,W)
3580 CALL SOUND(A,C3,V,A2,W)
3590 CALL SOUND(A,A3,V,C1,W)
3600 CALL SOUND(A,C3,V,C1,W)
3610 CALL SOUND(A,F3,V,A2,W)
3620 CALL SOUND(A,C3,V,A2,W)
3630 CALL SOUND(A,A3,V,C1,W)
3640 CALL SOUND(A,F2,V,C1,W)
3650 CALL SOUND(A,G1,W)
3660 CALL SOUND(A,AU,V,G1,W)
3670 CALL SOUND(A,G2,V,C1,W)
3680 CALL SOUND(A,AU,V,C1,W)
3690 CALL SOUND(A,E3,V,G1,W)
3700 CALL SOUND(A,AU,V,G1,W)
3710 CALL SOUND(A,G2,V,C1,W)
3720 CALL SOUND(A,E2,V,C1,W)
3730 CALL SOUND(A,F1,W)
3740 CALL SOUND(A,A3,V,F1,W)
3750 CALL SOUND(A,F2,V,C1,W)
3760 CALL SOUND(A,A3,V,C1,W)
3770 CALL SOUND(A,D3,V,F1,W)
```



```
3780 CALL SOUND(A,A3,V,F1,W)
3790 CALL SOUND(A,F2,V,C1,W)
3800 CALL SOUND(A,D2,V,C1,W)
3810 CALL SOUND(A,E1,W)
3820 CALL SOUND(A,G2,V,E1,W)
3830 CALL SOUND(A,E2,V,C1,W)
3840 CALL SOUND(A,G2,V,C1,W)
3850 CALL SOUND(A,C3,V,E1,W)
3860 CALL SOUND(A,G2,V,E1,W)
3870 CALL SOUND(A,E2,V,C1,W)
3880 CALL SOUND(A,C2,V,C1,W)
3890 CALL SOUND(A,D1,W)
3900 CALL SOUND(A,F2,V,D1,W)
3910 CALL SOUND(A,D2,V,C1,W)
3920 CALL SOUND(A,F2,V,C1,W)
3930 CALL SOUND(A,G2,V,AS,W)
3940 CALL SOUND(A,A3,V,AS,W)
3950 CALL SOUND(A,AU,V,G1,W)
3960 CALL SOUND(A,D2,V,G1,W)
3970 CALL SOUND(A,G2,V,E1,W)
3980 CALL SOUND(A,A3,V,E1,W)
3990 CALL SOUND(A,AU,V,C1,W)
4000 CALL SOUND(A,E2,V,C1,W)
4010 CALL SOUND(A,F2,V,A2,W)
4020 CALL SOUND(A,G2,V,A2,W)
4030 CALL SOUND(A,A3,V,F1,W)
4040 CALL SOUND(A,C2,V,F1,W)
4050 CALL SOUND(A,D2,V,BF,W)
4060 CALL SOUND(A,F2,V,BF,W)
4070 CALL SOUND(A,A3,V,BF,W)
4080 CALL SOUND(A,C3,V,BF,W)
4090 CALL SOUND(A,AU,V)
4100 CALL SOUND(A,A3,V)
4110 CALL SOUND(A,G2,V)
4120 CALL SOUND(A,AU,V)
4130 CALL SOUND(A,D3,V)
4140 CALL SOUND(A,F3,V)
4150 CALL SOUND(A,E3,V)
4160 CALL SOUND(A,D3,V)
4170 CALL SOUND(A,C3,V)
4180 CALL SOUND(A,E3,V)
4190 CALL SOUND(A,G3,V)
4200 CALL SOUND(A,AV,V)
4210 CALL SOUND(C,AV,V,E3,W,
C3,W)
4220 CALL SOUND(C*2,AV,30)
4230 CALL SOUND(B,A4,V,F3,W,
C3,W)
4240 CALL SOUND(A,AU,V,D1,W)
4250 CALL SOUND(A,G3,V,E3,W,
D1,W)
4260 CALL SOUND(A,A3,V,AS,W)
4270 CALL SOUND(A,F3,V,D3,W,
AS,W)
4280 CALL SOUND(A,G2,V,C1,W)
4290 CALL SOUND(A,F3,V,C3,W,
C1,W)
4300 CALL SOUND(A,G2,V,C1,W)
4310 CALL SOUND(A,E3,V,AU,W,
C1,W)
4320 CALL SOUND(C*4,F3,V,C3,
W,A3,W):: GOSUB 4330 :: GOTO
 170
4330 FOR I=1 TO 1000 :: NEXT
 I
4340 A=INT(RND*40+A):: B=A+A
 :: C=B+B :: RETURN
```

>HINTS AND HELPS<

Have you ever wanted to put comments on a catalog listing, especially you are passing the disk to a friend? One way to do it is to write the comments in with pencil but a neater way is to use TI Writer. Here's how:

1_When you go into DM1000 to catalog the disk, first change the output device. Press FCTN 3 when the main menu screen appears and change "PIO" default to "DSKn.filename". You don't want printer codes and you don't want to save this change permanently, so press "N" for both of these questions.

2_Go through the normal proceedure to catalog your disk. Once the catalog is displayed, press FCTN 7 to print. However, instead of printing to the printer, you will print to the disk, giving you a DV80 file under the name you gave it in step 1.

3_You can now access this file with TI Writer to add your comments and for printing out. It is suggested that you turn off word wrap (CTRL 0). If comments take more than one line then you will have to insert lines (FCTN 8).

By using this method you can make the listing more informative to the person getting it. For example, programs that are related can be grouped together regardless as to their names.

A couple of neat ideas from Stan Corbin out of the ROM newsletter:
To remove a strip of labels from your printer when you have finished printing labels, place a piece of paper between the labels and the platen cover. The label strip can now be pulled out backwards from around the platen roller. (Otherwise you can gum up your printer and possibly have an expensive repair to do! Printer labels are designed to only roll up!).

Many programs come with 8x5.5 sized manuals. These can be an ideal place to store a backup copy of the program. Paste a disk sleeve onto the inside cover, with the opening adjacent to the binding of the book. This prevents the disk from falling out but still allows access when the manual is opened.

And one for newsletter editors or anyone who needs to do cut-and-paste :

FaberCastell makes a glue stick called the UHU STIC. The thing that makes it so unique is that is goes on PURPLE but dries clear. This makes it ideal for cut-and-paste type operations. [There is also a blue equivalent from Pelikan, called Peli fix Effect - both have been used in making up past TI*MES].

Ramdisk owners, here is a short program to drop you into Extended Basic without doing the time consuming search for a LOAD program on drive one. This has appeared in various newsletters.

```
10 CALL INIT :: CALL LOAD(-31952,255,0, 255,0)
20 END
```

= = = = = = = = = = = = = = = = =

DISK LIBRARY NOTES:

The disk library is listed on text files contaned on FOUR SSSD disks, and a copy is yours in return for four blank disks and return postage. Please write to Stephen Shaw, 10 Alstone Road, STOCKPORT, Cheshire, SK4 5AH.

The copying fee for disks is one pound per disk side - you supply the disks. There is a handling charge of one pound per order, for any number of disks. If you prefer, the library can supply disks for an additional one pound each - these are guaranteed to be in good condition.

Listing on next page---->

The following are disks added since the last issue:

>HARRISON SOFTWARE UTILITIES VOLUME 2: Machine code to CALL LINK to in your XB programs: MENUMAKER will quickly produce a very neat menu and scan for keypresses from DATA statements; plus two utilities to fill string and numeric arrays from DATA statements- no longer need you wait several minutes while a large string array is filled from DATA statements!

>EASY DATA -COMMERCIAL PROGRAM- NOT FREEWARE, NOT PUBLIC DOMAIN!!! from Harrison Software, Vn 1.01. A program which allows you to use easy/peasy DATA statements to store your data. The machine code links supplied allow you to store the data into arrays, sorted on any two fields. What you then do with it is up to you. Very efficient, very easy to use bare bones data base. For anyone struggling with TI BASE but for whom PRK is too restricted and XB on its own too slow. THIS IS A COMMERCIAL OFFERING. PRICE INCLUDING DISK IS FOUR POUNDS.

>UTIL 28 is now filled with an additional very useful program which ARCHIVEs Program files for use with EdAs Type 5 loaders (RUN PROGRAM FILE machine code). This program has two plusses- files are smaller than with ARCHIVER, and -wow- as they contain a run-time unarchiver, they will run in their smaller format. The only negative is that only machine code program files work, and a few of those fail (too large- exceptional! - and some with tiny secondary files). As an example, BARRAGE occupies 51 sectors. Archiver takes it to 45 sectors which need unarchiving before use. PFC (Program File Compiler) reduces Barrage to 44 sectors, which you can RUN immediately. By K Holtman of Holland.

>OPA CATALOGUE. A catalogue of hardware from Canada. All very low run production by one man, but some exceptionally interesting stuff. Fancy a 600k+ preprogrammed module? Like to run ExBas, Multiplan, Logo, Mini Memory, without changing the module? Lots of stuff in this one, and very subject to change (eg improvements).

The following are some older disks, details extracted from the disk catalogue, firstly from the GAMES lists:

>BEHNKE. Two files for Tunnels of Doom and some XB games- 10pin bowls, breakout, forestfire and skyrescue. John is in the Chicago group and wrote the Tunnels of Doom editor sold by Asgard.

>CHINESE CHESS. Full disk game for TWO players. Astonishing use of XB- see what XB is capable of, and perhaps learn a new game of chess... rules are included.

>CONTRACT BRIDGE Vn 3.01 by John H Bull, (TWO DISK)- play against your computer, who plays according to the rules but not too well. Replay hands to improve your score. Fifty preplayed hands supplied.
You need to know Bridge or have the rules available as the "tutorial" section isn't.

>CRPYTOGRAM-SONG VERSES. One hundred song verses for the computer to encode (simple replacement) and for you to decode. By Jim Peterson. You can add your own verses too.

>CRYPTOGRAM-BIBLE VERSES. As above but the contents differ!

>GAMES-1 : Revised Oct 1986: From time to time a program is so thoroughly revised that I have to amend a disks contents - sorry, but you get a better service this way than having no revsions! This disk contains revised files for WARGAME and for FROGS but in the process I have had to move BASEMENTS! and Air Traffic Control on to Games-5 below. Bonus: A new game, 3D Noughts and Crosses! Entire disk written by Ron Johnson (UK).

>GAMES 5 : This first significant FREEWARE arcade game in machine code! BUZZARD BATE is a variant of PAC MAN but faster than the module. A running man avoids four buzzards and has a few flame throwers to beat them off. Not too hard but an attractive program. Also, AIR TRAFFIC and BASEMENTS! (a long text adventure) moved from GAMES-1, possibly the only XB Chess playing program (OK it plays badly...), 4-in-a-row, Aardvark, Knights and Solitaire. Schmoo.

>GAMES 6: American Monopoly, Chainlink, Cyber/dice, Jail, and in m/c: Vader (singular!) and SORGON I, the classic computer chess program VERY SLOW but a good player. Depthcharge.

>GAMES 7. 3D Os&Xs (TI-prog), BREAKOUT(M-code),CAKE adventure (easy),Cannibals, Jumper (by S Michel), Over-reaction, Switch, and Torpedo Alley.
>CO LIST (PLUS TXB). The program by Tony McGovern which will prepare program LISTS 28 columns wide- just like they appear on screen. Ideal for print outs that someone has to key in! Together with a fast action bagatelle type game which has lots of options.

> COMIC 1: Animation editor with ENGLISH documentation, allows you to chain up to 100 TI Artist pics together for SMOOTH animation effects in a machine code environment. Includes ExBas loader. Includes sample animations: Two TI Artist instances come to life- a Ghost and Pluto. And if you enjoyed Ray Kazmers picture of Garfield and Odie on page 42 of TI*MES issue 16, take a look at Ray's animated version!
>COMIC SHOW Vn 4.0 (1988): This is the latest version of COMIC, with English docs by the author, and 3 additional samples of animation. (PPYJAMA demo will not load with Funlweb- use the load menu supplied on this disk)

>CREATIVE FILING SYSTEM by Mark Beck, THREE DISKS PLEASE, Version 7. Either the best or second best database program for the TI (opinions vary- some put PRBASE first). CFS is full of useful utilities and has math capability. Lots of docs on disk.
>DEMO DISK containing: A machine code program giving you key-press access to speech extracted from some TI Modules; an animated space picture; three programs similar to mini-mem LINES, called LINES, TRIANGLES and RECTANGLES, and a logo-type program called ROSE, which must be loaded from Funlweb Load Option 3-good inputs for Rose are 13 then 313. AND ALSO a program to print to printer an 8x10 pic of the Mona Lisa. LIGHTSHOW- feed music into tape socket; SCRATCH music maker(music?); and MUSIC DEMO with themes from three modules- Midnite Mason, Cerberus, and Demon Driver.

>DISK UTILITIES Vn 4.12 by John Birdwell. A very complete disk utility which now allows you to change the name of a file when copying (think carefully about this!), check free space on destination disk BEFORE copying starts, sector edit changes in inverse to make them stand out... etc etc.
You may check a disk for bad sectors (non destructive) and mark out bad sectors from the BitMap without reinitialising the whole disk. Add comments to files, add date to disks. Excellent printout showing location of FDRs and each file segment. Phew! Excellent program.
Plus Q4CAT, which will read 4 disks and print the catalogues in 4 columns of condensed print.

>DSKU/M..The same as the above, but modified for owners of Myarc disk controllers who do not have either an 80 track eprom or a Geneve.

>DM1000. VN 3.5 Possibly the widest used disk manager and certainly a classic in the TI world. From Canada, this program can deal with any disk controller, and will catalogue, initialise, and rename disks, list,copy, move, delete, protect, unprotect, and view files. Although DM1000 is included on the FUNLWEB disks, you need to order THIS item to obtain the DM1000 documentation.
- *** The Ottawa group have advised June 88 that versions of DM1000 over Vn 3.5 may not be reliable, and you should not under ANY circumstances use Vn 4.0. You will need this 3.5 disk for the docs, AND will find Vn 3.5 on the Funlweb 4.1 disk (rewritten by Tony).

>DM1000 SOURCE CODE TO VERSION 3.5. TWO DISKS REQUIRED.

>DRAWMASTER V 1.2 from France, with English docs and windowed choices! Cheaper than TI Artist, but fully compatible with it-loads AND saves TI Artist _P and _C files. Some menu choices appear to be inactive. Press 1,2 or 3 to pull down a menu, and experiment. XB or EA5 load. Unique compact disk format (no not those!) available as an option, saves pictures WITH COLOUR in IV254 files, can be lots less than 50 sectors of TI Artist. No text available as far as I can see, unless you draw it! TWO DISKS.

>ENHANCED DISPLAY PACKAGE Vn 2.2 from Paragon Computing, programmer C A Provance. TRUE Freeware! IBM style- useful documentation on disk, after that you get what you pay for! Immediately usable and good demo program. This disk contains a program which places machine code utilities into memory for your XB programs to use by means of CALL LINKS.
There is a clock WITH ALARM!, windows, and display commands are amended for both 32 and 40 column screens. There are routines to save and load screen displays, PEEKV, GTEXT, and a much extended and useful CHRSET. Disk contains 15 pages of docs and a good demo program.
IMPORTANT: EDP supplied by me is configured for 50 cycle mains: the clock/alarm runs quite accurately ON 50 CYCLE MAINS. It will NOT run accurately on US 60 Hz mains. The electricity supply in the UK is required to maintain a short term frequency accuracy of 2%. This clock has been measured as better than .5%, well within supply constraints.

>EDP Vn 2.4US: Please request by THIS full name! A sightly unbugged version but with US clock-just runs slowly on UK consoles! The author has kept no record of amendments and nothing too drastic seems to have changed.

= =

MACHINE CODE

A Graphics Programming Language (GPL) Primer
Routine to Read/Edit Text from KSCAN by Mack McCormick

I have always avoided delving into the study of GPL because I felt it was too difficult, cumbersome, executed too slowly, and had little to offer. Boy, was I wrong. It makes writing routines used by BASIC a snap in assembler. For example, I recently needed a routine to read text from the screen which would allow full editing including erase, insert, delete, quit, bonk tone at right margin, and enter/up arrow/down arrow. I also wanted the neat auto-repeat feature used by TI where there is a slight pause before the key takes off repeating.

I began to consider writing the routine but then remembered that an identical routine resided in GPL in GROM (Graphics Read Only Memory) in the console. I first consided using the routine from GROM but then remembered that it added the screen offset of >60 to each character and I didn't need that. I could have done some fancy trick to make it work but decided to convert GPL to 9900 assembler code.

You'll find two programs here. One to link you to the routine in console GROM from a CALL LOAD from E/A BASIC and the identical (almost) routine in 9900 assembler code ready to link into any program you may have that needs this utility.

I've included in the 9900 routine the actual GPL code used by the Pre-Scan routine of the monitor so you may see what conversions were necessary. Try reading the GPL instructions (marked with three *) to get a flavor for GPL). If the GPL gets in the way if using the routine in your program you may delete the GPL statements though they will have no effect if they remain.

It has really become obvious to me why TI invented GPL though I used to condem them for it. The major reason is that it saves about 41% more code than straight assembler. GROM as you may know is only used by TI and is a chip which supplies a byte at a time to a memory mapped address and auto-increments to the next byte (like VDP RAM) unless you change the address to be read from. It's a great way to save memory.

TI calls it medium speed memory. It is 6K bytes big and resides on 8K boundaries. It is an ideal medium to hold the console BASIC routines because the TMS9900 CPU chip in the console can only directly address 64K. The GPL actually does not execute any code. GPL is interpreted in console ROM beginning at >0024 and extending to >D18. This interpreter is straight assembler code which acts as directed by the GPL bytes coming from the GROM. Hence you see one reason TI BASIC is slow.

It is an interpreted by GPL and GPL is interpreted by assembler. Two interpretations! Instructions in GPL usually have two operands and most instructions can access RAM, GROM, or VDP RAM. Most instructions are single byte operands unless the operand is preceded by a D for double operand. GPL uses two stacks a data stack at >83A0 and a subroutine address stack at >8380 (this allows arbitrary nesting of subroutines). Here are a few types of instructions:

```
DATA TRANSFER  -Single/Double Byte
               -Block to Block
               -Formatted Block Moves
ARITHMETIC     -add, subtract, multiply, divide, negate, absolute val.
LOGICAL        -AND, XOR, Shifts.
CONDITION      -Arithmetic and Logical
BRANCHING      -Conditional/Uncond
BIT MANIPULATION -Set, reset, test.
SUBROUTINE     -Call, Return
STACK OPNS     -Push and Pop
MISC           -Random Number, KSCAN, Coincidence Detection, Sound, I/O
```

The closest language to GPL is assembler and any experienced assembler programmer should have little difficulty learning GPL. One major difference is the use of MACRO instructions by the assembler such as REPEAT....UNTIL and IF....THEN....ELSE. Very similar in this respect to 99000 assembly language.

A few words about how memory is addressed. Here are a few of the most common ways and their syntax.

```
     5 represents the decimal byte 5.
   >33 represents hex 33.
110011 represents binary 110011.
  5506 represents the decimal number 5506.
  :A: is the ASCII equivalent >41.
```

Well this has been a *very* general overview of GPL. Let's look at some actual GPL source code and my intrepretation of the 9900 assembler equivalent. This routine could have been shortened but I tried to keep it as close to GPL as possible. Hope you enjoy it. If you have questions just ask. My 6 GPL manuals cover thousands of pages and we have just skimmed the surface here. I plan to write a GPL disassembler and interpreter to convert GPL to 9900 object code within the next six months if my schedule permits. That should make the job easy!


```
       DEF  START
       REF  KSCAN,VSBW,VSBR,GPLLNK

* JUST A LITTLE ROUTINE TO TEST SUBROUTINE *
START  LWPI WS
       MOVB @H00,@KEYVAL SCAN ENTIRE KEYBOARD
LOOP   BL   @READLN
       DATA >002,>2FE    START, END POSITONS
       JMP  LOOP

*******************************************************************************
* This is the console GPL READLN routine at (>2A42 in GROM 1) converted to 9900
* assembler. Interprets BACKSPACE, INSERT, DELETE, and FORWARD. Uses SCRATCH
* PAD RAM. Total number of characters may be limited by changing the start
* value of ARG+2 (upper limit) and entering at READL1. VARW is the start address
* of the field. VARA is the current highest write address.
* Entering at READL1 allows us to pre-specify the minimum number of characters
* to be read for default creation.
* Entering at READ00 allows specification of the initial cursor position. In
* this case ARG+6 has to be set to the cursor position and ARG+4<>0.
* Programmer responsibility to insure that VARW <= ARG+6 <= VARA <= ARG+2
* ARG+4 indicates if the line has been changed. If so, ARG+4=0.
* This is a possible call:
* BL   @READLN
* DATA >1DF,>35D   LOWER,UPPER SCREEN LIMITS
*******************************************************************************

* EQUATES *
WS     EQU  >8300        MY WORKSPACE
ARG    EQU  >835C
VARW   EQU  >8320        ABS LOWER LIMIT
VARA   EQU  >832A        CURRENT END OF LINE
TEMP   EQU  0            R0 USED FOR TEMP STORAGE
TEMP1  EQU  1            R1 USED FOR ADDL TEMP STORAGE
R1LB   EQU  WS+3
TEMP2  EQU  2
TEMP3  EQU  3
TIMER  EQU  >8379        VDP TIMER INC EVERY 1/60 SEC.
KEYVAL EQU  >8374        KEYBOARD TO SCAN
RKEY   EQU  >8375        KEY CODE
STATUS EQU  >837C        GPL STATUS BYTE

* CONSTANTS * (Should be EQU with byte values in code to save memory.)
H00    BYTE 0
H01    BYTE 1
HFF    BYTE >FF
H508   DATA 508
H60    DATA 60
H14    BYTE 14
H766   DATA 766

BREAK  BYTE >02
DLETE  BYTE >03
INSRT  BYTE >04
CLRLN  BYTE >07
BACK   BYTE >08
FORW   BYTE >09
DOWN   BYTE >0A
MVUP   BYTE >0B
CHRTN  BYTE >0D
CURSOR BYTE >1E
SPACE  BYTE >20
```

```
VARV   BYTE 0              (This is at >8301 in GPL but I use >8300 for workspace)
VAR1   DATA 0              AUTO REPEAT COUNTER (This is 1 byte at >830D in GPL)

EVEN

READLN
* The GPL code stores >35D at ARG+2 but to give more utility replaced with the
* next two lines of code.
***    DST  >35D,@ARG+2  GPL DOUBLE STORE
       MOV  *R11+,@VARW  START ADDRESS OF THE FIELD
       MOV  *R11+,@ARG+2 UPPER LIMIT
***    DST  @VARW,@VARA
       MOV  @VARW,@VARA  NOTHING ENTERED YET
* VARA SHOULD POINT TO A SPACE LOCATION OR END OF FIELD

READL1
***    ST   1,@ARG+4     STORE BYTE=1 TO ARG+4
       MOVB @H01,@ARG+4  MEANS NO CHANGE IN LINE
READL2
***    DST  @VARW,@ARG+5 HAD TO USE ARG+6 BECAUSE OF WORD BOUNDARY PROBLEMS
       MOV  @VARW,@ARG+6 POSITION CURSOR AT START OF FIELD
READ00
***    CLR  @VAR1        CLEAR BYTE. I HAD TO USE WORD BECAUSE 9900 IS SO MUCH
***                      FASTER
       CLR  @VAR1    COUNTER FOR AUTO REPEAT
* This is where we return to exit INSERT mode.
READ01
***    CLR  @ARG+7       USED ARG+8 BECAUSE HAD TO USE ARG+6 & ARG+7 ALREADY
       MOVB @H00,@ARG+8   NORMAL OPERATION MODE
***    ST   CURSOR,@VARV
       MOVB @CURSOR,@VARV VARV USED FOR CURSOR/CHARACTER
READ$1
* Input 1 char and alternate cursor and character for blink
***    EX   @VARV,RAM(@ARG+5) EXCHANGE @VARG WITH WHATS AT LOCATION ARG+5 IN VDP
       MOV  @ARG+6,TEMP  EXCHANGE VARV,ARG+6
       BLWP @VSBR
       SWPB TEMP1
       MOVB @VARV,TEMP1
       BLWP @VSBW
       MOVB @R1LB,@VARV

***    CLR  @TIMER
       MOVB @H00,@TIMER  SET VDP TIMER TO ZERO

***    $REPEAT           MACRO. REPEAT CODE UNTIL $UNTIL IS TRUE
L00001 LIMI 2            ENABLE INTERRUPTS SO THE VDP TIMER (>8379) CAN INC
       LIMI 0            DISABLE INTERRUPTS SO THE VDP WON'T GET MESSED UP
***    SCAN              SCAN THE KEYBOARD
       BLWP @KSCAN       SCAN FOR A CHARACTER
***    BS   READ$2       BRANCH ON COND BIT (EQ) SET
       MOVB @STATUS,@STATUS   EQUAL BIT SET?
       JNE  READ$2       FOUND A NEW CHARACTER
***    INC  @VAR1        INCREMENT THE BYTE @VAR1 BY ONE
       INC  @VAR1    INC AUTO-REPEAT COUNTER
***    $IF @RKEY .NE. >FF THEN  MACRO. IF RKEY NOT EQ >FF THEN EXECUTE THE
***                      FOLLOWING CODE OTHERWISE SKIP TO THE $END IF TERMINATOR
       CB   @RKEY,@HFF   OLD KEY?
       JEQ  L00002       YEP
***       $IF  @VAR1 .HE. 254 THEN  HIGHER OR EQUAL
       C    @VAR1,@H508  HOLD OLD KEY FOR A WHILE
                         HAD TO DOUBLE 254 TO SLOW DOWN ASSEMBLY CODE
       JLT  L00002       BEFORE STARTING REPEAT
```



```
***    SUB  30,@VAR1     SUBTRACT BYTE
       S    @H60,@VAR1   CONTROL REPEAT RATE
***    B    READ$3       UNCONDITIONAL BRANCH
       JMP  READ$3
***       $END IF
***    $END IF
***    $UNTIL @TIMER .H. 14  TERMINATOR FOR REPEAT UNTIL HIGHER THAN 14
L00002 CB   @TIMER,@H14
       JLE  L00001       TIME NEXT CHARACTER SWITCH
***    BR   READ$1       BRANCH COND BIT RESET. USED TO SAVE ONE BYTE OF MEMORY
       JMP  READ$1       RESTART CHAR BLINK CYCLE
READ$2
***    CLR  @VAR1
       CLR  @VAR1   CLEAR AUTO REPEAT COUNTER
READ$3
***    $IF  @VARV .NE. CURSOR THEN
       CB   @VARV,@CURSOR IF NE EXCHANGE AGAIN
       JEQ  L00003

***    EX   @VARV,RAM(@ARG+5)
       MOV  @ARG+6,TEMP  EXCHANGE VARV,ARG+6
       BLWP @VSBR
       SWPB TEMP1
       MOVB @VARV,TEMP1
       BLWP @VSBW
       MOVB @R1LB,@VARV

***    $END IF
***    $IF @RKEY .L. : : THEN   IF RKEY LESS THAN SPACE THEN EXECUTE CODE

L00003 CB   @RKEY,@SPACE IF .LT. SPACE THEN CONTROL CHAR
       JLT  L00004
       B    @L0000C

* THIS IS WHERE YOU WOULD TRAP ALL CONTROL CODES *
*  HANDLE BREAK CHAR FIRST
*      CB   @RKEY,@BREAK
*      JNE  LABLE

* BACK ARROW - SPACE BACK ONE POSITION
***    $END IF
***    $IF @RKEY .EQ. BACK GOTO RBACK  GOTO's DO NOT REQUIRE AN END IF TERM
L00004 CB   @RKEY,@BACK  BACK ARROW?
       JNE  B00002       TO FIX OUT OF RANGE ERROR
       B    @RBACK

* RIGHT ARROW - FORWARD SPACE
***    $IF  @RKEY .EQ. FORW GOTO FORW
B00002 CB   @RKEY,@FORW
       JNE  B00003       TO FIX OUT OF RANGE ERROR
       B    @RFORW

* INSERT  *
***    $IF @RKEY .EQ. INSRT THEN
B00003 CB   @RKEY,@INSRT
       JNE  L00005
***    ST   1,@ARG+8
       MOVB @H01,@ARG+8  SET INSERT MODE FLAG
***    $END IF
```

```
* DELETE - DELETE THE CURRENT CHAR
***     $IF @RKEY .EQ. DLETE THEN
L00005 CB   @RKEY,@DLETE
       JNE  L00006
***    CLR  @ARG+4
       MOVB @H00,@ARG+4  INDICATE A CHANGE IN LINE
***    $IF @VARA .DNE. @ARG+6 THEN  THE D MEANS DOUBLE OR WORD OF MEMORY COMPARE
       C    @VARA,@ARG+6 EMPTY LINE?
       JEQ  L0001F       YEP.
***    DST  @VARA,@ARG
       MOV  @VARA,@ARG   MOVE EVERYTHING FROM THE RIGHT
***    DSUB @ARG+5,@ARG  DOUBLE BYTE (WORD) SUBTRACT
       S    @ARG+6,@ARG  OF THE CURSOR TO THE LEFT
***    MOVE @ARG FROM RAM(1(ARG+6)) TO RAM(@ARG+5) THIS IS A BLOCK MOVE OF @ARG
***    BYTES OF VDP RAM FROM WHATS AT ADDR ARG+6 PLUS 1 TO WHATS AT ADDRESS
***    ARG+6. IN SHORT MOVE EVERYTHING ON SCREEN ONE BYTE LOWER.
       MOV  @ARG,TEMP2   COUNTER
       MOV  @ARG+6,TEMP

       INC  TEMP         MOVE @ARG FROM RAM(1(ARG+6)) TO RAM(@ARG+6)
L00008 BLWP @VSBR
       DEC  TEMP
       BLWP @VSBW
       INCT TEMP
       DEC  TEMP2
       JNE  L00008

***    DDEC @VARA        DECREMENT THE WORD (DOUBLE) AT VARA
       DEC  @VARA        PRE-UPDATE END OF STRING

***    $IF RAM(@VARA) .EQ. : :+OFFSET GOTO READ01 OFFSET IS SCREEN OFFSET >60
       MOV  @VARA,TEMP
       BLWP @VSBR
       CB   @TEMP1,@SPACE
       JNE  B00001       TO RESOLVE OUT OF RG ERROR
       B    @READ01
***    DINC @VARA        INCREMENT THE WORD OF MEMORY AT VARA
B00001 INC  @VARA

