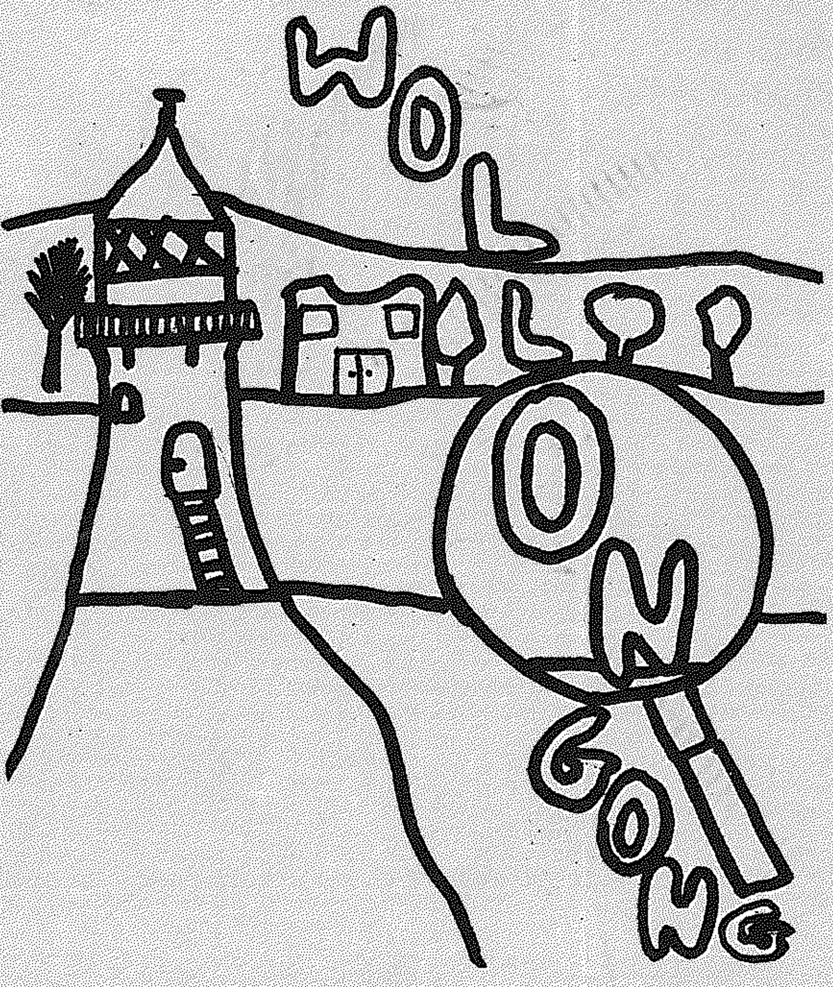
NEWS DIGEST

Focusing on the TI-99/4A Home Computer

Volume 6, Number 9 **October 1987**

Registered by Australia Post - Publication No. NBH5933



Angela Reay, Corrimal Primary 1987

P.O. Box 214, Redfern, New South Wales, Australia, 2016 **$2**

TISHUG NEWS DIGEST

TI99/4A Owners Home Computer
User Group
TIsHUG NEWS DIGEST

OCTOBER 1987

Correspondence to:

PO Box 214
REDFERN NSW 2016

Texpac BBS: Tel.: (02)319.1009

MEMBERSHIP AND SUBSCRIPTIONS:

Joining Fee	$ 8.00
Annual Family Dues	$25.00
Dues O'seas Airmail	US$30.00
Publications Library	$ 5.00
Texpac BBS	$ 5.00
BBS Membership: Other TI User Group Members	$10.00
Public Access	$25.00

GROUP GENERAL MEETING:

First Saturday of each Month at Shirley House, Church Street Burwood. Starts 2:00pm

COMMITTEE MEETINGS:

Before the main monthly meeting. Starting at 12:30 pm.

TIsHUG NEWS DIGEST ISSN 0819-1984

CONTENTS

Unfortunately this months TND has a number of regular columns missing. It appears that work commitments os sickness has caught up with the authors. However, not all is lost. Our friend Jim Peterson of Tigercub fame has provided the club with quite a few articles. A number of them are split into parts and will be continued for a number of months.

I received some bouquets and brickbats for the last issue of the TND. The bouquets are always welcome and go a long way to justify all the hours spent in producing the magazine. The brickbats??? Well, I guess I deserved that one. See the article on Text Manipulation by D.N.Harris. On the subject of Daniel, he is one of the few people letting me know what he is doing. Frequently it is not printable and other times it is. His normal means of communications is by the BBS and suprisingly he is only cassette based. This magazine should be a vehicle to tell others what you are doing. The BBS mebers particularly can leave mail, no matter how short, and if it is of interest then it will be published.

Please read carefully the article announcing the SOFTWARE COMPETITION. It is important that we as a club support the overseas groups as they are supporting us. Software is just one way we can do that.

I received a letter from Geoff Trott during the week. He is in Britian at present and has made contact with the Brits. It appears That they are somewhat ignorant of what is happening here in Aistralia. He was also questioned about GRAPHX. Apparently it has not been reached them yet. Geoff has many names and addresses that he will pass on to Terry when he gets back, early Nov.

Don't forget the big Auction day next meeting.

Bob Montgomery

986191

Software Spoke
3 Sth Ck Rd
Dee Why
9131390

CO-ORDINATOR'S REPORT

.... Chris Buttner

The Memorandum and Articles of Association for the club's incorporation are now the the Corporate Affairs Commission. You should soon be enjoying the personal protection offered by that incorporation.

My work this past month has prevented my continuing, temporarily, the Data Base programme started in the magazine last month. I can now see the light at the end of the tunnel so with your indulgence, the article will resume with the November issue. If there is something in the programme so far which you don't understand, ask me (personally or mail) or discuss it with other members at a regional group meeting.

Our editor, Bob Montgomery, has now been provided with a printer buffer to use with the club daisy wheel printer. It is one of the pBUFF units sold in kit form by the club and works perfectly. The buffer size is 256K so rest assured Bob can handle anything you send him by way of magazine articles. Let's all try to have the magazine reflect what is happening in the Australian TI community.

While on the subject of pBUFF, I can highly recommend it as a worthwhile project. It is very fast and lets you get on with computer work rather than being at the mercy of the printer (whatever its speed). If you feel building the kit is beyond you, there are club members who will assemble it for a moderate fee.

Do you have one of the RAM cards sold in kit form by the club? If so have you got a copy of John Johnson's ram disk operating system? No - then go out and get one quickly. You will then enjoy menu selection of your favourite applications (no more TI colour bar screen: just your menu ready to go!). Don't be affraid to customise the menu. The directions in the source code are very easy to follow and no programming knowledge is necessary. Try to get the latest version (I have seen Version 5 but understand there is now a Version 6).

One of the benefits I derive from the new ROS is the ability to use the RAM card with my Myarc 512K memory card. Previously, these two cards did not get along at all. Fortunately, this has now been solved and there are no incompatibilities. If your system is similar to mine, be sure to use the CALL NF() to correctly configure your system. The value for NF is the actual number of floppy drives you have connected (ram cards don't count). Failure to do this may result in your not being able to access one of your floppy drives.

With the proliferation of ram cards I believe it is timely to issue a word of caution which quite simply is BACKUP. The cards are such a joy to use that it is very easy to forget data can be lost. I have data files and programmes on my cards but they are always backed up at the end of each session. Put your backup onto a floppy disk, label it and keep it in a safe place: particularly if you have customised your operating system.

I recently had the unfortunate experience of seeing a 20Mb hard disk "give up". I can assure you it is really frustrating not to mention time consuming when something such as this happens. The remedy is really quite simple and cheap. Don't be caught! Back it up now.

Our October meeting is traditionally our auction day. There will be time set aside before the action starts to allow your to reach a private treaty. Of course this means first in best dressed so don't be late.

If you are a country member attending our meetings please make yourself known to myself, Terry Phillips or Russell Welham. If you have special interests we can then direct you to one or more of the discussion groups which invariably get under way.

Continued on P16

LET'S ROUND UP THE MAVERICKS! by Jim Peterson

A maverick, for the information of you tenderfeet, is a young Texas critter which has lost its mama. There are over a million of them hiding in the closets of America, and I think it's time for a roundup!

There are perhaps 200, possibly 300, TI user groups in the United States and elsewhere in the world. A few boast of several hundred members, but some have no more than a dozen, and I doubt that the average is more than 50 users actually paying dues and attending meetings. That computes to at most 15,000 members of the "organized" TI world. Of course, there are many others who keep in contact by subscribing to those magazines which support the TI, and still others who are kept up to date on new developments by the catalogs from the big mail order houses. Still, no matter how you compute it, there are certainly well over a million owners of the TI-99/4A who have no way of knowing that our computer is still alive and well.

These people have read that Texas Instruments abandoned the computer. They have seen the supplies of hardware and software disappear from the big retail stores. Many of them bought their computer during the final suicide sales, therefore never got on the mailing list for the Texas Instrument newsletter.

And yet, relatively few of the TI-99/4A are showing up in the classified ads and in the garage sales. A recent national survey found that the TI-99/4A was owned by more people than any computer except the Commodore.

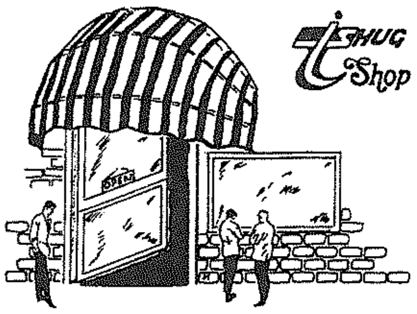
True, many of these owners are only interested in plugging in a module and playing a game. But some have a deeper interest - and even five percent of a million is an awful lot of people!

When I bought my TI, in March of 1982, I searched in vain through the articles and ads of every magazine on the newsstand, for anything relating to my computer. It almost seemed that there was a conspiracy of silence. I had taught myself to program, and written dozens of programs, before I finally made contact with the TI world. I was once a maverick, and I can sympathize with those who are mavericks now.

Is your user group dwindling away, as some of your members move on to bigger but not necessarily better computers, while others become so polarized in their interests that they have little in common with each other? Are your givers tired of giving to your getters, and your doers tired of being used by your users? Do you miss the enthusiasm and excitement of your first meetings, when everyone was learning together? Does your group need a transfusion of fresh blood? The donors are out there and waiting, if you can find them!

Do you want to see new hardware, new software, new publications for your computer? The bigger the market, the more that will be produced to be marketed. And the market is there - it just doesn't know that it's there!

The user groups are the only ones who can round up the mavericks. You can do it by publicizing your meetings, by letting the TI owners in your community know what you can do for them. You can get newspaper publicity and television publicity. Some of you are already offering classes in programming or in computer use to the general public, to the schools, to libraries, to senior citizens, to foster children, to the handicapped. These are very fine endeavors in themselves, and they can also bring the publicity which will attract new members. And here and there among those new members will be an ingenious hardware hacker or programming genius who will make our computer better than ever. o



TIsHUG SHOP OCTOBER 1987

Well it appears the person who got the Z80 chip in his set of memory chips has found a way of using it as he has not asked for a replacement 6264LP-15 chip.

RAM DISK CARD STATUS REPORT:-

The second run of RAM CARD PCB's have sold out. This makes 122 cards total that have been sold.

A short supply of components remain:

(a)	6264L-15 RAM CHIPS (13)	$ 71.50
(b)	" " " (24)	$132.00
(c)	256K Ram expansion kit	$ 47.00
(d)	Batteries AAA in stick form	$ 14.00

PRINTER BUFFER :-
All the existing orders have been filled and it appears that the numbers required for bulk buying of components in the future will not be met, so I think it is better for the members to buy their own.