***    ST   : :+OFFSET,RAM(@VARA)
L0001F MOV  @VARA,TEMP
       LI   TEMP1,>2000
       BLWP @VSBW
***    BR   READ01
       B    @READ01

* CLEAR - Clear the entire input line
***    $IF @RKEY .EQ. CLRLN THEN
L00006 CB   @RKEY,@CLRLN
       JNE  L00009
***    $REPEAT
***    ST   : :+OFFSET,RAM(@VARA)
       MOVB @SPACE,TEMP1
CLRLIN
       MOV  @VARA,TEMP   SO WE CAN FIDDLE WITH VALUE
       BLWP @VSBW
       DEC  @VARA        PRE-UPDATE END OF LINE
***    $UNTIL @VARA .DL. @VARW  DOUBLE LESS THAN
       C    @VARA,@VARW  UP TO AND INCL FIRST POS
       JHE  CLRLIN
```



```
***    DINC @VARA
       INC  @VARA        UNDO LAST SUBTRACTION
       CLR  @ARG+4
       MOVB @H00,@ARG+4  INDICATE CHANGE
***    BR   READL2
       B    @READL2      RESTART EVERYTHING
***    $END IF

* GENERAL EXIT POINT
***    $IF @RKEY .NE. CHRTN THEN
L00009 CB   @RKEY,@CHRTN ONLY REACT ON CR/UP/DOWN
       JEQ  L0000A
***    $IF @RKEY .NE. MVUP THEN
       CB   @RKEY,@MVUP
       JEQ  L0000A
***    $IF @RKEY .NE. DOWN GOTO READ$1
       CB   @RKEY,@DOWN
       JEQ  L0000A
       B    @READ$1
***    $END IF
***    $END IF
***    $IF @VARA .DEQ. @ARG+2 THEN  DOUBLE EQUAL
L0000A C    @VARA,@ARG+2 CHECK FOR BLOCK ON LAST POSITION
       JNE  L0000B
***    $IF RAM(@VARA) .NE. : :+OFFSET THEN
       MOV  @VARA,TEMP
       BLWP @VSBR
       CB   TEMP1,@SPACE  BLOCKED?
       JEQ  L0000B
***    DINC @VARA
       INC  @VARA        POINT BEYOND LAST CHAR IN LINE
***    $END IF
***    $END IF
L0000B RT                ENTER THE CURRENT LINE
***    $END IF (THIS IS FROM THE $IF THAT CHECKED FOR CTRL CODES)

* INSERT ROUTINE *
***    $IF @ARG+8 .NE. 0 THEN INSERT
L0000C CB   @ARG+8,@H00  INSERT MODE
       JEQ  L0000D
READ$4
***    DST  @VARA,@ARG
       MOV  @VARA,@ARG   USE ARG AS TEMP FOR INSERT
***    $WHILE @ARG .DH. @ARG+6
L0000F C    @ARG,@ARG+6  MOVE EVERYTHING UP TO CURSOR LOCATION
       JLE  L0000E
***    DDEC @ARG
       DEC  @ARG         COPY LOWER LOCATION TO HIGHER ONE
***    ST   RAM(@ARG),RAM(1(ARG)) GO FROM HIGH TO LOW IN VDP RAM
       MOV  @ARG,TEMP
       BLWP @VSBR
       INC  TEMP
       BLWP @VSBW
       JMP  L0000F
***    $END WHILE        TERMINATOR FOR WHILE
***    $IF @VARA .DL. @ARG+2 THEN
L0000E C    @VARA,@ARG+2 ONLY UPDATE VARA AS UPPER
       JHE  L0000D
***    DINC @VARA
       INC  @VARA        HASN'T BEEN REACHED YET
```

```
***     $END IF
***     $END IF
***     ST   @RKEY,RAM(@ARG+6)
L0000D MOVB @RKEY,TEMP1  DISPLAY THE CHARACTER
       MOV  @ARG+6,TEMP
       BLWP @VSBW
***     CLR  @ARG+4
       MOVB @H00,@ARG+4  INDICATE CHANGE IN LINE
READ05
***     $IF @ARG+5 .DEQ. @ARG+2 THEN
       C    @ARG+6,@ARG+2 HIT RIGHT MARGIN?
       JNE  L0002F
***     CALL TONE2        CALL ANOTHER GPL ROUTINE IN THIS CASE BONK
       MOVB @H00,@STATUS CLEAR THE STATUS BYTE BEFORE ACCESSING GPL
       BLWP @GPLLNK      GIVE A BAD RESPONSE TONE
       DATA >0036
***     BR   READ$1
       B    @READ$1      STAY IN CURRENT MODE
***     $END IF
***     DINC @ARG+6
L0002F INC  @ARG+6       UPDATE CURRENT ADDRESS
***     IF @ARG+6 .DH. @VARA THEN
       C    @ARG+6,@VARA CHECK FOR LAST NEW HIGH LIMIT
       JLE  L00010
***     DST  @ARG+5,@VARA
       MOV  @ARG+6,@VARA UPDATE NEW HIGH LIMIT
***     $END IF
***     $IF @VARA .DL. >2FE GOTO READ$1
L00010 C    @VARA,@H766
       JHE  L00011       TO FIX OUT OF RANGE PROBLEM
       B    @READ$1      STILL SOME SPACE TO GO
L00011
* This is where we could scroll the screen if needed
* UPDATE POINTERS IF YOU SCROLL *
***     CALL SCROLL       SCROLL THE SCREEN
***     DSUB 28,@VARA
*       S    @H28,@VARA   BACK TO START OF LINE
***     DSUB 32,@VARW
*       S    @H32,@VARW   BACKUP START LINE ADDRESS
***     DSUB 32,@ARG+2
*       S    @H32,@ARG+2  ABSOLUTE HIGH LIMIT BACKS UP TOO
***     DSUB 32,@ARG+6
*       S    @H32,@ARG+6  CURRENT CURSOR POSITION ALSO
***     BR   READ$1
       B    @READ$1      START WITH SOMETHING ELSE

* FORWARD CURSOR MOVE
RFORW
***     CLR  @ARG+8
       MOVB @H00,@ARG+8  LEAVE INSERT MODE
***     BR   READ05
       B    @READ05      USE REST OF LOGIC

* BACK CURSOR MOVE
RBACK
***     $IF @ARG+5 .DH. @VARW
       C    @ARG+6,@VARW CHECK BOTTOM RANGE
       JLE  L00012
***     DDEC @ARG+6
       DEC  @ARG+6
```



```
***     $END IF
***     BR   READ01
L00012 B    @READ01

       END

*************************************
* THIS IS A ROUTINE TO DIRECTLY ACCESS *
* THE GROM READLN ROUTINE. USE CALL    *
* LOAD("DSK1.FILENAME") AND CALL LINK  *
* ("DSK1.START") FROM E/A BASIC TO SEE *
* IT BECAUSE OF SCREEN OFFSET.         *
*************************************

       DEF  START

GPLWS  EQU  >83E0         ADDRESS FOR GPL WORK SPACE

H00    BYTE 0
WS     BSS  >20           MY WORKSPACE
       EVEN

START
       LWPI WS            POINT TO MY WORKSPACE

       LI   R0,>2
       MOV  R0,@>8320     START SCREEN ADDRESS FOR SCAN

       MOVB @H00,@>837C  CLEAR THE STATUS BYTE SO WE DON'T GET AN ERROR
       BLWP @GPLLNK       LINK TO THE ROUTINE IN GROM
       DATA >2A42

       MOVB @H00,@>837C  RETURN TOTHE CALLING PROGRAM ON ENTER
       LWPI GPLWS
       B    @>0070

* YOU COULD HAVE PLACED AN END STATEMENT HERE AND REF'd GPLLNK INSTEAD
* OF USING THIS ROUTINE.
* GPLLNK ROUTINE *

GPLLNK DATA UTILWS,XGPL  VECTOR FOR THE GPLLNK BLWP

UTILWS EQU  >2094
SUBSTK EQU  >8373
FLAG2  EQU  >8349
SVGPRT EQU  >2030

H20    BYTE >20
EVEN

XGPL
     MOVB @SUBSTK,R1
     SRL  R1,8
     MOV  *R14+,@>8304(R1)
     SOCB @H20,@FLAG2
     LWPI GPLWS
     MOV  @SVGPRT,R11
     RT

       END
```

As a result of having far too much to print at the time, we have missed out on the earliest days of the Tigercub, and now we have the greatest pleasure in sharing with you the very first... ?

TIPS FROM THE TIGERCUB

NO #1

Tigercub Software
156 Collingwood Ave.
Columbus, OH 43213

Here's a tip for beginners- Don't use EDIT!! There are two ways to bring a program line to the screen. You may type EDIT, the line number, and press ENTER...or you may just type the line number & press FCTN together with the Down-arrow or Up-arrow key. But when you graduate to Extended Basic,you will find that you can only use the second method. Then, while you're trying to break the EDIT habit, you are apt to get confused, type the line number, hit ENTER - delete the program line entirely!!!

****

Are you tired of that Blankety Blinking Cursor? This wont work in Basic but if you're in XB try 1 CALL COLOR(0,11,1)

****

Have you ever spent an hour looking for a bug, and finally found that you had typed an 0 for an o, or vice versa? I'll never understand why Texas Instruments didn't slash the 0. You can easily do it with this line, 1 CALL CHAR(48,"003A444E546444B8"). Trouble is, any redefined character with an ASCII code below 128 will only be re-defined while the program is running, so your 0's will still be unslashed while you are keying in a program or listing it. However, you can add a temporary line 2 GOTO 2, then key in or list a screenfull of program lines, type RUN, and watch to be sure that all your 0's become slashed and your 0's do not.

Have you ever been typing in a program, and the console suddenly jumped back to the title screen,and you were sure that you didn't have a finger any-where near that infernal QUIT key? But maybe you were drinking coffee with one hand and trying to press FCTN and 1 simultaneously with the other? So, if you don't have anything valuable in the computer right now, try pressing FCTN, space bar, H and N all at the same time. Oops!

Another useless bit of info- try FCTN 5,6, and 7 all together. Break!!

****

```
100 CALL CLEAR :: PRINT "TIG
ERCUB CHARACTER ENLARGER":@
$:@$:"by Jim Peterson":@$:@$
:@$:@$:"SELECT SIZE 1, 2 OR
3":@$:@$
110 CALL KEY(0,K,ST):: IF (S
T=0)+(K<49)+(K>51)THEN 110 :
: S=K-48
120 DIM HX$(96),c$(16),M$(16
)
130 DATA 0000,0001,0010,0011
,0100,0101,0110,0111,1000,10
01,1010,1011,1100,1101,1110,
1111
140 CALL CHAR(33,"FFFFFFFFFF
FFFFFF"):: FOR J=0 TO 15 ::
READ C$(J):: NEXT J
150 CALL SOUND(100,800,0):: 
PRINT "READY - TYPE CHARACTE
R"
160 CALL KEY(0,K,ST):: IF (S
T=0)+(K<32)+(K>127)THEN 160
```



```
:: CALL CLEAR :: PRINT "WAIT
, PLEASE..." :: CALL CHARPAT
(K,HX$(K-31))
170 FOR J=1 TO LEN(HX$(K-31)
):: A$=SEG$(HX$(K-31),J,1)::
 IF ASC(A$)>57 THEN 180 :: B
=ASC(A$)-48 :: GOTO 190
180 B=ASC(A$)-55
190 FOR L=1 TO 4 :: X=VAL(SE
G$(C$(B),L,1)):: FOR M=1 TO
S :: M$(J)=M$(J)&CHR$(32+ABS
(X>0)):: NEXT M :: NEXT L ::
 NEXT J
200 CALL CLEAR :: FOR J=1 TO
 16 STEP 2 :: FOR N=1 TO S :
: PRINT TAB(11-S^2);M$(J);M$
(J+1):: NEXT N :: NEXT J
210 PRINT @$:@$ :: FOR J=1 T
O 16 :: M$(J)=NUL$ :: NEXT J
 :: GOTO 150
```

TIPS FROM THE TIGERCUB

NO #2

****

Tnis program will play and print the frequency codes of the two "secret" sub-octaves of bass notes on the TI-99/4A. - Jim Peterson, Tigercub Software

```
110 DEF R(X)=INT(X+.5)
120 F=1652
130 FOR J=1 TO 25
140 READ N$
150 PRINT N$;" =";R(F)
160 CALL SOUND(500,30000,30,
30000,30,F,30,-4,0)
170 F=F/1.059463094
180 IF J<>12 THEN 200
190 RESTORE
200 NEXT J
210 DATA A,A FLAT,G,F#,F,E,E
 FLAT,D,C#,C,B,B FLAT,A
```

To play these bass notes,the CALL SOUND must contain 3 tones and a noise. The first two tones may be either an audible or in- audible frequency and volume. The third must be the frequency code (which isn't the actual frequency) for the note,with an inaudible vol- ume; and the noise must be -4,with an audible volume.

This program plays tremolo notes. Change the value in line 150 to 1.01 or 1.03 for less or more tremolo. The tune is "St. James Infirm- ary Blues" - Jim Peterson, Tigercub

```
110 FOR J=1 TO 60 STEP 2
120 READ A,B
130 FOR L=1 TO A
140 CALL SOUND(-1000,B,0)
150 CALL SOUND(-1000,B*1.02,
0)
160 NEXT L
170 NEXT J
180 CALL SOUND(-1,30000,30)
190 DATA 2,330,2,294,4,330,4
,294,4,330,4,294,4,262,8,220
200 DATA 2,330,2,294,6,330,2
,294,4,330,4,262,12,247
210 DATA 2,294,2,262,4,294,4
,262,4,294,2,330,2,294,4,262
,8,220
220 DATA 4,262,4,262,4,220,4
,262,4,247,16,220
```

****

```
100 REM - WHAT HAPPENED? by
Jim Peterson, Tigercub Soft-
ware
110 CALL CLEAR :: FOR K=65 T
O 90 :: CALL CHARPAT(K,A$)::
 FOR J=15 TO 1 STEP -2 :: CH
$=CH$&SEG$(A$,J,2):: NEXT J
:: CALL CHAR(K,CH$)
120 CH$=NUL$ :: NEXT K :: DI
SPLAY AT(14,3):"VT EHT DENRU
T OHW !YEH" :: DISPLAY AT(12
,13):"?NWOD EDISPU"
130 INPUT Q$ :: GOTO 130
```

TIPS FROM THE TIGERCUB

NO #3

I'm just a one-man User's Group pretending to be a business - not a business pretending to be a User's Group!

Here's a lifesaver that was passed on to me verbally - I don't know who to credit for

discovering it... It's 2 A.M., You just got you got the last bug out of your new program, you sleepily put a new tape in the recorder, type OLD CS1, hit ENTER and ...ooooh!, you meant to type SAVE CS1!! But all is not lost - just type Shift E,hit ENTER, get an IO error message, and start over.