The parts remaining are:-

(a)	P-BUFF PCB,EPROM,CRYSTAL,8255	$ 51.00
(b)	41256 memory chips (8 req'd.)	$ 40.00
(c)	BPIO board and components	$ 27.00
(d)	Computer sharer board & comp	$ 18.00
(e)	Printer sharer board & comp	$ 18.00
(f)	Plastic box (D/S 2508) small	$ 9.00
(g)	" " (D/S 2505) large	$ 9.50
(h)	9 volt transformer	$ 6.50

NOW THE STANDARD ITEMS:-

(a)	HFi DS/DD 5 1/4" Disks (box)	$ 19.00
(b)	Spike Protectors	$ 29.00
(c)	T.I. Joystick handles	$ .50
(d)	Peter Schubert's mini-expansion unit	
	DS/DD Disk controller card	$190.00
	Mini-PE mother board (with one of either-32K mem :: PIO :: RS232 port)	$ 85.00
	Extra options on mother board	
	32K memory	$ 50.00
	PIO printer port	$ 50.00
	RS232 port	$ 50.00
	Finished painted box for Mini PE	$ 35.00

SECOND HAND ITEMS :-

(a)	Grom Ports	$ 12.00
(b)	Ivory Console Cases	$ 2.00

BOOKS :-

(a)	Back issues of SND	$ 1.00
(b)	Technical manual	$ 15.00
(c)	TI-writer manual	$ 15.00
(d)	Editor Assembler manual	$ 28.00
(e)	TI LOGO Curriculum guide	$ 10.00
(f)	TI 3 ring binders	$ 4.00
(g)	Micropendiums 1986-June to Dec./1987-Jan.to July	$ 3.00

SOFTWARE:-

(a)	Club Software Tapes	$ 3.00
(b)	Club Software Disks	$ 5.00
(c)	Picasso Publisher V1.1 (Arto Heino)	$ 20.00

POSTAGE:-
Please NOTE that with all mail orders YOU have to pay postage and packaging.

If you are phoning the SHOP please note that I am NOT normally available before 7pm week days. (02)639 5847

The next meeting of TIsHUG is a AUCTION at 2pm, 3rd October 1987 in the Woodstock Community Centre, Church St, Burwood.

Please have all items in by 2pm for pre-auction viewing and sale by private barter. The auction will commence at 2.30pm.

Topics of future meetings...

7/11/87...Tutorial Day
5/12/87...Christmas Party
No January meeting
6/2/88 ...Annual General Meeting & election of committee

Regular meetings 2pm first Saturday of the month, except January.

FRANK AND ERNEST
AUCTION
FOLKS, ONE OF OUR PATRONS HAS LOST HIS WALLET WITH $1000 IN IT AND IS OFFERING $100 REWARD TO THE FINDER.
$110!
THAVES

```
*****************************
* PICASSO PUBLISHER Ver 1.3 *
*****************************
```

This article is mainly to clear up a few doubts and to enhance the poductivity of those who own a copy.

SAVING,LOADING & USING FILES

To save your Picasso file press 'S' then enter your file name eg; DSK1.PIC the file will be saved an 84 sector DISPLAY VARIABLE 80 type. The reason I chose that particular file format is so I can use graphic files from other computers without too much fuss.

To load a Picasso file press 'G' then enter the file name. If the file is not 84 sectors in length it will still load it but will show an error before returning to the picture screen, if you press '8' (UNDO) then you will see the part of the file that loaded.

To create a Picasso file from XB I have included a few routines on the disk.

```
10 CALL INIT :: CALL LOAD("DSK1.CHARS/O")
20 OPEN #1:"DSK1.YOURTEXT"
30 OPEN #2:"DSK1.PICASFILE"
40 FOR X=1 TO 42 :: LINPUT #1:A$
50 A$=SEG$(A$,1,60)
60 A$=A$&RPT$(" ",60-LEN(A$))
70 FOR Z=1 TO 60
80 CALL LINK("CHRPAT",ASC(SEG$(A$,Z,1)),8,B$)
90 PRINT #2:B$
100 NEXT Z :: NEXT X
110 CLOSE #1
120 CLOSE #2
```

This routine will read a TEXT file from the disk then send it back in a format suitable for Picasso. This routine could be changed so you could read a long TEXT file and create many Picasso files. Just add:

```
25 N=0
30 OPEN #2:"DSK1.PICASFILE"&CHR$(65+N)
35 ON ERROR 200 :: IF EOF(1)THEN 200
110 CLOSE #2
120 N=N+1 :: GOTO 30
200 CLOSE #1
```

If you want to change the letter fonts in your text just add:

```
15 OPEN #3:"DSK1.FONT-?" :: FOR X=0 TO 9
16 LINPUT #3:A$
17 CALL LINK("CHAR",32+X*10,A$)
18 NEXT X :: CLOSE #3
```

Or if you want to change the fonts at a particular text line(s) just add:

```
15 INPUT "HOW MANY FONTS ?":AM
16 FOR E=1 TO AM
17 INPUT "FONT FILE NAME: ":F$(E)
18 INPUT "AT WHAT LINE ?  ":L(E)
19 NEXT E
45 LL=LL+1
65 FOR D=1 TO AM
66 IF LL=L(D)THEN GOSUB 300
67 NEXT D
300 OPEN #3:F$(D) :: FOR G=0 TO 9
310 LINPUT #3:V$
320 CALL LINK("CHAR",32+G*10,V$)
330 NEXT G :: CLOSE #3 :: RETURN
```

You can of course do all of the above by using the LOAD TEXT FILE option in Picasso file menu and load your fonts as you need with LOAD FONTS.('2')

Once you have created your Picasso files you can now start to do things with it eg. adding graphics,borders,large letters heads(V1.3)etc....

You could create your own library of pictures by saveing only the screen area using the SAVE CURRENT SCREEN option in the file menu.

PICASSO VERSION DIFFERENCES

V1.1	= Fixed the bug so you can Catalog Drives 3,4,5,6
V1.2	= Added extra option on File Utilities so you can Save the Screen area only. Cleaned up the garbage that came on the screen when booting. Added the SET-SCREEN function so you can set the screen before using the UNDO. Added a XB Program on the Disk so you can us CSGD Files, which will give you Large Fonts.
V1.3	= Added an option when Printing so you can Overstrike each line up to 7 times, incase you need a very dark copy.

The LINE routine in the 1987 MAY issue of TND was incorrect:

```
10600 SUB LINE(X1,Y1,X2,Y2):
: X1=INT(X1):: Y1=INT(Y1)::
X2=INT(X2):: Y2=INT(Y2):: X3
,X4,X5,Y3,Y4,Y5=0
10610 IF X1<X2 THEN X3=1 ::
X4=X2-X1 ELSE IF X2<X1 THEN
X3=-1 :: X4=X1-X2
10620 IF Y1>Y2 THEN Y3=-1 ::
 Y4=ABS(Y2-Y1)ELSE IF Y2>Y1
THEN Y3=1 :: Y4=ABS(Y1-Y2)
10630 IF Y4>X4 THEN X5=X4/Y4
 :: Y5=1 :: Z1=Y4 ELSE IF X4
>Y4 THEN Y5=Y4/X4 :: X5=1 ::
 Z1=X4
10640 FOR Z=1 TO Z1 :: CALL
PLOT(X1,Y1):: X1=X1+X5*X3 ::
 Y1=Y1+Y5*Y3 :: NEXT Z :: SU
BEND
```

* * *

TI99-OPOLY UPDATES of FAIRWARE PROGRAM.

Another minor bug has been discovered in the Main program of TI99-OPOLY. The bug will cause the program to terminate in an error if a player attempts to View the non existant property number 0. This change raises the program to Version 1.7; the actual change being the insertion of " J<1 OR " in line 4160 of the extended basic program. Line 4160 should be changed as shown and both the version number and date of revision raised in line 100.

```
100 OPTION BASE 1 :: ON WARNING NEXT :: ON BREAK NEXT !
TI99-OPOLY V1.7  Ross Mudie 6th July 1987

4160 IF J$="" THEN 4260 ELSE J=VAL(J$):: IF J<1 OR J>40
THEN 4150
```

TI99-OPOLY AMERICAN NAMES VERSION.

Version 2.1 of TI99-OPOLY is now available with the American names, in lieu of the English names in Version 1 of the game.
Copies of either version will be available from the author for $10 programmer's fee plus $5 to cover disk, jiffy bag and postage.

People overseas please DO NOT send stamps, or disks in paper or light cardboard mailers. Foreign stamps can not be used in Australia and disks in paper or light cardboard mailers always arrive damaged.

Enquiries to:
Ross Mudie, 47 Berowra Waters Rd, BEROWRA. N.S.W. 2081.
AUSTRALIA.

REVIEW OF AT MINI PE SYSTEM.

by Ross Mudie of TIsHUG, 8th September 1987.

1. INTRODUCTION.

The AT MINI EXPANSION system really lives up to its name, "Advanced Technology". In a small diecast aluminium box which plugs directly into the TI99/4A console, the designer Peter Schubert, has built a 32K memory expansion, PIO port, one fully implemented RS232 port, one partially implemented RS232 port and a disk controller capable of controlling up to four double sided, double density disk drives, either 5.25 or 3.5 inch.

The system which Peter supplied for me to try included a 3.5 inch disk drive in the same sized box as the mini system, powered from a 9 volt plug pack. The disk drive box sat neatly and conveniently on top of the mini PE system.

The Mini PE system is powered from the 5V DC in the console which Peter states is adequately rated to power the mini system as well as the console. I would enter a note of caution here. I would not recommend powering the Mini System from a console fitted with a power supply having only 2 wires for the AC input. These power supplies have a very nasty habit of failing with the switch mode series pass transistor short circuit. This destroys consoles and would destroy the mini system as well. The way to identify such a power supply is to look at the low voltage AC input socket on the rear of the console. If there are only 2 pins, beware!

The Mini System has the PIO, both RS232's and the 32K memory expansion on 1 Printed Circuit Board (PCB), whilst the disk controller is on the other PCB. The PCBs are stacked one on top of the other by means of multi-pin connectors. On the left side of the RS232 board a 44 way connector is provided to plug into the console, whilst on the right side the printed wiring provides a continuation of the expansion bus into which a speech synthesizer may be connected or other peripherals. The PIO has the same type of 16 pin socket as the standard TI RS232 card so when reviewing I just transferred my printer cable to this socket. The RS232 ports are brought out on a 25 pin D connector with the same pinout as the standard TI RS232. This retains compatability with standard TI connections and in my case I just plugged the RS232 cable connector off my normal RS232 card onto the mini system. The disk drive connected onto the disk controller via a 34 way edge connector at the back of the controller card in the same area as the RS232 card connectors. There is a little congestion of cables here but everything fits together well. I could not suggest any better alternative.

The mini system makes the expanded TI99/4A much more portable than a system with the standard TI PE box. The dramatically smaller size and less weight, especially if teamed up with a 3.5 inch disk drive, make it practical to use a single disk system when travelling.

2. THE RS232 CARD.

The PIO port operated quite normally with my BMC BX80 printer, including recognition of the busy line when the printer was off line. The first RS232 port is fully implemented so I tested this out by transfering to and from the standard TI RS232 card. I used OLD RS232 & SAVE RS232 for basic and extended basic programs and LIST RS232. I also tried PF RS232 from the E/A editor of Funnel Writer and all worked fine. RS232/2 is only partly implemented, it provides the receive & transmit data lines only, which is fine for use with a dumb modem. I loaded the terminal emulator off the disk Peter provided, it was Mass Transfer. After configuring it for RS232/2 I rang the TEXPAC BBS and operated the BBS quite normally at its only speed of 300 bauds. I went to Main Menu 2 of the BBS and downloaded a program into extended basic. Later I used the SENDMAIL program to send a file to the BBS. In short normal operation of the BBS via the partially implemented RS232/2 was achieved using my dumb modem.

Additional enhancements in the RS232 EPROM include new baud rates of 50, 19200, 1275 (1200/75) and 7512 (75/1200). All of the speeds are in addition to the normal speeds provided by TI in their RS232 card. Additional file names made available in the new EPROM include VIATEL, VIATEL2, MIDI, MIDI2, RTTY, RTTY2 for serial access and PRINT for parallel.

3. DISK CONTROLLER.

The disk controller adds all the extra calls of the CORCOMP disk controller to the basic console in addition to providing control for up to 4 disk drives for any mixture of single or double sided & single or double density. With double sided double density 1440 256 byte disk sectors are available.

The disk controller software modifies the initial master screen when the computer powers up showing that the new disk controller is attached.

The extra calls are CALL LR, CALL ILR & CALL LLR which load utilities & load option 3 editor/assembler type programs from basic or extended basic. CALL RUN allows loading and running of e/a option 5 program files. The XLIR routines allow access to the above routines from a running extended basic program. CALL WRTRG allows access to the VDP Write Only registers whilst CALL MOVEM allows movement of memory blocks including VDP, CPU RAM and ROM. CALL EXEC allows execution of assembly which is in memory by branching to an absolute address. CALL LINK("MPOKE"), MPEEK, VPOKE, VPEEK allow poking to and peeking at CPU & VDP memory locations. CALL MGR also loads the the Corcomp disk manager. Most of the new calls are shown in the article on pages 17 & 18 of the July 1987 TND.