```
100 CALL CLEAR
110 RANDOMIZE
120 DATA TIGERCUB SOFTWARE,P
RESENTS,THE,CHAMELEON,SCREEN
BORDER,AND,WIPE, by Jim Pete
rson," ","          TOUCH ANY
KEY"
130 REM - M$ COMPOSED OF PAI
RS OF HEX CODES WHICH ARE MI
RROR IMAGES OF EACH OTHER
140 M$="1800665AC342DB667E18
8100995AC3A5E78142BD24DB6600
81429924007E5AC3A53C241800FF
DB5AFF7EFF0099188100660018"
150 RESTORE 120
160 REM - PRINTS TEXT FOR DE
MONSTRATION
170 FOR P=1 TO 10
180 READ A$
190 REM - TAB TO CENTER TEXT
200 PRINT TAB(15-LEN(A$)/2);
A$;" "
210 NEXT P
220 GOSUB 300
230 REM - PAUSE, WAIT FOR AN
Y KEY
240 CALL KEY(0,K,ST)
250 IF ST=0 THEN 240
260 GOSUB 440
270 GOTO 150
280 REM - SUBROUTINE TO PICK
PATTERN AND COLORS, DRAW BOR
DER
290 REM - RANDOMLY SELECT AN
Y STRING OF 8 SYMMETRICAL PA
IRS FOR HEX CODE, DEFINE CHA
RACTER
300 CALL CHAR(128,SEG$(M$,IN
T(43*RND+1)*2-1,16))
310 REM - RANDOMLY SELECT FO
REGROUND AND BACKGROUND COLO
RS BETWEEN 3 AND 16
320 X=INT(14*RND+3)
330 Y=INT(14*RND+3)
340 REM - IF BACKGROUND IS S
AME COLOR AS FOREGROUND, PIC
K ANOTHER
350 IF Y=X THEN 330
360 REM - DRAW THE BORDER
370 CALL COLOR(13,X,Y)
380 CALL HCHAR(1,2,128,31)
390 CALL HCHAR(24,2,128,31)
400 CALL VCHAR(1,2,128,24)
410 CALL VCHAR(1,31,128,24)
420 RETURN
430 REM - SUBROUTINE FOR ALT
ERNATING WIPES. VALUE OF T A
LTERNATES BETWEEN 1 AND 2
440 T=T+1-ABS(T=2)*2
450 ON T GOTO 470,510
460 REM - LEFT TO RIGHT WIPE
470 CALL VCHAR(1,3,128,768)
480 CALL CLEAR
490 GOTO 530
500 REM - TOP TO BOTTOM WIPE
510 CALL HCHAR(1,1,128,768)
520 CALL CLEAR
530 RETURN
```

Of course, that routine can be varied in many ways. For example, try changing line 300 CALL CHAR(1280"FF"&SEG$(M$,INT(43*RND+1)*2-1,12)&"FF"). The basic algorithm of line 140(which can be any combination of the mirror-image pairs)and lines 300-350,has endless uses for putting colorful graphics on your screen. For instance -

```
100 REM - TIGERCUB RANDOM BA
RS, by Jim Peterson
110 CALL CLEAR
120 RANDOMIZE
130 M$="0018243C425A667E8199
00A5BDC3DBE7FFFFE7DBC3BDA500
817E665A423C24180018243C425A
667E8199A5BDC3DBE7FFFFE7DB"
140 FOR CH=40 TO 142 STEP 8
150 CALL CHAR(CH,SEG$(M$,INT
(43*RND+1)*2-1,16))
160 X=INT(14*RND+3)
170 Y=INT(14*RND+3)
180 IF Y=X THEN 170
190 CALL COLOR(CH/8-3,X,Y)
200 CALL HCHAR(23*RND+1,31*R
ND+1,CH,10*RND+1)
210 CALL VCHAR(23*RND+1,31*R
ND+1,CH,10*RND+1)
220 Z=INT(10*RND)
230 IF Z<>0 THEN 250
240 CALL CLEAR
250 IF Z <>1 THEN 270
260 CALL SCREEN(INT(15*RND+2
))
270 NEXT CH
280 GOTO 140
```

Hey! I just thought of something else to try that Chameleon Screen Border and Wipe. Try changing - 250 IF ST=0 THEN 220

The previous routine generated random redefined characters which have 2-way symmetry, left and right.

The following routine generates characters which have 4-way symmetry and are even more interesting, although the routine is a bit slower.

```
100 CALL CLEAR
110 RANDOMIZE
120 DIM A$(16)
130 DATA 00,18,24,3C,42,5A,6
6,7E,81,99,A5,BD,C3,DB,E7,FF
140 FOR J=1 TO 16
150 READ A$(J)
160 NEXT J
170 FOR CH=40 TO 152 STEP 8
```

→ ->continued on next page

There is some evidence to suggest these early TIPS have been rekeyed from printed material. Some minor keying errors have been corrected, we apologise for any we missed!

==========

25 REASON WHY YOUR TI99/4A IS BETTER THAN WOMEN

compiled by Garry Christensen Brisbane User Group Australia

1. You don't have to wine and dine a 4A.
2. Your 4A doesn't mind if you play around with other computers.
3. Your 4A doesn't get headaches.
4. Your 4A doesn't demand equality.
5. If you change to another computer, you don't have to pay alimoney.
6. Your 4A comes with instructions.
7. Your 4A always does what you tell it.
8. Your 4A never goes shopping with the credit cards.
9. If your 4A is on the phone to long, you can simply hang up.
10. Your 4A talks less.
11. Your 4A doesn't complain when your hands are cold.
12. Your 4A doesn't mind what time you get home.
13. Your 4A doesn't care how long the lawn is.
14. Your 4A is ready to go 24 hours a day.
15. Your 4A doesn't want you to re-decorate every couple of years.
16. It's easy to upgrade your 4A.
17. Your 4A is easy to turn on.
18. Your 4A never needs a 'screen-lift'.
19. Your 4A has a better memory.
20. Your 4A doesn't care if you respect it or not.
21. Your 4A looks just as good first thing in the morning.
22. You can't catch a virus off your 4A.
23. Your 4A doesn't invite its friends around for Tupperware/Naughty Nickers parties.
24. Your 4A doesn't take ages to make itself look presentable.
25. You can't offend your 4A with sexist jokes.

Please note #25 above! Any ladies reading this are invited to consider replacing "wife" with "husband"- or perhaps submitting their own lists (The TI99/4A never has a hang over...).

As a Home Computer the TI99/4A is an ideal partner in a menage a trois of course...

```
180 FOR L=1 TO 4
190 X=INT(16*RND+1)
200 B$=B$&A$(X)
210 C$=A$(X)&C$
220 NEXT L
230 CALL CHAR(CH,B$&C$)
240 B$=NUL$
250 C$=NUL$
260 NEXT CH
270 FOR S=2 TO 16
280 Y=INT(15*RND+2)
290 Z=INT(15*RND+2)
300 IF Z=Y THEN 290
310 CALL COLOR(S,Y,Z)
320 NEXT S
```

That's the routine. Now, to try it out...

```
330 T=T+1
340 IF T>1 THEN 170
350 CH=40
360 TX=0
370 FOR X=1 TO 12
380 CALL HCHAR(X,1+X,CH,29-X
-TX)
390 CALL HCHAR(25-X,1+X,CH,2
9-X-TX)
400 CALL VCHAR(X,1+X,CH,25-X
-TX)
410 CALL VCHAR(X,31-X,CH,25-
X-TX)
420 CH=CH+8
430 TX=TX+1
440 NEXT X
450 GOTO 170
```

For a different effect,try changing...

```
180 FOR L=1 TO 3
190 X=INT(8*RND+1)
230 CALL CHAR(CH,"00"&B$&C$&
"00")
```

****

A tip for beginning programmers: Don't use character sets 15 and 16 (ASCII codes 144-159) unless you really need to. Then, when you get to Extended Basic your program will run without modification in Extended Basic, and usually faster and better.

OUT OF MEMORY...so that's all for now. You won't hurt my feelings if you mention Tigercub Software to your friends. I have some bargain programs they might like. Just tell them I'd like a dollar for my catalog to cover bankruptcy court fees.

TIPS FROM THE TIGERCUB
#25
Copyright 1985

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

Here's a bit of psychedelic blues - -

```
100 REM - FRANKIE & JOHNNIE
    by Jim Peterson
110 DIM S(12)
120 CALL SCREEN(2)
130 FOR R=1 TO 12
140 CALL COLOR(R+1,1,1)
150 FOR T=R TO 25-R
160 CALL HCHAR(T,R,32+R*8,34
-2*R)
170 NEXT T
180 NEXT R
190 DATA 262,294,311,330,349
,392,440,494,523,587,40000
200 FOR N=1 TO 11
210 READ S(N)
220 NEXT N
230 FOR J=1 TO 110 STEP 2
240 CALL COLOR(A+1,1,1)
250 READ T,A
260 CALL COLOR(A+1,A+2,A+2)
270 FOR TT=1 TO T
280 CALL SOUND(-999,S(A),0)
290 NEXT TT
300 NEXT J
310 RESTORE 330
320 GOTO 230
330 DATA 2,1,2,2,2,4,2,7,1,1
1,1,7,2,6,4,4,2,1,1,11,13,1
340 DATA 2,1,2,2,2,4,2,7,1,1
1,1,7,2,6,4,4,12,1
350 DATA 1,11,3,1,2,5,2,6,2,
7,2,9,1,11,1,9,2,10,4,7,1,9,
1,11,7,9
360 DATA 4,7,2,8,2,9,1,11,3,
9,1,11,1,9,4,8,2,7,6,6
370 DATA 4,4,1,11,3,4,4,3,16
,2,1,11,4,7,2,6,4,7,4,6,20,1
,8,11
```

You can too have a blank space in your disk filenames! Just use FCTN V for the blank, instead of the space bar. You can even have a diskfull of 10 programs with invisible filenames consisting of 1 to 10 of those FCTN V's.

However, those invisible characters can do strange things when you list your disk catalog to a printer. (and they are not recognized by the Disk Manager. - Ed.)

If you want to INPUT a string with leading and/or trailing blanks, just enclose the whole works in quotation marks. Try this -

```
100 INPUT A$ !type TEST
110 PRINT A$;LEN(A$)
120 INPUT A$ !type " TEST "
130 PRINT A$;LEN(A$)
140 GOTO 100 !you can even
input a blank string of 136
characters.
```

I really shouldn't tell you this, but if you want to make it difficult for someone to LIST your program, just insert a garbage line, every 5th line or so until you run out of memory, consisting of REM followed by 4 or 5 lines of random characters typed with the CTRL key held down.

Here's a program that can actually read your mind!

```
100 CALL CLEAR
110 PRINT "TIGERCUB MIND REA
DER PROGRAM": :
120 PRINT "I'll bet you a do
llar I can guess what you ar
e thinking.": :
130 GOSUB 440
140 PRINT "And I'll bet  ano
ther dollar I can tell if wh
at you are  thinking is cor
rect.": :
150 GOSUB 440
160 PRINT "And I'll bet anot
her dollar I'm right BOTH ti
mes.": :
170 GOSUB 440
180 PRINT "And I'll bet one
more dollar I can guess what
 you'll be thinking a minute
 from  now.": :
190 GOSUB 440
200 PRINT "OK....": :
210 GOSUB 480
220 PRINT "You're thinking t
hat a compu-ter can't possib
ly know   what you are thin
king.............right?": :
230 GOSUB 480
240 PRINT "So I told you wha
t you were":"thinking.......
.right?": :
250 GOSUB 480
260 PRINT "You owe me a buck
.": :
270 GOSUB 480
280 PRINT "And you're absolu
tely      right..I can't re
ad your   mind.": :
290 GOSUB 480
300 PRINT "So I told you cor
rectly that":"what you were
thinking was":"correct......
right?": :
310 GOSUB 480
320 PRINT "You owe me anothe
r buck.": :
330 GOSUB 480
340 PRINT "So I was right BO
TH times...right?": :
350 GOSUB 480
360 PRINT "That makes three
bucks you  owe me.": :
370 GOSUB 480
380 PRINT "And now it's a mi
nute later":"and you're thin
king you've":"been played fo
r a sucker....":"...right?":
 :
```

```
390 GOSUB 480
400 PRINT "...so you owe me
four bucks.": :
410 GOSUB 480
420 PRINT "NEVER NEVER bet a
gainst a  computer!! "
430 END
440 PRINT "Want to bet? Type
 Y(yes)": :
450 CALL KEY(3,K,ST)
460 IF (ST=0)+(K<>89)THEN 45
0
470 RETURN
480 FOR D=1 TO 800
490 NEXT D
500 RETURN
```

Since the manual doesn't mention it, some folks don't know that you can use IMAGE and PRINT USING for output to the printer. Try this -

```
100 OPEN #1:"PIO"
110 INPUT "NAME? ":N$
120 INPUT "AMOUNT? ":A
130 PRINT #1,USING "########
##########              ####
.##":N$,A
140 GOTO 110
```

Of course, you could also add a line:-

```
105 IMAGE "################
          ####.##"
```

AND change line 130 to:

```
130 PRINT #1,USING 105:N$,A
```

XB: If you have a program on disk which is so long that you must type CALL FILES(1) before you can load it, AFTER you load it, type CALL FILES(3) and SAVE it back to disk.
It will now be INT/VAR 254 format and will load without CALL FILES(1). If you then need sometime to make a casette copy, just load it in, type CALL FILES(1)and SAVE it back to disk again.
(Extended Basic only this!)

#26

Copyright 1985

Barry Ensley warns that when FCTN V is used for a blank in a filename, as mentioned in Tips #25, it is not recognized by the Disk Manager.

And the last word - I think - on the challenge to quickly scramble the numbers 1 to 255. Ian Swales sent me, from Belgium, two routines which beat everyone else - and then sent me two more which beat his first ones! His PEEK version -

```
100 DIM A(255),C(255):: FOR
K=255 TO 1 STEP -1 :: RANDOM
IZE :: CALL PEEK(-31808,B)::
 J=INT(B*K/256+1):: C(K)=MAX
(J,A(J)):: A(J)=MAX(K,A(K)):
: NEXT K
```

And see if you can unravel the logic of this truly elegant bit of code!

```
100 DIM A(255):: RANDOMIZE :
: FOR K=255 TO 1 STEP -1 ::
J=INT(RND*K+1):: T=MAX(J,A(J
)):: A(J)=MAX(K,A(K)):: A(K)
=T :: NEXT K
```

To give you an idea of Barry Traver's knowledge of our computer, try this one. I've figured out the why, but I'll have to ask Barry to explain the why of the why!

```
100 ! LINPUT PUZZLE/BUG by
      B.A. Traver
110 ! QUESTIONS? Send SASE
      to Barry Traver
120 ! 552 Seville St.
      Phila. PA 19128
130 CALL CLEAR :: PRINT "LIN
PUT PUZZLE/BUG":"BY BARRY TR
AVER"
140 PRINT "Can you figure ou
t why your computer will not
 obey?"
150 PRINT "Why won't it stop
 when you tell it to?": : :
160 LINPUT "Want me to stop?
 (YES/NO)":W$
170 IF W$="YES" THEN STOP EL
SE 160
180 END
```

I've said it before, there is more than one way to skin that poor cat. This is my routine to alternate between the #1 and joysticks.

```
Z=Z+1+(Z=2)*2 :: CALL (JOYST
(Z,X,Y)
```

Compact, isn't it? Now, the Reading-Berks 99ers publish a newsletter called "A Byte of Info", which is hardly more than a byte long, but the August byte was a mouthful! Check this -

```
100 Z=2
110 Z=1/Z*2 :: CALL JOYST(Z,
X,Y)
```

And this! Elegant!