The controller allows for 4 different disk drive head step times. A group of switches on the disk controller board allow the head step pulse to be set to 6 or 15 milliseconds. By making a strapping change on the PCB the head step can be changed to 3 or 10 mSecs. This range of options should allow almost any 5.25 or 3.5 inch disk drive to be used on the controller.

4. OVERALL OPERATION.

Having a normal TI PE box I think that the thing I really miss is the roar of the fan. The mini system doesn't need a fan since it barely gets warm after after many hours of operation and the openings in the case appear to provide adequate ventilation.

The other thing that I miss is the activity LEDs which are provided in the standard TI system. For normal operation one doesn't need these indicators but they do help one to understand what the computer is doing at times.

5. CONTENTS OF THE DISK.

The Mini System is provided with a disk which provides Disk Management, Funnel Writer text editor & editor/assembler, Disk Copiers, Mass Transfer communications program and some documentation files. It is recommended that when received this disk is backed up promptly.

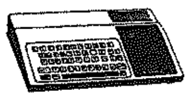
6. PRICING.

The system can be obtained partly or fully equipped. The fully equipped mother board with both RS232s, PIO and 32K memory expansion is $230, compared with the TI RS232 card which cost $160 + 32K card which $199 when I purchased my original system. The PE box cost $200, compared to $35 for the mini system. The mini disk controller is dearer at $190 but compared to $100 but is considerably enhanced. If a purchaser didn't want all the options on the mother board, e.g., the RS232s, then these components could be omitted and the price would be reduced to $135.

Continued on P6

THE COMMUNICATORS

Special Interest Group for Users
of the TEXPAC Bulletin Board Service.
by Ross Mudie, SYSOP, 6th September 1987.



1. BBS USERS LIST.

The users of the BBS have available a list of user's and their suburbs or town. Commencing early October user's names will be added to this list. If any user of the BBS objects to their real name being added to the list of system names (which are used for mail) then please contact the SYSOP on the BBS or write to PO Box 214 Redfern 2016, on or before 6th October 1987. All users who fail to notify objection will be included in the list.

2. MEETING INFORMATION, HELP NEEDED PLEASE.

A file of meetings is maintained on the BBS for both TIsHUG and regional group meetings. This file is also used by the editor of the TIsHUG News Digest for the magazine. Secretaries of the regional groups are requested to advise the SYSOP of meeting dates, times, location and subject to enable this file to remain both up to date and accurate.

3. PAUSING AND ESCAPING MENUS AND LISTINGS.

The BBS alows users to PAUSE or ESCAPE from any menu or file listing.

To PAUSE press <CTRL> S , to UNPAUSE use <CTRL> Q .

To ESCAPE press E once only.

The action to Pause or Escape will take effect at the start of each new line, by pressing the appropiate key more than once or pressing ENTER the Pause or Escape may be ignored by the BBS.

To correct an incorrect key press, use <CTRL> H to perform a backspace which rubs out characters as as it goes. This also works in the password area even when the characters are not echoed.

4. SOFTWARE CHANGE DATE.

The programs and files on the BBS are changed on a monthly basis, around the beginning of each month. All users are requested not to leave getting any program or file off the BBS until the very end of the month as you may miss out if other commitments force me to change the disks earlier than the first of the month.

If you miss out on a particular program or file then you may request it to be placed on the BBS again. It is also possible to put old program or news disks on the BBS for a few hours during some week days. Send mail on the BBS to SYSOP for any such request.

5. SESSION LENGTHS ON THE BBS.

The BBS software calculates the session length, dependant on the time of day and advises users when they should log off. The session times are 60 minutes for calls which start after midnight and before 2pm and 30 minutes for calls which start after 2pm and before midnight. If you want more time on the BBS then please allow a 30 minute break for others to get on the BBS before calling again.

Observance of this request will help share the usage between all users.

6. How to Join the BBS.

Members of TIsHUG pay an additional $5 membership fee to gain access to the BBS. All requests for BBS membership should be sent to the SECRETARY TIsHUG, PO Box 214, Redfern. N.S.W. 2016. You should nominate your preferred user name (a nickname is usually suitable). The minimum system for BBS access is a TI99/4A, TV, TE2 module, RS232 card or stand alone and a Modem, with ready access to your telephone line. If living outside the Sydney metropolitan area, then remember that using the BBS can do wonders for your phone bill! o

From P.5

When I added up the prices I came up with $455 for a new mini system in 1987 compared with $659 for virtually the same facilities in the standard TI system in 1984. Since then I have added a Triple Tech card and RAM disk in the PE box which could not be added to the mini PE box in their present form. It appears to be feasible for the ramdisk to be relaid out using the latest 32Kx8 static RAM chips instead of the existing 8Kx8 in the existing design. (Peter is planning to do a PCB design for a suitable ramdisk which may also include an additional 8K Super Cart memory).

7. OTHER FUTURE DEVELOPMENTS.

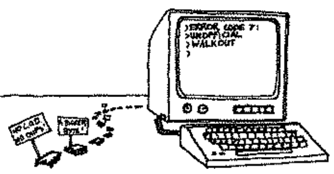
Peter is already working on laying out the enhanced disk controller on a standard sized TI PE box board for people who wish to upgrade from the TI disk controller which only allows 3 drives and single density. The AT disk controller will also add the extra calls to the large PE box system. Other planned enhancements for the mini system include EPROM updates to provide extra facilities and the Forti music system.

8. SUMMARY.

To sum up the mini Peripheral system provides a very viable and cost effective way of expanding a TI99/4A home computer. The system will provide the necessary interface to use a printer, modem and up to four disk drives or interconnection of two computers. The system is compact and I found it worked satisfactorily; this review was totally prepared on a TI99/4A using the mini PE system without any problems.

It is also a good way to provide a portable TI99/4A system.

The acid test is would I buy it myself? If I didn't already have an expanded disk system my answer would be yes! o



Hardware News

A "PIO" PORT For MINI-PE system

By Peter Schubert
6th August 87

The option of a true "PIO" printer port inside the Mini-PE Box is now available with the design of an all-new main PCB (or Motherboard) for the Mini-PE System.
This new board is actually a Multi-function board consisting of the following functions;

32K MEMORY EXPANSION
RS232 PORTS 1 AND 2
PIO PRINTER PORT

It also has the 44 way connection to console and thru-connect for other peripherals, and of coarse, the expansion buss for other Mini-PE boards to attach (such as the amazing little DSDD Disk Control card).
The board is designed so that any single function can be provided, or a combination of functions, or all of them can be provided. A partial board can be expanded over a period of time as the TI budget requires, by returning the board (or Mini-PE) to me for fitting.
The PIO is TI standard in operation and the connector used to attach the printer cable. In addition the "PIO" can also be opened as "PRINT", and because it has a different CRU address of >1400 it can possibly be used with existing TI hardware (or the PIO add on board described in June TND) as a second PIO port).
The RS232 connector is similar to the TI RS232 Card for TI PEB and has both ports in same connector. However this board does not use all connections on port 2. Only transmit and receive data are connected.
The 32K is the new single-chip version(plus some logic) which has proven itself on earlier board, and has now been adopted by the club for console 32k memory kits.
A most important feature of this new board(as with earlier RS232 board) is its ability to run from console power supply (no external power required). I have also provided the option to power it from the regulated supply on the Disk control board when fitted with external DC Plugpak. This may be advisable in the future when the board is complete with all options and another board may be added to the Mini-PE system. A RAMDISK of up to 400K or 1600 sectors has been designed.

PRICES

Mini-PE motherboard (with one of either 32K Memory :: PIO :: RS232 port)	$85

Extra options on motherboard;

RS232	$50
PIO	50
32K	50
RS232/2 (second port)	30
DSDD Disk Control board (complete)	190
Diecast Box (painted and fitted)	35

Mail orders can be sent to the following address;
P.Schubert
P.O.Box 28
Kings Cross
2011 N.S.W.

Or to the Club Shop address.
Please dont forget to allow for cost of postage. Also I would prefer that you supply your own box if possible as I dont have enough time to do these. Fitting instructions are included as well as some info on RS232 and PIO connection details.

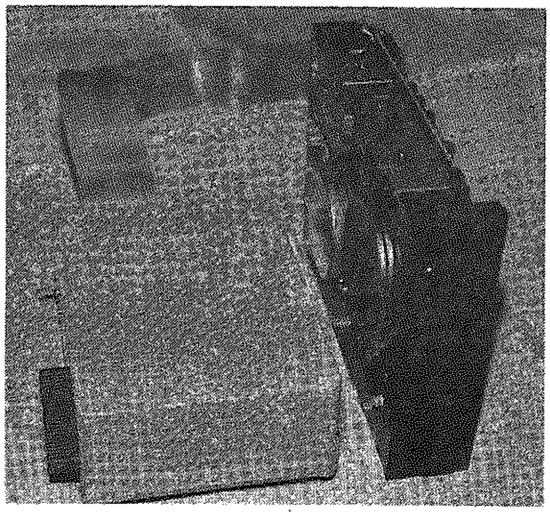
MINI-PE DISK CONTROL BOARD

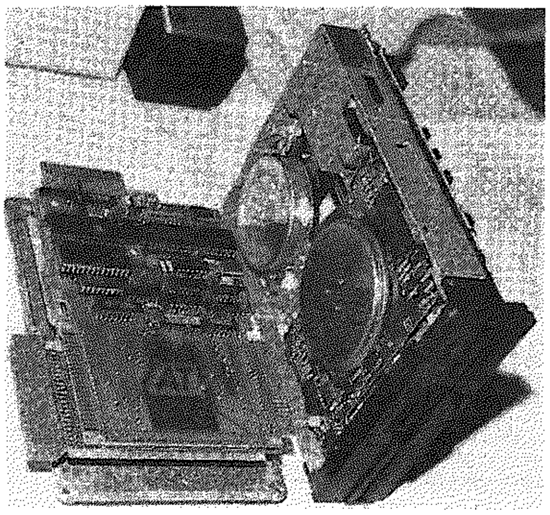
I have finished the first lot of boards, and most of these have been sold. An order for more PCB's has been placed so more will be available to cope with the response to this product. I have had some feedback from users of Mini-PE systems and results have been excellent.One installation I have seen has the Mini-PE mounted inside a home-made PE box with 2 drives and a Ribbon cable (unshielded!!!) connecting to console. All is powered from PE box (including console). Very good work!
If you would like to see a Mini-PE system, or have it demonstrated give me a call on (02) 358 5602

NEW PE BOX CARD

I will be making a new card for those of us who have a TI PE BOX. It will be a true multi-function card designed to "make room" in the box by providing on one card the functions of three TI cards. Great! room for more RAMDISKS I hear you say.
The following will be provided either singly, or together as desired;
32K MEMORY
RS232 ports 1 and 2
PIO port
DSDD Disk Controller

The last also gives you the chance to upgrade to double density and so save heaps of disks.
Watch out for this one soon. o





Jenny's Younger Set.

SPRITE CIRCLES

Here is a program that will make a sprite move in a circle by plotting the sprit around the circle's circumference. By altering the variable R (radius), you can increase or decrease the size of the circle. Be careful not make the radius too big - 32 is as large as I would go - as bad values can occur. A small circle, say of radius 3 or 4, will produce a good "hovering effect with a larger sprite pattern than that provided, and that might be useful for a ghost etc in a game.

```
100 CALL CLEAR :: CALL SCREEN(2)
:: CALL CHAR(128,"60F0F060")
110 FOR D=1 TO 99999 STEP 8
120 A=D*(22/7)/180
130 R=18
140 X=R*COS(A)+138
150 Y=R*SIN(A)+96
160 CALL SPRITE(#1,128,6,Y,X)
170 NEXT D :: GOTO 110
```

You will notice that the circle is not perfectly smooth. You can obtain a smoother circle by decreasing the step rate in line 110. This however will slow down the program. Similarly, by increasing the step rate, the program will speed up, but the circle will be less smooth. This can be partially improved by then decreasing the circle's radius. The co-ordiates of the centre of the circle are 96 down by 138 across. This of course can be altered to suit the direction of rotation and can be changed by swapping over COS and SIN in lines 140 and 150.