```
Z=Z=0 :: CALL JOYST(Z+2,X,Y)
```

Here is another of those programs that write a program. This one will read a screen of graphics and/or text and convert it into a RUNable program of DISPLAY AT statements which will recreate the screen.

First, we need a file of the hex codes of all the normal characters, to check against to see if any have been redefined. Rather than key in all 95 of the 16-digit codes, let's write a program to write a program of them -

```
110 OPEN #1:"DSK1.HEXCODES",
VARIABLE 163 :: LN=30000 ::
FOR D=32 TO 124 STEP 8 :: FO
R CH=D TO D+7 :: CALL CHARPA
T(CH,CH$)
120 D$=D$&CHR$(179)&CHR$(200
)&CHR$(16)&CH$ :: NEXT CH
130 PRINT #1:CHR$(INT(LN/256
))&CHR$(LN-256*INT(LN/256))&
CHR$(147)&SEG$(D$,2,LEN(D$))
&CHR$(0):: LN=LN+1 :: D$=""
:: NEXT D
140 PRINT #1:CHR$(255)&CHR$(
255):: CLOSE #1 :: END
```

RUN that to create a MERGE format program of DATA statements. Now, key in the GRAFWRITER program -

```
31000 SUB GRAFWRITER
31001 OPEN #1:"DSK1.PG",OUTP
UT,DISPLAY ,VARIABLE 163
31002 RESTORE 30000 :: L=300
00 :: GOSUB 31018
31003 FOR CH=32 TO 127 :: CA
LL CHARPAT(CH,CH$):: RECD A$
 :: IF CH$=A$ THEN 31004 ELS
E GOSUB 31019 :: GOSUB 31018
31004 NEXT CH
31005 FOR CH=128 TO 143 :: C
ALL CHARPAT(CH,CH$):: IF CH$
=RPT$("0",16)THEN 31006 ELSE
 GOSUB 31019 :: GOSUB 31018
31006 NEXT CH
31007 PRINT #1:L$&CHR$(157)&
CHR$(200)&CHR$(5)&"CLEAR"&CH
R$(0):: GOSUB 31018
31008 FOR R=1 TO 24
31009 M$=L$&CHR$(162)&CHR$(2
40)&CHR$(183)&CHR$(200)&CHR$
(LEN(STR$(R)))&STR$(R)&CHR$(
179)
31010 FOR C=3 TO 30 :: CALL
GCHAR(R,C,G):: CALL HCHAR(R,
C,42):: IF F=0 AND G=32 THEN
 31013
31011 F=1 :: IF FF=1 THEN 31
012 ELSE CC=C-2 :: FF=1
31012 A$=A$&CHR$(G)
31013 NEXT C :: IF CC=0 THEN
 CC=1 :: A$=" "
31014 PRINT #1:M$&CHR$(200)&
CHR$(LEN(STR$(CC)))&STR$(CC)
&CHR$(182)&CHR$(181)&CHR$(19
9)&CHR$(LEN(A$))&A$&CHR$(0)
31015 L=L+10 :: F,FF,CC=0 ::
 M$,A$="" :: GOSUB 31018 ::
NEXT R
31016 PRINT #1:L$&CHR$(134)&
CHR$(201)&L$&CHR$(0):: GOSUB
 31018
31017 PRINT #1:CHR$(255)&CHR
$(255):: CLOSE #1 :: SUBEXIT
31018 L1=INT(L/256):: L2=L-2
56*L1 :: L$=CHR$(L1)&CHR$(L2
):: L=L+10 :: RETURN
31019 PRINT #1:L$&CHR$(157)&
CHR$(200)&CHR$(4)&"CHAR"&CHR
$(183)&CHR$(200)&CHR$(LEN(ST
R$(CH)))&STR$(CH)&CHR$(179)&
CHR$(199)&CHR$(16)&CH$&CHR$(
182)&CHR$(0):: RETURN
31020 SUBEND
```

Next, Enter MERGE DSK1.HEXCODES to merge in those DATA statements. Then save the program by SAVE DSK1.GRAFWRITER,MERGE

Now, load any program which has a screen you would like to copy. Run the program to the point where the screen display is ready, then break it with FCTN 4.

Put in a temporary line going to itself, such as 1001 GOTO 1001, and run the program again to be sure you found the right place. Then replace that temporary line with CALL GRAFWRITER :: STOP

Put in the disk containing the Grafwriter program and enter MERGE DSK1.GRAFWRITER. Then RUN the program. When it stops, type NEW, then MERGE DSK1.PG and then RUN!

Now for a Tigercub challenge uoat I can't answer! Can one of you assembly programmers tell me how to PEEK out of Extended Basic for screen color and character set colors, so I can reproduce them in that program?

And, thanks to Jerry Glaze in the Southern Nevada UG newsletter, by way of the Tidewater newsletter - you don't need SIZE with DISPLAY AT - just a semicolon!

```
100 DISPLAY AT(12,1):RPT$("*
",28):: DISPLAY AT(12,1):"SE
E?";
```

MEMORY FULL! - Jim Peterson

On Saturday May 16th, the LIMA User Group held a multi-user group conference, attended by many famous names in the TI world, including Dolores Werthes who joined us in Romiley a few years back.

Although a small user group, they supply -free of charge- a good venue for a get together, and organise a whole series of presentations on a wide series of topics.

These presentations are videotaped for the benefit of those unable to attend - obviously these are amateur recordings, and equally obviously they are to NTSC standards - you can see professional NTSC by watching the colours splurge on the BBC transmissions of Star Trek-The Next Generation!

This year I have requested copies of the tapes, and once received will be having them transferred to UK PAL standards.

The quality is not expected to be brilliant - of the sound and picture that is- but this is a great way to keep up. Harrison Software and OPA of Canada are expected to be giving presentations amongst others.

Details of what is on each tape may be available by the time this reaches you, or very shortly thereafter.

If you are interested in purchasing PAL copies of the tapes (for about 14 pounds each) or in hiring them (could be a long waiting list! for say four pounds per tape per week including outward post and packing) please drop me a line as soon as possible.

Stephen Shaw 10 Alstone Road STOCKPORT Cheshire SK4 5AH

* * * * * * * * * * * * * * * * *

A SMALL GRAPHIC PROGRAM. For Ex Bas + The Missing Link but any language which can set pixels on will do:

```
100 FOR T=0 TO 2000 STEP 0.006
110 X=SIN(0.99*T)-0.7*COS(3.01*T)
120 Y=COS(1.01*T)+0.1*SIN(15.03*T)
130 ! or try other fractional multipliers
140 ! this formula gives X from -1.7 to +1.7
     and Y from -1.1 to +1.1 so...
150 x=x*54+96 :: y=y*100+120
160 CALL LINK("PIXEL",X,Y)
170 NEXT T
180 END
```

It takes a while but the result is quite attractive. You can take even longer by concentrating on a part of the image (different scaling) to see the finer details.

TIPS FROM THE TIGERCUB

No. 65

Tigercub Software
156 Collingwood Ave.
Columbus, OH 43213

****

My three Nuts & Bolts disks, each containing 100 or more subprograms, have been reduced to $5.00. I am out of printed documentation so it will be supplied on on disk.

My TI-PD library now has well over 500 disks of fairware (by author's permission only) and public domain, all arranged by"category and as full as possible, provided with loaders by full program name rather than filename, Basic programs converted to XBasic, etc. The price is just $1.50 per disk(!), post paid if at least eight are ordered. TI-PD catalog #5 and the latest supplement is available for $1 which is deductible from the first order.

It is a bit of a nuisance to have to hit Enter after inputting a single character such as Y or N for "yes" or "no". CALL KEY accepts a single character without Enter, but has no blinking cursor to tell you that it is waiting. I should have had this one in my Nuts & Bolts years ago - the CALL KEY WITH CURSOR subprogram! R is the row, C is the TAB position, V$ is the validation string, such as "YyNn", and the character selected is returned in K$.

```
30000 SUB CALLKEY(R,C,V$,K$)
30001 CALL HCHAR(R,C+2,30)::
 FOR T=1 TO 3 :: CALL KEY(0,
K,S):: IF S<>0 THEN 30004
30002 NEXT T :: CALL HCHAR(R
,C+2,20):: FOR T=1 TO 3 :: C
ALL KEY(0,K,S):: IF S<>0 THE
N 30004
30003 NEXT T :: GOTO 30001
30004 IF POS(V$,CHR$(K),1)=0
 THEN 30001 ELSE K$=CHR$(K)
30005 SUBEND
```

And for a demonstration of the use of that subprogram, here is a little game that no one will ever play to the end -

```
100 DISPLAY AT(3,6)ERASE ALL
:"THE ULTIMATE TEST":"":" An
swer the question with a num
ber according to whether the
 number or color shown,"
110 DISPLAY AT(8,1):"or the
note sounded, was 1stor 2nd
or 3rd, etc."
120 DISPLAY AT(23,6):"PRESS
ANY KEY" :: DISPLAY AT(23,6)
:"press any key" :: CALL KEY
(0,K,SS):: IF SS=0 THEN 120
ELSE CALL CLEAR
130 DATA 2,BLACK,3,GREEN,5,B
LUE,9,RED,12,YELLOW,14,PURPL
E
140 FOR J=1 TO 6 :: READ C(J
),C$(J):: CT$=CT$&CHR$(J)::
W$=W$&CHR$(J+48):: NEXT J ::
 T=2 :: DL=500 :: V$="12"
150 RANDOMIZE :: T$,NN$=CT$
:: FOR J=1 TO T :: X=INT(RND
*LEN(T$)+1):: X$=SEG$(T$,X,1
):: T$=SEG$(T$,1,X-1)&SEG$(T
$,X+1,255):: Y(J)=ASC(X$)
160 X=INT(RND*LEN(NN$)+1)::
X$=SEG$(NN$,X,1):: NN$=SEG$(
NN$,1,X-1)&SEG$(NN$,X+1,255)
:: S(J)=ASC(X$):: NEXT J ::
FOR J=1 TO T
170 Z(J)=INT(89*RND+10):: FO
R K=1 TO J-1 :: IF Z(J)=Z(K)
THEN 170
180 NEXT K :: NEXT J :: CALL
 CLEAR :: CALL COLOR(3,16,1,
4,16,1)
190 FOR J=1 TO T :: CALL SCR
EEN(C(Y(J))):: CALL SOUND(-9
99,110*S(J),0):: DISPLAY AT(
12,12):Z(J):: FOR D=1 TO DL
:: NEXT D :: NEXT J
200 CALL CLEAR :: CALL SCREE
N(16):: CALL COLOR(3,2,1,4,2
,1):: X=INT(3*RND+1):: W=INT
(T*RND+1):: ON X GOTO 210,23
```

```
0,210
210 IF X=1 THEN Q$=C$(Y(W))E
LSE IF X=3 THEN Q$=STR$(Z(W)
)
220 DISPLAY AT(12,1):"WHICH
WAS ";Q$ :: GOTO 240
230 CALL SOUND(1,30000,30)::
 DISPLAY AT(12,1):"WHICH WAS
?" :: FOR D=1 TO 200 :: NEXT
 D :: CALL SOUND(500,110*S(W
),0)
240 CALL CALLKEY(12,20,V$,K$
):: Q=ASC(K$)-48
250 IF Q=W THEN DISPLAY AT(1
5,12):"RIGHT!" ELSE DISPLAY
AT(15,12):"WRONG!"
260 IF Q=W THEN DL=DL-50 ELS
E DL=DL+50
270 IF DL<100 THEN DL=500 ::
 T=T+1 :: V$=SEG$(W$,1,T)
280 GOUO 150
290 SUB CALLKEY(R,C,V$,K$)
300 CALL HCHAR(R,C+2,30):: F
OR T=1 TO 3 :: CALL KEY(0,K,
S):: IF S<>0 THEN 330
310 NEXT T :: CALL HCHAR(R,C
+2,20):: FOR T=1 TO 3 :: CAL
L KEY(0,K,S):: IF S<>0 THEN
330
320 NEXT T :: GOTO 300
330 IF POS(V$,CHR$(K),1)=0 T
HEN 300 ELSE K$=CHR$(K)
340 SUBEND
```

I have warned repeatedly over the years, in these Tips and in Micropendium and elsewhere, that printing program listings through the Funlweb Formatter usually results in garbled listings that cannot be keyed in correctly - but I still see the garbled listings published. Here is a fix to the Funlweb FO file that will partially solve the problem - Boot DSKU. Select 1. File Utilities. Select 5. Find String. Enter filename FO and the drive number. Enter H for hex. Enter the string 2A23214026 . Enter replace string 7C2321605C . When the string is found, enter R for replace, then CTRL W, hit Enter twice to accept the defaults. Thereafter, use FCTN Z instead of & to underline, FCTN C instead of @ to double-strike, and FCTN A instead of * to call a value added file. I don't know why Texas Instruments didn't do that in the first place, and I wonder why the McGoverns didn't make that fix.

Now, can anyone tell me how to replace the ^, which tends to disappear, and the period, which will make the whole line disappear if it happens to be at the beginning of the line?

If you are one of the few who are still interested in recreational computing - the use of the computer to solve puzzles and math problems just for the fun of it - you might be interested in Recreational and Educational Computing, published 8 times a year at 909 Violet Terrace Clarks Summit PA 18411. The annual subscription is $36. Program listings are in dialects of Basic other than TI but usually not hard to convert.

That is where I found this ridiculously short, simple and fast card shuffling routine.

```
100 DIM C(52)
110 FOR X=1 TO 52 :: C(X)=X
:: NEXT X
120 FOR X=52 TO 1 STEP -1 ::
 I=INT(RND*X+1)
130 T=C(I):: C(I)=C(X):: C(X
)=T :: NEXT X
```

In the same place, I read a routine to extract a root to 16-digit accuracy instead of the 8 digits available on a PC from the basic formula ROOT=NUMBER^(1/POWER). We don't need it - our obsolete 16k 16-bit computer can give us 14-digit accuracy from the basic formula!

The same publication gave me the idea for this little game -

```
100 DISPLAY AT(3,6)ERASE ALL
:"THE GAME OF N":"":"You and
 the computer will take tu
rns adding to a num- ber to
reach a goal."
110 DISPLAY AT(8,1):"If you
reach the goal, you win. Yo
u get to go first andyou sho
uld be able to win almost
every time."
120 RANDOMIZE :: N=INT(RND*1
5)+15 :: R=INT(4*RND+3):: S=
R+1 :: D=N-INT(N/S)*S :: T=0
130 DISPLAY AT(13,1):"The go
al is";N:"":"Maximum input i
s";R :: DISPLAY AT(19,1):RPT
$(" ",28j6)
140 DISPLAY AT(17,1):"Your n
umber?" :: ACCEPT AT(17,14)S
IZE(1)VALIDATE(DIGIT):A :: I
F A<1 OR A>R THEN DISPLAY AT
(15,1):"" :: GOTO 130
150 T=T+A :: DISPLAY AT(21,1
):"Total is";T :: IF T=N THE
N DISPLAY AT(23,1):"YOU WIN!
" :: GOSUB 190 :: GOTO 120
160 IF N-T<S THEN P=N-T :: T
=T+P :: DISPLAY AT(19,1):"Co
mputer adds";P :: DISPLAY AT
(21,1):"Total is";T :: DISPL
AY AT(23,1):"COMPUTER WINS!"
 :: GOSUB 190 :: GOTO 120
170 IF T=0 THEN P=D ELSE IF
(N-T)/S=INT((N-T)/S)THEN P=I
NT(R*RND+1)ELSE Y=N-T :: P=Y
-INT(Y/S)*S
180 T=T+P :: DISPLAY AT(19,1
):"Computer adds";P :: DISPL
AY AT(21,1):"Total is";T ::
GOTO 140
190 DISPLAY AT(24,8):"PRESS
ANY KEY" :: DISPLAY AT(24,8)
:"press any key" :: CALL KEY
(0,K,S):: IF S=0 THEN 190 EL
SE T=0 :: RETURN
```

REC also printed a puzzle which seemed so simple that I could not see why. It goes like this -

A game show host shows you three curtains. Behind one is a new car, behind the other two are goats. You choose one. The host, who can peek behind the curtain, opens one of those you did not pick, and shows a goat. Then he offers to let you change your choice. Should you switch, stand pat, or does it make no difference?

You now have a 50-50 bet, so it makes no difference, right? But some very distinguished mathematicians were saying you should switch, so I wrote this computer simulation to prove them wrong. Key it in, run it, and be surprised. Do figures lie? Do computers lie? Is there something wrong with my simulation?

```
100 CALL CLEAR
110 DATA CAR BEHIND,A PICKS,
HOST SHOWS,A WINS,B WINS,C W
INS
120 FOR J=1 TO 3 :: READ M$
:: DISPLAY AT(J,1):M$ :: NEX
T J :: FOR J=12 TO 14 :: REA
D M$ :: DISPLAY AT(J,1):M$ :
: NEXT J
130 FOR J=1 TO 1000 :: RANDO
MIZE :: X=INT(3*RND+1):: DIS
PLAY AT(1,13):X !RANDOMLY PL
ACE CAR
140 A=INT(3*RND+1):: DISPLAY
 AT(2,13):A !PLAYER CHOOSES
150 E=INT(3*RND+1):: IF D=X
OR D=A THEN 150 :: DISPLAY A
T(3,13):D :: ! HOST PICKS CU
RTAIN WITH GOAT
160 IF A=X THEN AA=AA+1 :: D
ISPLAY AT(12,7):AA ! A DOES
NOT SWITCH
170 B=INT(3*RND+1):: IF B=A
OR B=D THEN 170
180 IF B=X THEN BB=BB+1 :: D
ISPLAY AT(13,7):BB ! B SWITC
HES
190 C=INT(3*RND+1):: IF C=D
THEN 190
200 IF C=X THEN CC=CC+1 :: D
ISPLAY AT(14,6):CC ! C CHOOS
ES RANDOMLY
210 NEXT J
```

Here is an improved version of a program that was in a Tips long ago, to strip out the extra blanks from a

Filled and Adjusted Funlweb Formatter file -

```
100 DISPLAY AT(3,6)ERASE ALL
:"TIGERCUB UNFILLER":"":" To
 remove extra spaces from":"
a TI-Writer text which has":
"been Filled and Adjusted by
"
110 DISPLAY AT(8,1):"the For
matter, prior to":"reformatt
ing."
120 DISPLAY AT(15,1):"Input
file? DSK" :: ACCEPT AT(15,1
6):IF$ :: OPEN #1:"DSK"&IF$,
INPUT
130 DISPLAY AT(17,1):"Output
 file? DSK" :: ACCEPT AT(17,
17):OF$ :: OPEN #2:"DSK"&OF$
140 LINPUT #1:M$ :: P=1
150 X=POS(M$,"a",P):: IF X=P
 THEN P=P+1 :: GOTO 150
160 X=POS(M$,"  ",P):: IF X=
0 THEN PRINT #2:M$ :: GOTO 1
80
170 M$=SEG$(M$,1,X)&SEG$(M$,
X+2,255):: GOTO 160
180 IF EOF(1)<>1 THEN 140 ::
 CLOSE #1 :: CLOSE #2
```

While a program is running, the computer periodically pauses for a fraction of a second to do a "garbage collection", getting rid of information it no longer needs, to make room in memory. If this pause occurs at a critical moment in program execution, it can cause problems. Thanks to the Sydney User Group in Australia, here is a CALL LOAD which will force a garbage collection just before that critical point -

```
CALL LOAD(-31885,144,"",-318
58,81,169,152,0)
```

Here is a neat one from Bruce Harrison. Key it in, (you can skip the lines that start with an asterisk) and assemble it, then use ALSAVE to imbed it in any program that opens a disk file. Put CALL LINK("DEVICE",DEV$) at the beginning of the program and change any line reading OPEN #1:"DSK1.FILENAME" - or whatever - to read - OPEN #1:DEV$&".FILENAME" (don't forget the period before the filename!). Now you can load the program from any drive and it will open the file on that same drive!

```
* STRING ASSIGN DEVICE NAME
* PLACES DEVICE NAME IN AN
*  XBASIC STRING
* HARRISON SOFTWARE
* 8 OCTOBER 1990
* FOR USE WITH ALSAVE AND XB
* TAKES ONLY 42 BYTES MEMORY
STRASG EQU  >2010
WS     EQU  >20BA
       DEF  DEVICE
DEVICE
* USE OUR WORKSPACE
       LWPI WS
* GET THE CRU BASE IN R12
       MOV  @>83D0,R12
* GET ROM ADDRESS FOR DEVICE
*  IN R2
       MOV  @>83D2,R2
* ENABLE THE ROM
       LDCR @ONES,0
* ADDING 4 PUTS US AT THE
*  LENGTH BYTE
       AI   R2,4
* FIRST PARAMETER
       LI   R1,1
* NOT AN ARRAY VARIABLE
       CLR  R0
* ASSIGN DEVICE NAME TO A
*  STRING
       BLWP @STRASG
* CLEAR CRU, DISABLE ROM
       LDCR R0,0
* LOAD GPL WORKSPACE
       LWPI >83E0
* RETURN TO GPL INTERPRETER
       B    @>006A
* WORD TO TURN ON ROM IN CRU
ONES   DATA >0101
       END
```

Getting short on memory, so more next time.

Jim Peterson

BOOK REVIEW...

FRACTALS FOR THE CLASSROOM
published by Springer Verlag
by Peitgen, Jurgens, Saupe.
In two volumes, available separately, of 350 pages each, hard back.
Part 1: Introduction to Fractls and Chaos
Part 2: Mandelbrot Set and Complex Iterators.

These two books deal more with the math of this interesting family of computer graphics, but there are easily utilised short computer programs included.
Aimed at the "advanced secondary student" the math is perhaps more easily presented than in many Fractal books, and the entire presentation is very readable. If you have an interest in coming to terms with these graphics, rather than just keying in short programs, this could be your best bet.
There are around a hundred illustrations in each volume, including reproductions of the original math papers for certain topics, usually quite short. There is an interesting photograph of the mathematician Julia, forgotten until Mandelbrot appeared on the scene.
My copies came from Germany but you can order the books from any good bookshop.
If you would really like to obtain a good clear understanding of the math, there are also some paperback "support" books available of 128 pages each, under the subheading "Strategic Activities". At the moment there are two volumes available. These books are based on practical work by yourself, not at too high a level - and the answers are in the back. There are a few short listings, for something called "Casio" and for the TI-81, which is the more easily converted.

The first volume of the Activity books includes 9 colour slides of graphic material, the books have specially "slit" pages to allow them to lie flat in a photocopier (educational copying is permitted), and all of these books are on acid free paper for a long life.

These are really handsome books and well worth looking at by anyone with a serious interest in the pretty graphics programs we have been publishing over the past few years.

CATALOGUE RECEIVED:

Media Magic P O Box 598, Nicasio, Ca, USA, 94946
64 pages of computer/math/graphic/science materials.
An unusual catalogue, filled with much to stimulate and inform. Books and videos deal with fractals, chaos, visualisation, virtual reality, machine intelligence, computer graphics and art, mathematics..... you get the idea!
There are posters, postcards, calendars, T-shirts, magazines, and software for some odd computers not being TI99/4as.
UK readers will be glad to know that several (34 to be precise) of the videotapes are available in PAL standard for only $7 extra, which is a lot less than you pay for conversion here.
Videos tend to be around an hour, but some are much shorter. Included are video animations exploring the mandelbrot set, and examples of computer animations.
A very interesting catalogue - I have not dealt with them but they look like a solidly based organisation. They prefer orders are paid for by credit card. ====stephen shaw====

MENU CHOICES by Stephen Shaw April 1992

The TIPS program allows you to input the name of a graphic - but it goes a little farther than that. You do not have to key in the full name of the graphic, just enough to identify it. If there is more than one possible match, the program selects the first possible.

For example, if the graphics on file are ANT, APE, APPLE, when you select A you will have an ANT, if you type AP you will have an APE and for APP you get an APPLE.

We can copy this into our Basic programs when a typed selection is required, like this:

```
1 ! AS TIPS CHOICE
2 !
3 ! INPUT AS LITTLE AS IS
4 ! REQUIRED TO IDENTIFY
5 ! CHOICE
6 !
7 ! FIRST MATCH IS
8 ! SELECTED IF MORE THAN
9 ! ONE CHOICE
10 !
92 !
93 ! DATA IS SORTED
94 !
100 DIM A$(25)
110 DATA BACK,BAG,BOA,BODY,BOND,BONE,CAB,CABIN,CABINET,CAKE,CAR,CARD,CARE,CAROL,
CARP,CART,CASE,CASK,CAT,,,
120 FOR T=1 TO 19 :: READ A$(T):: NEXT T
130 REM
140 PRINT "SELECT FROM":
150 FOR T=1 TO 19 :: PRINT A$(T);"  ";:: NEXT T
160 PRINT "":"":"":""
170 REM
180 INPUT B$
190 LA=LEN(B$)
200 FOR T=1 TO 19
210 IF B$=SEG$(A$(T),1,LA)THEN 240
220 NEXT T
230 PRINT "UNABLE TO MATCH":"":"" :: GOTO 140
240 PRINT "MATCHED ON ";A$(T):"":"":""
250 GOTO 130
260 END
```

Users of other computers will be used to more sophisticated selection routines, and we can go a little way towards these in Basic - not quite as fast as machine code perhaps, but usable... in the following listing, the four arrow keys are usable... see if you can follow the different inputs for different choices. And don't type too fast when inputting a choice...

----> program on next page ->

48

```
100 ! AS PC CHOICE
110 !
120 ! INPUT AS LITTLE AS IS
130 ! REQUIRED TO IDENTIFY
140 ! CHOICE
150 !
160 ! FIRST MATCH IS
170 ! DISPLAYED. TYPE MORE
180 ! IF REQUIRED OR USE
190 ! ARROW KEYS E&S (WITH
200 ! FCTN KEY! )
210 !
220 !
230 ! DATA IS SORTED
240 !
250 DIM A$(25)
260 !
270 DATA BACK,BAG,BOA,BODY,BOND,BONE,CAB,CABIN,CABINET,CAKE,CAR,CARD,CARE,CAROL,
CARP,CART,CASE,CASK,CAT,FORD,FORK,FORT,,,
280 FOR T=1 TO 22 :: READ A$(T):: NEXT T
290 REM
300 PRINT "SELECT FROM":
310 FOR T=1 TO 22 :: PRINT A$(T);"  ";:: NEXT T
320 PRINT "":"":"":""
330 PRINT "after first letter typed,   use fctn e and fctn x to    move up and d
own list or    carry on typing":"":"";
340 ROW=24 :: COL=3
350 CALL HCHAR(ROW,COL,30):: CALL KEY(5,X,Y):: IF Y>0 THEN 370 ELSE CALL HCHAR(R
OW,COL,32):: GOTO 350
360 !
370 CALL HCHAR(ROW,COL,X):: COL=COL+1 :: B$=B$&CHR$(X):: LB=LEN(B$)
380 !
390 FOR T=1 TO 22
400 IF B$=SEG$(A$(T),1,LB)THEN DISPLAY AT(24,1):A$(T):: GOTO 460
410 !
420 IF B$<A$(T)THEN T=T-1 :: CALL SOUND(100,140,4):: DISPLAY AT(24,1):A$(T):: B$
=SEG$(B$,1,1):: LB=LEN(B$):: GOTO 460
430 !
440 NEXT T :: CALL SOUND(100,200,4):: T=22 :: DISPLAY AT(24,1):A$(T)
450 !
460 ! FIRST LETTER CHOSEN
470 ! NOW IS IT WHAT WE WANT?
480 CALL HCHAR(23,3,32,28)
490 CALL HCHAR(23,3,95,LB)
500 !
510 CALL KEY(5,X,Y):: IF Y<1 THEN 510
520 IF X=13 THEN 640 ! GOT IT
530 IF X=11 AND T>1 THEN T=T-1 :: DISPLAY AT(24,1):A$(T):: B$=SEG$(A$(T),1,LB)::
 GOTO 460
540 IF X=11 AND T<2 THEN CALL SOUND(200,200,4):: GOTO 460
550 IF X=10 AND T<22 THEN T=T+1 :: DISPLAY AT(24,1):A$(T):: B$=SEG$(A$(T),1,LB):
: GOTO 460
560 IF X=10 AND T=22 THEN CALL SOUND(200,200,4):: GOTO 460
570 IF X=8 AND LB>1 THEN B$=SEG$(B$,1,LB-1):: LB=LEN(B$):: GOTO 390
580 IF X=9 AND LB<LEN(A$(T))THEN B$=SEG$(A$(T),1,LB+1):: :: LB=LEN(B$):: GOTO 39
0
590 !
600 IF X<32 THEN 390
610 !
620 CALL SOUND(100,800,13)
630 B$=B$&CHR$(X):: DISPLAY AT(24,1):B$ :: LB=LEN(B$):: GOTO 390
640 CALL CLEAR :: PRINT "CHOICE WAS":"":A$(T)
650 RUN                          [end]
```

49

HIGH RESOLUTION GRAPHICS by Stephen Shaw April 1992

In the Reader to Reader column of MICROpendium, March 1992, Chuck McConnell of Ohio wrote asking for a way to plot graphics at pixel level without using a disk drive (which rules out such fun programs as The Missing Link and JBM103).

My first TI computer was a TI99/4, which did not HAVE a high resolution mode of any sort, and given a long standing interest in graphics, I have since that time tried (almost) every graphics program available.

TI Logo was designed for the TI99/4, and in the absence of pixel graphics, utilised a routine which continually redefined characters (or tiles) as you drew on the screen. Sooner or later you ran out of characters, and in the colourful terms of TI Logo, you ran out of ink!

Back in 1982, we were blessed with a routine in TI Basic which allowed high resolution graphics plotting, continually redefining characters, thanks to Peter Brooks, a felow founder member of the first UK user group. This was painfully slow, and as TI Basic does not have CALL CHARPAT had to make use of a large string array, and also boolean algebra had to be done the hard way.

Then along came Extended Basic, which I think most people now have? And Gary Harding rewrote Peter's routine making use of CHARPAT and OR. The program below is an example of its use. The maths is placed in a subroutine, so the only variable you need to avoid in your inserted routines is S, which keeps track of which character we are redefining.