By making various alterations to the program, you can produce some quite interesting results. Here are some examples of the experiments that I have tried:

To get a figure eight, try the following:

```
130 X=R*SIN(A)+138
140 Y=R*SIN(2*A)+96
```

To obtain a vertical elipse, alter line 130 to read:

```
X=2*R*COS(A)+138
```

The value "2" can be altered to vary the height and width of the elipse. For a horizontal elipse, change line 140 to read:

```
Y=2*R*SIN(A)+96
```

Again alter the value "2" to vary the shape of the elipse. A three leaf clover can be obtained with the following:

```
130 X=20*COS(A)*COS(3*A)+138
140 Y=20*SIN(A)*COS(3*A)+96
```

More leaves can be added by increasing "3" in both lines to a higher number. NB: a value of "4" doesn't give 4 leaves, neither does "5" etc.

Try these:

```
130 X=R*COS(2*A)+138  or

130 X=R*COS(3*A)+138  or

130 X=R*COS(A)+138
140 Y=R*COS(2*A)+96  or

130 X=R*COS(.5*A)+138
140 Y=R*COS(2*A)+96
```

Please note that another change can be made to this program which involves putting the values X and Y into a CALL MOTION statement. The alterations are as follows:

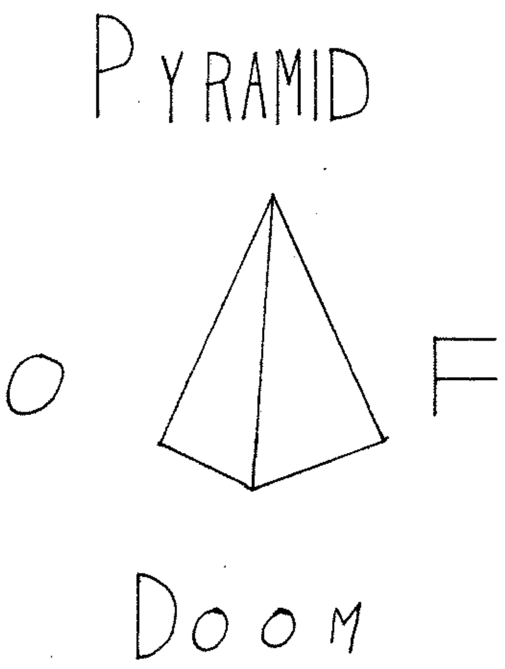
```
105 CALL SPRITE(#1,188,6,96,96)
160 CALL MOTION(#1,Y,X)
```

I will leave it to you to decide which method produces the best results, but I should mention that for a sprite to maintain circular motion (actually moving, not just plotted) it has to undergo continual changes in direction, and thus the result is a bit jerky. Anyhow, you can decide which method you would prefer.

Happy Computing

Joshua Rust

* * *



In this adventure, one is in ancient Egypt to explore a pyramid with 13 treasures (an emerald bracelet just one of them). but first you must get inside .... dig a hole in front of the pyramid then take it from there. Once inside there is a mummy and burning leaves (hint: CANTEEN-BURNING LEAVES), then to the north a brick wall and to the South a tall room with a bar (hint: find ALCOVE). Up in the revolving room in the South a ledge (hint: ROPE) with a hole in the ceiling and to the North a prison cell (hint: DON'T STAY THERE TOO LONG) EXTRA HINT: BEFORE you find the ladder LOOK AT THE SKULL otherwise it's just too late then and you will have to do the whole thing again.

Just one more hint: KEEP SAVING THE GAME AS YOU GO ALONG.

Vincent Maker

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$="X" you are telling the computer to "go to the box labeled A$ and put the character "X" in it". Or, more accurately, "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 practical. 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 multidimensional 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 computer 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 mailing 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 computer 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 accomplish 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 increases. 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 sorting routine which is very fast for a completely random 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. Continued on P10

From P9

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 alphabetic 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 capitalized initial letters...!