```
100 ! hi res plotting - ti ex bas only
110 ! after brooks, harding etc.
120 ! initialise:
130 S=31 :: FOR C=0 TO 14 :: CALL COLOR(C,16,2) :: NEXT C
140 CALL HCHAR(1,1,S,768) :: CALL SCREEN(2)
150 !
160 !
170 !
180 ! YOUR PROGRAM HERE
190 !
200 !
210 FOR COL=40 TO 140 STEP 100 :: FOR ROW=20 TO 160
220 CALL PLOT(ROW,COL,S) :: NEXT ROW :: NEXT COL
230 !
240 FOR RAD=0 TO 6.5 STEP 0.0125
250 CALL PLOT(36*SIN(RAD)+99,36*COS(RAD)+76,S) :: NEXT RAD
260 !
270 !
280 !
10000 GOTO 10000
10010 STOP
10020 !
30000 SUB PLOT(R,C,S)
30010 IF R>190 OR C>254 THEN SUBEXIT
30020 IF R<1 OR C<1 THEN SUBEXIT
30030 R=INT(R+.4) :: C=INT(C+.4)
30040 Y=INT(R/8+0.875) :: X=INT(C/8+0.875)
30050 H$="0123456789ABCDEF"
30060 B=C-X*X+8 :: P=2*R-16*Y+16+(B<5)
30070 IF B>4 THEN B=B-4
30080 CALL GCHAR(Y,X,H)
30090 IF H>31 THEN 30120 ELSE IF S=143 THEN SUBEXIT
30100 S=S+1 :: D$=RPT$("0000",4) :: CALL CHAR(S,D$)
30110 CALL HCHAR(Y,X,S) :: H=S :: GOTO 30130
30120 CALL CHARPAT(H,D$)
30130 N=(POS(H$,SEG$(D$,P,1),1)-1)OR(S^(4-B))
30140 D$=SEG$(D$,1,P-1)&SEG$(H$,N+1,1)&SEG$(D$,P+1,16-P)
30150 CALL CHAR(H,D$) :: SUBEND
31000 ! ORIGINAL ROUTINE TIDINGS OCT 1982
```


Yes it is just a little slow, but remember it is all Extended Basic with no extras required, no disk drive, no 32k ram!

Chuck also wrote directly to me, asking about using the Drawnplot routines to be found in the Triton module Super Extended Basic.

When driven from a program, Drawnplot has only a limited set of commands, but sufficient for our purposes. The biggest drawback -to me! - is that the image does not appear on the screen until it is finished. For a lengthy chaotic or fractal image this can mean a long time with nothing obvious happening, so in the program below I have added a screen counter. During processing some odd characters appear on the screen - ignore them! - Drawnplot does not really like you to use the screen while it is plotting!

The program below is a routine for a chaotic graphics plot, and really does take a very long time to finish! The end result is interesting as total order, represented by a single line, becomes total chaos after repeated bifurcation.

Super Extended Basic requires that you have the 32k ram attached, and you enter the graphics mode by typing the command sequence:

```
 CALL FILES(2)
 NEW
 CALL INIT
 CALL DRAWNPLOT
```

Now you can input or load your program as follows:

```
10 ! high resolution graphics using
20 ! triton super extended basic and 32k ram
30 ! after brooks, harding etc
40 !
100 CALL LINK("GCLEAR")
110 ! ORBITDGM PROGRAM
120 ! OR insert your program here:
130 FOR C=-2 TO 0.25 STEP .00625
140 X=0 :: M=160*(C+2) :: FOR I=0 TO 200
150 X=X*X+C :: IF I<50 THEN 170
160 N=(180/4)*(2-X) :: CALL PSET(M,N)
170 NEXT I
180 CALL LINK("MOVE",80,160)
190 CALL LINK("LABEL","Press E to Exit")
200 CALL LINK("SHOW")
210 STOP
10000 SUB PSET(X,Y)
10010 CALL LINK("MOVE",X,Y) :: CALL LINK("DRAW",X,Y)
10020 SUBEND
```

\- - - - -

It would be amiss of me not to make this a complete article by covering some other possibilities...

For Myarc Extended Basic, you require the Myarc module, Myarc expansion memory, and the disk and rom chip supplied with the module. The listing immediately above needs the following changes:

```
100 CALL GRAPHICS(3)
180 REM
190 CALL WRITE(0,160,80,"* done *")
200 REM
10010 CALL POINT(1,X,Y)
```

MYARC XB is fast and has the unique ability of being able to tell you if a pixel is on or off - and like the Missing Link you can have sprites in high resolution mode!

For THE MISSING LINK, a commercial disk from Texaments requiring any Extended Basic plus 32k ram and disk system, the following amendments are required to the program:

```
100 CALL LINK("CLEAR")
180 REM
190 CALL LINK("PRINT",160,80,"* DONE *")
200 REM
10010 CALL LINK("PIXEL",X,Y)
```

- - - - -

There is a French utilitiy in circulation called JBM103, which would require the following:

```
100 CALL LOAD(-31890,56,0) :: CALL LOAD(-31964,56,0)
105 CALL LINK("CLEAR") :: CALL LINK("SCR2")
180 REM
190 REM
200 REM
10010 CALL LINK("POINT",16,X,Y)
```

- - - - - - - -

The graphics enthusiast thus has a choice of programming environments- each choice has something to offer. My present first choice tends to be The Missing Link, but there are occaisions when Myarc XB is necessary.

Both The Missing Link and JBM103 can save your art forms in TI Artist format, and I have lost count of the number of utilities you can use with this format!

Enjoy.

. . . . . .

Using USING
by Mark Schafer

This is a new and different animal for me, as I normally do not write tutorials. It came up at a recent meeting that some people are having trouble with PRINT USING. Since I consider it to be no problem, I volunteered to write an article about it. As for the title, remember that USING can also be used in a DISPLAY USING statement. There's a lot of ground to cover, so let's get started with....

Beginner's Stuff

USING represents a method by which you can get data to print in a specific format instead of the default format. The syntaxes are:

```
PRINT USING format:print-list
DISPLAY [option-list:] USING format[:print-list]
```

where format is a line number or a string expression. A line number would be the line where an IMAGE statement is found which would contain the string expression which is the format. A string expression can be as simple as a string literal enclosed in quotes or a complicated expression made up variables, function calls, whatever. Now let's compare the two methods of printing:

```
100 PRINT -56.7;109.2850
110 PRINT USING "####.# ###.####":-56.7,109.285
```

RUN this program and it looks like this:

```
 -56.7  109.285
  -56.7 109.2850
```


USING allows you to better control the spacing as well as put trailing zeroes to the right of the decimal point.

Let's look at the format string. It is made up of fields with optional text. Fields are like fill-in-the-blanks. They mark the place where an unknown value will be printed. Text is printed the same way every time. Fields are made up of pound signs (#) with possibly a decimal point, a minus sign, or maybe some circumflexes (^). I will cover circumflexes in a later section.

Hash (#) signs take the place of a digit or sign. So in line 110 above, I printed -56.7 with the field "####.#". -56.7 has only three characters to the left of the decimal point, so the initial character is left blank; the second one is where the minus sign went. Numbers are always aligned with the decimal point. If there isn't one, it's assumed to be at the end.

If there's a minus sign in the field, it always goes at the beginning. It indicates that you want the minus sign to appear immediately to the left of the number if it is negative. However, this is the same thing another pound sign would do, so you never really need to use a minus sign (except for a more advanced purpose to be discussed later.) In other words, "-##.##" does the same thing as "###.##".

The decimal point indicates where you want it to be. The number of pound signs you put to the left of it dictates how many digits you want to the left; the number you put on the right indicates how many decimal places you want. The decimal is always printed even if there are no pound signs after it.

If there are fewer digits in the value to the left of the decimal point than there are in the field, spaces are used to fill it out. If there are too many, the computer will refuse to print your value and fill the field with asterisks (*). This means your value is too high and/or there isn't enough places in your field. This includes the minus sign, if any.

If there are fewer digits in the value to the right of the decimal point than there are in the field, zeroes are used to fill in the remainder. If there are too many, the computer will estimate the number to the number of places given. So .## can be used to estimate to the nearest 100th, .# to the nearest tenth, and if there are no decimal places, it will estimate to the nearest unit.

Below is a table of how various values will be printed with various fields. The left side represents the value; across the top are the fields.

```
        #   ##   ###   #.#   ##.##   ###.###
2       2    2     2   2.0    2.00     2.000
-13     *   **   -13   ***   *****  -13.000
9.671   *   10    10   9.7    9.67     9.671
125.678 *   **   126   ***   *****  125.678
-.385   *   -0    -0   -.4    -.39     -.385
-3.05   *   -3    -3   ***   -3.05    -3.050
```

Generally, XB handles format strings like this: It keeps printing text until it comes to a field (characterized by a pound sign). Then it looks for the next value to be printed. If there is one, it prints it in the format specified; if there isn't one, it terminates there and doesn't print anything else. However, you must specify at least one value and there must be at least one field in the format. Except, curiously, DISPLAY USING doesn't need any values. Why anyone would use DISPLAY USING without any values is beyond me. Anyway, if it reaches the end of the string, and there's more values to be printed, it goes to the next line and starts over at the beginning of the string. So it's ok if the number of fields doesn't match the number of values.

Going back to syntax, if you want to use the format "I HAVE $###.##", you have three ways of doing it:

```
120 IMAGE I HAVE $###.##
130 PRINT USING 120:M

140 A$="I HAVE $###.##"
150 PRINT USING A$:M

160 PRINT USING "I HAVE $###.##":M
```

Notice that if use the first method using IMAGE, you don't need quotes. The only time you need quotes with an IMAGE statement would be when you have leading or trailing spaces. If you're going to use it repeatedly in a program, IMAGE is the most efficient method. If you're only using it once, go with the third method. Practically the only time you need the second method would be when the format string is constructed so you may not know exactly what it looks like. Remind me, and I might discuss that later.

The IMAGE statement must be on a line by itself. The computer ignores it when it comes to it in a program the same as it does the DATA statement, so you can put it anywhere.

Circumflexes are used to denote scientific notation which leads us to the next section....

Intermediate Stuff

You may want the number to come out in scientific notation (E format) even if the number normally wouldn't. To do this you put four or five circumflexes at the end of the field. Four means you want two digits in the exponent; five means you want three. If you put less than four, they will be treated as text; if you put more than five, the first five will be used, and the rest will be treated as text.

There's a little something different about E format. It always reserves the first character for the sign, so you'll need at least two #'s in the mantissa (or precede the field with a sign). Using the same numbers as above, the table can be amended thusly:

```
        ##^^^^ ####^^^^ #.####^^^^^
2        2E+00  200E-02  .2000E+001
-13     -1E+01 -130E-01 -.1300E+002
9.671    1E+01  967E-02  .9671E+001
125.678  1E+02  126E+00  .1257E+003
-.385   -4E-01 -385E-03 -.3850E+000
-3.05   -3E+00 -305E-02 -.3050E+001
```

This format also estimates whenever it isn't given enough places to express the exact value or tacks on zeroes when given too many. You'll probably not need this format unless you're dealing with exceedingly low or high numbers.

Time to switch to another concept. USING can also be used to format text values. No bells or whistles here, though. Any field that can be used to format a number can also be used to format text. Text is just left-justified, and all characters within a field are treated the same. I can give you a table, but I assure you, it's quite boring:

```
          ### ###.### -###^^^^
BOB       BOB BOB     BOB
JOHN      *** JOHN    JOHN
25% OFF!  *** ******* 25% OFF!
```

In general, it makes more sense to use only #'s when constructing a text field. You may want to do something like column 2 if you're printing a table of numbers, and you want to use the same format string for the column headers as in the table.

Of course you can mix numeric and text values for the same format as in something like:

```
170 IMAGE ##### HAS $###.##.
180 PRINT USING 170:"MARY",123.4

This will print:

MARY  HAS $123.40.
```

Note the computer recognizes the period at the end as a period and not as a decimal point since the field already has one. Even if it didn't, it wouldn't make any difference since the decimal point would be printed at the end.

Now suppose you want part of a line formatted instead of a whole line. Don't worry; you can always put a semicolon at the end of a PRINT or PRINT USING statement, so you can stay on the same line for the next PRINT. You can also use this technique if you need to print a # as text instead of a field character. However, you cannot put a comma at the end of a PRINT USING statement or the computer will think you left out a value.

Another problem you may have is suppose you want to concatenate two fields, say ## with ###. If you put them together, you get #####, and the computer will see that as one field. Well, you could try using "##-##". The computer will then recognize that as two fields, ## and -##. But if your second field can have three digits, this won't work. If this is the case, you will need to split up the format string into two PRINT USING statements. Perhaps like this:

```
190 PRINT USING "##":A;
200 PRINT USING "###":B
```

You cannot confuse the computer with pound signs, minus signs, circumflexes, decimal points, etc. It will always know where one field ends and another begins. Except in the case like above where two fields collide. I'll give you another way to handle that in....

Advanced Stuff

There isn't very much in the way of advanced stuff because this is such a small subset of a much bigger entity. I always consider undocumented features to be advanced stuff. That's right, boys and girls, USING has an undocumented feature! By "etc." in the above paragraph, I meant the plus sign! It can also be used like the minus sign in a field. What it will do is float the sign in front of the number be it positive or negative. Let me illustrate:

```
         +#  +###  +###.##^^^^
2        +2    +2  +200.00E-02
-13      **   -13  -130.00E-01
9.671    **   +10  +967.10E-02
125.678  **  +126  +125.68E+00
-.385    -0    -0  -385.00E-03
-3.05    -3    -3  -305.00E-02
```

If it's negative, you get a minus sign; if it's positive or zero, you get a plus sign. There isn't one word about this in the XB manual. This can be used to write format strings for equations, like this:

```
200 IMAGE ###x^2+##x+##
210 PRINT USING 200:2,7,10
220 PRINT USING 200:-15,-11,1
230 PRINT USING 200:+9,33,-14
```

This will print:

```
 2x^2 +7x+10
-15x^2-11x +1
 9x^2+33x-14
```

Suppose you don't want it to print those leading spaces when there aren't enough digits to fill the field. This is where you would use a variable for the format string. You would construct it so that each field would only have as many #'s as it needed to print the number (since they're all integers, that can easily be accomplished with LEN(STR$(X))) and concatenate the necessary text and use the variable as the format string. For instance, I have a program that uses a define function like this:

```
240 DEF MF$(X)=" $"&RPT$("#",LEN(STR$(INT(X))))&".##"
```

That function generates a format string for money so that there will no spaces between the dollar sign and the amount. The reason I did it this way was to get trailing zeroes after the decimal point to look good. However, user-defined function calls take a long time in Extended BASIC, so if need speed or if you only need it once, put a formula like this directly where you need it.

There's another manual correction that must be made (although it may have been made already in some addendum I don't know about). The syntax for PRINT USING with files is wrong. I won't reprint it here because I don't like glorifying the incorrect. The correct syntax is as follows:

```
PRINT [#file-number,[REC record-number,]USING format:print-list
```

That translates specifically as something like:

```
250 PRINT #1,USING 200:A,B,C
260 PRINT #2,REC 15,USING 200:A*4,B*4,C*4
```

The file being referenced is most likely the printer, but it could be a disk file. If it is, it must be a DISPLAY format file; you will get a FILE ERROR if is INTERNAL format. Obviously line 260 isn't refering to the printer.

PRINT USING is very good for printing to a printer because the printer has so many more columns to print in. So there is a good chance that you may have a program in which a particular PRINT USING is only being used to print to the printer and never to the screen. If this is the case, there is another solution to the concatenated field problem mentioned in the last section. You can print some unprintable character between the two fields. The computer will then recognize them as two fields but they will be together on the printout! Something like this would be typical:

```
270 A$="##"&CHR$(0)&"###"
280 PRINT #1,USING A$:A,B
```

CHR$(0) doesn't do anything on most printers, so the two fields will appear consecutive. Note that you couldn't do that in an IMAGE statement since you can only use characters and not expressions. The expression could've also been constructed within line 280 itself.

This should about exhaust the subject of USING except to remind you that you can also use DISPLAY USING if you need to format values somewhere else on the screen (by using the AT option) or if want to BEEP or clear the screen before doing it (ERASE ALL).

This test is from "Algorithm Design" by G.J.Christensen,see next TI*MES.

1. Write an algorithm for placing a phone call.
2. Include in the algorithm for question 1 the possibility of busy signal, no dial tone and no answer.
3. Write an algorithm to test if a string is a palindrome. (A palindrome is a word or sentence that reads the same forward as backward, ignoring the spaces. eg RADAR or A MAN A PLAN A CANAL PANAMA)
4. Devise an algorithm for a programme that will ask for numbers to be input until the value 0 is entered then print the largest number that was input.
5. Write an algorithm for a programme that will ask for a input of a single positive number then print the number with the order of the digits reversed. eg. 13542 results in 24531.

ARRAYS AND SORTS
by Jim Peterson

The concept of arrays, and especially of multi- dimensional arrays, is very difficult for many people to grasp. The following is the best explanation that I know of.

A variable name is a box in which you store some- thing. When you write A$="X" you are telling the compu- ter to "go to the box labeled A$ and put the char- acter "X" in it". Or, more acurately, "go to the box labeled A$, throw away any- thing you find in it, and put "X" in it."

A simple array such as A$(3) is a row, labeled A$, of at least 3 boxes, labeled (1), (2), (3), and maybe more. When you tell the computer that A$(3)="X" you are again telling it to go to the row of boxes labeled A$, find the box labeled (3), and put "X" in it.

A 2-dimensional array such as A$(3,3) is a row, labeled A$, of at least 3 filing cabinets, labeled (1, and (2, and (3, and each having at least 3 drawers labeled 1) and 2) and 3). So, you can use A$(3,3)="X" to tell the computer to find the row of filing cabinets labeled A$, go to the one labeled (3, and open the drawer labeled 3) and put "X" in it.

And in a 3-dimensional array, A$(3,3,3)="X" tells the computer to find the A$ row of cabinets, find the one labeled (3 and find the drawer labeled ,3, and find the folder in that drawer labeled 3) and put.....

Finally, you can write A$(2,2,2,2,2,2,2)="X" to tell the computer to find row A$; cabinet (2 ; drawer ,2 ; folder ,2 ; paper 2, in the folder; line 2, on the paper; word 2, on the line; and letter 2) of the word!

Yes, TI Extended Basic can handle 7-dimensional arrays, but it is not very practi- cal. Try running this -

```
100 DIM A(3,3,3,3,3,3,3)
```

and you will get MEMORY FULL IN LINE 100. Arrays with several dimensions are very wasteful of memory. I don't think I have ever seen a program that used more than a 4-dimensional array, and very rarely more than 3 dimensions.

Now then - A$(J)="X" means "go to the box labeled "J", find the number in it, then go to the row of boxes labeled A$ and find the box in that row which is labeled with that number....."

And even something as horrible-looking as A$(Y(J),Z(A,B))="X" just tells the computer to -

1. go to box J and find the number in it;
2. go to row of boxes Y and find the number in box number J of that row;
3. go to box A and find the number in it;
4. go to box B and find the number in it;
5. go to the row of filing cabinets labeled Z, find the one labeled with number A, open the drawer labeled with number B and find the number in it;
6. go to the row of filing cabinets labeled A$, find the one labeled with the number you found in Y(J), open the drawer labeled with the number you found in Z(A,B) and;
7. put the "X" in it!

Simple, isn't it?

Remember that, in a multi- dimensional array, only the last dimension holds the value; the others are just pointers to its location.
A$(2,3)=A$(3,3) throws out whatever is in the 3rd drawer of the 2nd cabinet of the A$ row, and replaces it with whatever is in the 3rd drawer of the 3rd cabinet of that row, but the contents of the 3rd drawer of the 3rd cabinet are unchanged.

Also remember that box X or box X(1) or cabinet drawer X(1,1) or whatever, contain a 0 until you put something else in; box X$ or X$(1) or drawer X$(1,1) contain nothing at all until you put a string value into them. When you put something in the box, you throw away whatever was previously in the box. And to empty a box without putting anything in, you put a 0 in a numeric box or "" into a string box.
Enough, on that subject. Now, when you have all your data crammed into an array, the next thing you will probably need to do is to sort it into alphabetic or numeric sequence.
Sorting is one of the hardest jobs that you can give to a computer, and one of the things that a com- puter is the slowest at doing. Your TI can figure your bank balance in a split second, but might take half an hour to sort your mail- ing list.
Here's why. You can sort a bridge hand of 13 cards into sequence in 13 moves or less, by simply pulling out each card and slipping it back into its proper place. But, suppose those 13 cards were in 13 boxes, and you had to sort them without removing them from the boxes, except that you could hold one card in your hand?
Even if you could figure out the best way, it would take you far more than 13 moves.
That is the problem that the computer has. You have just learned that the com- puter stores all those values in labeled boxes, or file drawers, and therefore must sort them by shuffling them from one box to another, emptying a box to shuffle into by holding one value in a temporary box while its value is compared with the others to find its proper place.
Of course, you could just set up a new row of empty boxes, and then search through the old boxes for the lowest value and move that to the first box in your new row, etc. - but that would double the amount of memory that the job would require. This would be no problem for a small array, but the computer can sort small arrays fast enough by the one-row method - it is the largest arrays that are too slow by the one-row method and would need too much memory by the two-row method.
Many ingenious routines have been written to accom- plish these one-row sorts. I have written a program called "Sort Watcher" which enables you to actually watch various sorts taking place on the screen. It will also tell you the number of swaps and comparisons that were made.
This program demonstrates that the time required for a sort increases greatly as the size of the array in- creases. Sorting an array of 20 does not take just twice as long as sorting an array of 10 - it may take 4 times as long. For this reason, some of the faster and more complex sorting routines divide an array into smaller segments to be individually sorted and then merged.

After an array has been sorted, my program will also let you change any value in any part of the array, and then let you watch the array being resorted. From this, you will learn that a sort- ing routine which is very fast for a completely ran- dom array may be very slow for an array which is already almost in sequence!
In fact, to add just one additional value to a sorted array, the fastest method is the simple "shoehorn" - just set up an empty box at the end of the row, and move each value down by one box until you come to the proper place for the new value.

A sorting routine can be either numeric or alphabetic depending on whether the variable names used are numeric or string. A numeric sort will be in strict numeric sequence and an alphabetic sort will be in ASCII sequence. That means that if all your strings are composed of upper case alphabetic characters, or all are lower case alpha- betic characters, you will get an alphabetic sort - but if they are mixed, all of the upper case strings will come before any of the lower case strings, because the upper case ASCIIs are 65-90 and the lower case are 97-122. And if you have lower case words with capi- talized initial letters...!
For the same reason, if you perform an alphabet sort of strings containing num- eric digits, you will not get a numeric sequence - 10000 will come before 2 because 1 has a lower ASCII code than 2. It would be extremely difficult to devise a sorting routine which could sort numeric digits numerically within strings. However, if all the numbers are the same length, such as ZIP codes, the ASCII sort will be numeric.

Sorting a multi-dimension array becomes a very com- plex task. If you swap values around without also swapping all the related values, you will end up with complete garbage. Swapping all the related values takes time, and a dimensioned temporary variable name is also required.

Another way around this is to combine the data from an array into simple strings, or set it up originally as simple strings, and then perform a simple sort based on a specified segment of the string. For instance, you could use TI-Writer with tab settings to create a mailing list having first name at tab 1, second name at tab 15, address at tab 25, city at tab 45, state at tab 55 and zip code at tab 65. Then you could sort into last-name alphabetic sequence by sorting on SEG$(M$(J),10,255), or into zip code sequence by sorting on VAL(SEG$(M$(J),70,5)).
When using TI-Writer to set up such a file, be very sure to save it by PF with the C option, not by SF, and don't leave any blank lines at the end or elsewhere.

Alternatively, elements of data can be crammed into a string separated by control codes, and sorted by position of the code -

```
FOR J=1 TO 5 :: READ A$ ::
M$=M$&CHR$(J)&A$  ::  NEXT J
and  then  sort on element X
by -
SEG$(M$(J),POS(M$(J),CHR$(X)
,1),255)
```

= = = = = = = = = = = = = = = = = = =
Plenty to think about (and perhaps disagree with?) in this discussion...

FILE PROTOCOL
by Mark Schafer

This article is way too late. About 10 years too late. TI should've read this article before they made TI Writer! They're using the wrong file format, and you probably are, too.
Let's look at how files are stored on disk. No, I'm not going to get technical and tell you how you can use sector editors to look at and/or change things in a file. This will be a general approach. First, you should be aware that disks are divided into sectors. Catalog programs usually tell you how many of those you have free on a disk. One whole sector is used for each file to tell the computer everything it needs to know about it before it tries to read or write to it (the header sector). But I'm not going to focus on that, either. I'm looking at the part of the disk on which the content of the file is stored.

Now how many file formats are there? Only counting the ones XB can manipulate on its own, there are four: Internal/Variable, Internal/Fixed, Display/Variable, and Display/Fixed. With any of these file types the computer never splits a record between two sectors. If there's not enough room on the current sector to store the next record, it will start it on the next sector. This is key to my discussion here.

Starting with variable-length files, the maximum number of characters you can get in one record is 254 (253 in internal in BASIC). There are 256 characters in each sector. As a quick technical excursion, the computer uses the other two characters as a length byte and an end-of-record marker (why it needs both is beyond me). Anyway, you can optionally specify the maximum record length when you open a file (in any language; that's the beauty about this article: it applies to ALL programming languages.) TI Writer files, for instance, are in DIS/VAR 80 format, so they are limited to 80 characters per record.

Get ready because here comes one of my key points! Why limit the record length to 80 characters when you can have 254? Well, you might say it takes up less disk space. Wrong! Each record in a variable-length file takes up only as many bytes as it needs to. In other words, if you changed every DIS/VAR 80 file to DIS/VAR 254, nothing would change. It would still take up the same amount of space and take the same amount of time to load. Don't do that because TI Writer won't be able to read them.

So, you might say, why reserve space for each record you'll never use? After all, TI Writer only puts 80 or fewer characters in each record; even if the format were DIS/VAR 254, the extra characters would probably never be used. The point is there's no reason not to. If they had done it this way, DIS/VAR^254 would be the standard that DIS/VAR 80 has become. Then all programs that used DIS/VAR files would be able to read each other's files, although, they may not be able to make sense out of them, but at least the option would be available.

So if you use DIS/VAR or INT/VAR, you should only use 254. The only reason not to now is to maintain compatibility with other programs. And that's unfortunate because that's a good reason. That's what I mean about being too late. Oh, well, if compatibility isn't important for a particular program you're working on, then go with 254. This will give you the ability to add more characters to the record should it prove necessary later on without having to change the record length.

Yes, I am also going to attack fixed-length record files. Now the maximum is 255 characters (254 in internal in BASIC). The length byte and the end-of-record byte aren't needed since all records are the same length. And since that is the case, the number of records in each sector is fixed. So in most cases, there will be unused bytes in every sector. We need to try to reduce the number of unused bytes without taking up more sectors.

Obviously, FIXED 255 doesn't work because you'd only get 1 record per sector no matter how small. I'll just give this one to you and tell you the reason afterwards. Take 256 and divide it by the number of bytes you plan to have in each record, and drop the decimal. Then take this number and divide it into 256, and drop the decimal again. This is what you should fix the length at. For those of you who are not mathematically inclined, let me give you a list of numbers: 1 through 19, 21, 23, 25, 28, 32, 36, 42, 51, 64, 85, 128, and 256. You can't use 256 since the computer cannot represent that number in a single byte, so you have to use 255 instead. If the length you want to fix is not on the list, move up to the next higher number that is. So if you want to use INT/FIX^43, 43 is not on the list, so you'd use INT/FIX 51 instead.

Let me explain. You see with 43 bytes per record, you'd get 5 records per sector on the disk. That would amount to wasting 41 bytes per sector (trust me). If you use 51 instead, you still get 5 records per sector but only waste 1 byte per sector. Same argument as above against "...but I'm reserving more space than I need." You'll have the ability to add more characters later on without taking up more space or changing your file format. This won't work out every time; you may have to change the file format some time, but there's no reason to use 43 when 51 costs you no more disk space. The same argument applies to every number not on the list.

This is very important for the numbers between 128 and 255. If you use anything in that range, you'll only get 1 record per sector. As an alternative to using 255, you might look for a way to reduce the number of characters in each record especially if you're only a little over 128. If you can get it down to 128, you won't believe the space you'll save! Suddenly, you'll get two records per sector taking only about half the space!

This rule also applies to variable-length record files. As a case in point, I had a program which used a variable-length record file with 64 records. I put so much space in each record, that the file took up the full 65 sectors (one more for the header sector). So I split it into two files, so I could get more records per sector. On one file I get two records per sector; I get three on the other, so the two files combined ngw take up only 48 sectors! If I had read this article before, they would both be INT/VAR 254 files. So remember even variable files can benefit from the number list given above. If you can lower the MAXIMUM number of characters per record in files with consistently long records, you can save serious disk space. But you'd still use VARIABLE 254.

In fixed-length records, the most efficient numbers are the powers of 2 (128, 64, 32, etc.) because they use every byte of a sector. This is why archived files are in INT/FIX 128 format. It wastes no space in the fewest reads.

Whereas all the above applies to any language, in assembly language you have the option of using PROGRAM files which use every byte up to the last sector, and are therefore the most efficient (not to mention quicker load time). So in assembly language, you might want to think about this format for program-specific files. This is the way fractals are stored in FRACTAL EXPLORER.

= =

>DEFAULT FILING<

by Steve Burns

As I sit here going through my disks of newsletter files wondering what this months "Editor's Desk" column will be about I am suddenly hit with a realization. Although I had carefully planned how I was going to organize my disks for past and present newsletter articles it did not happen that way.

My organizational method has again slipped into what is normally known in computer terminology as DEFAULT. Default, as you probably know, is what happens during the course of using a program that requires you to make a decision at a certain point and you don't.

This leaves the decision up to Fate (well actually, up to whatever the programmer has decided what is most likely to work, but quite often the two are close to equivalent).

How does this compare to my disk organization ? Very well. Through careful planning I am able to look through at least 10 or 12 disks each time I need to find a particular article. This is roughly comparable to the method I use to organize my files for new programs . The only difference is that I have to look through about 20-30 disks. I do have a method of sorting disks though... those with labels and those without. Nice neat method.

I used to think that it was because I was short on disks, or disk cases, but with those problems solved, I still find myself slipping in to the same old habits of sticking a file here and a disk there. Despite what others may think, I really DO know where everything is (uh, at least roughly). And despite all my excuses, I know that this mess is DEFAULT of de user.

= =