For the same reason, if you perform an alphabet sort of strings containing numeric 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 multidimension array becomes a very complex 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)
```

ANALYSIS OF SORTING ROUTINES

by Jim Peterson

Number of value changes made is shown above the number of value comparisons made. All sorts were made on the same portions of the same random array.

Number of records - 10 to 100

	10	20	30	40	50	60	70	80	90
BUBBLE									
	208	316	1018	1868	3218				
	52	182	450	805	1269				
SHAKER									
	109	311	1009	1855	3195				
	55	152	417	755	1172				
SWAP									
	52	130	207	286	354	457	539	642	
	54	209	464	819	1274	1829	2484	3239	
SHUTTLE									
	73	224	735	1360	2357				
	27	83	260	475	813				
EASY									
	102	323	611	979	1297				
	45	137	254	407	538				
QUICK									
	121	318	480	653	816	1032			
	108	292	440	613	764	969			
RESORT									
	43	120	317	552	911	1197			
	30	87	264	479	818	1084			
SHELL									
	35	109	206	351	557	633	691	857	1071
	30	92	150	226	364	422	485	581	683
WAZZIT?									
	59	184	345	578	775	1005			
	55	210	465	820	1275	1830			
INSERT									
	49	126	323	558	917	1203			
	21	68	235	440	769	1025			

Observations: the Wazzit? sort is one that I wrote, but I presume it has been done before under some other name. Some others of these may also be known under other names. The popular Bubble Sort is obviously the least efficient of them all, even for small arrays. The Quick Sort is not very quick. The Shell Sort is by far the best general-purpose sort when the file may be of any length and degree of randomness.

ANALYSIS OF SORT OF 5 DIFFERENT RANDOM ARRAYS OF 20 RECORDS BY 10 DIFFERENT SORTING ROUTINES.

- by Jim Peterson

Number of value changes is shown above number of value comparisons.

	1	2	3	4	5
BUBBLE					
	572	530	564	476	316
	204	174	209	165	182
SHAKER					
	467	433	455	379	311
	194	185	194	175	152
SWAP					
	129	117	132	123	131
	209	209	209	209	209
SHUTTLE					
	335	311	326	272	224
	121	113	118	99	83
EASY					
	318	334	371	392	323
	130	139	155	164	137
QUICK					
	298	298	267	290	318
	274	272	244	266	292
RESORT					
	155	151	154	140	120
	124	116	121	103	87
SHELL					
	142	118	139	127	109
	99	91	99	97	92
WAZZIT?					
	151	169	148	166	184
	210	210	210	210	210
INSERT					
	163	155	160	142	126
	105	97	102	84	68

OBSERVATIONS: The speed of a sort depends on the degree of randomness of the file, or by the distance that each record is from its correct position, but some sorting routines are less affected by this than others.

ANALYSIS OF RESORTING A SORTED ARRAY OF 50 RECORDS AFTER THE FIRST RECORD WAS CHANGED TO ZZZ OR THE LAST RECORD WAS CHANGED TO AAA OR THE CENTER RECORDS WERE CHANGED TO ZZZ AND AAA.

	ZZZ in 1st RECORD	AAA IN LAST RECORD	ZZZ/AAA IN MIDDLE
BUBBLE			
	199	295	242
	99	1275	950
SHAKER			
	202	204	201
	100	149	149
SWAP			
	246	246	243
	1274	1274	1274
SHUTTLE			
	148	148	148
	97	97	97
EASY			
	728	731	740
	342	345	347
QUICK			
	747	630	781
	686	638	759
RESORT			
	148	52	101
	98	98	98
SHELL			
	154	154	155
	270	271	272
WAZZIT?			
	198	198	199
	1275	1275	1275
INSERT			
	148	148	149
	49	49	49

OBSERVATIONS: The simple sorting routines may be better than the more complex and faster ones, for resorting a presorted file after a few changes have been made, or for adding a few records to an existing presorted file.

ANALYSIS OF ARRAY CAPACITY IN TI EXTENDED BASIC

by Jim Peterson

NUMERIC

NO. OF DIMENS	MAXIMUM DIMIMUM ACCEPTED	TOTAL RECORDS	LEAVING BYTES FREE	BYTES PER RECORD	BYTES FREE AFTER RUN
1	3050	3050	64	8.07	20
2	54,54	2916	269	8.3	202
3	14,13,13	2366	944	9.95	852
4	7,6,6,6	1512	2511	14.535	2398
5	4,4,4,4,3	768	4459	26	4322
6	3,3,3,3,2,2	324	6023	56.99	5862
7	3,2,2,2,2,2,2	192	1123	121.69	938

STRINGS - MAXIMUM ACCEPTED BUT NOT RUNNABLE

1	5900	5900	28	2.002	
2	76,75	5700	124	2.055	
3	18,17,16	4896	198	2.377	
4	8,8,8,7	3583	160	3.259	
5	5,5,5,5,4	2000	1022	5.409	
6	4,3,3,3,3,3	972	1580	10.555	
7	3,3,3,2,2,2,2	432	1450	24.05	

STRINGS - MAXIMUM RUNNABLE FOR 1-BYTE RECORDS

1	1682	1682	8464	7.009	50
2	41,40	1640	8384	7.118	166
3	12,11,11	1452	8082	7.606	795
4	6,6,6,6	1296	7022	8.75	497
5	4,4,4,4,4	1024	5572	11.17	398
6	3,3,3,3,3,2	486	5676	17.81	3183
7	3,3,2,2,2,2,2	288	4042	32.295	2539

REMARKS - Any array of more than 3 dimensions will cra with MEMORY FULL if it is not DIMensioned, even A(1,1,1,1)=1.

String arrays can be dimensioned for many more records than they will actually hold.

Return To Pirate's Isle.

Chapter 3

Clues for this month are:
Drop glass, crawl, crawl, crawl east, squint, get clock, get sign, crawl west, crawl west, drop clock, drop sign, wear glass, get map, get screw, get oyster, get mask, go pool, hold breath, swim down, swim east, swim up, go boat, get blade, go dock, wear glass, go north, dig down, drop blade,go hill, jump up, look pirate, look pirate, go crack, go shed, get hammer, go north, go crack, give rum, wake pirate, jump ledge, go down, go east, go sea, hold breath, swim down, feel silt, swim up, wear glass, go beach, take rock, look rock, look algae, drop rock, drop algae, unscrew lens.

With the school holidays upon us there should soon be the cries that the adventure is now complete. The clues so far have you about halfway through.

If any completes the adventure from this point please sent in your solution so the rest can finish.

BUG IN RS232/SA

A bug has crept into the Enhanced DSR ROM for the stand-alone RS232 as used in the MINI-PE Box. The problem appears when using the Command 'OLD RS232' and it fails to work. If you have this problem return the RS232 unit to me and I will r eplace the ROM (no charge of course). The address is;

P.Schubert
Box 28
Kings Cross 2011
Ph.358 5602

PLEASE NOTE!

I will NOT be at the October Meeting. If unsure which ROM version you have try downloading from TEXPAC, or a friends computer, or try the command 'OPEN #1:'VIATEL'. If it accepts this or 'MIDI', or 'RTTY', then its the one affected. CLOSING #1:"PETESAKE"

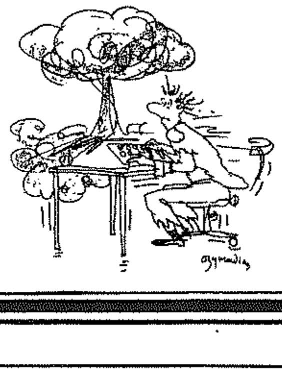
"And this one has 5 languages — Pascal, Basic, Fortran, Cobol and Occa."

```
100 CALL CLEAR
110 CALL SCREEN(3)
120 DISPLAY AT(2,12):"ASTRON
OMY"
130 INPUT "PRESS ENTER TO CO
NTINUE":A$
140 CALL CLEAR
150 DISPLAY AT(1,1):"WHICH P
LANET?"
160 DISPLAY AT(3,1):"1  MERC
URY"
170 DISPLAY AT(5,1):"2  VENU
S"
180 DISPLAY AT(7,1):"3  EART
H"
190 DISPLAY AT(9,1):"4  MARS
"
200 DISPLAY AT(11,1):"5  JUP
ITER"
210 DISPLAY AT(13,1):"6  SAT
URN"
220 DISPLAY AT(15,1):"7  URA
NUS"
230 DISPLAY AT(17,1):"8  NEP
TUNE"
240 DISPLAY AT(19,1):"9  PLU
TO"
250 DISPLAY AT(21,1):"10 EXI
T TO NEXT PROGRAM"
260 INPUT "WHICH PLANET?  ":
P
270 IF P=6 THEN GOTO 380
280 IF P=5 THEN GOTO 490
290 IF P=8 THEN GOTO 680
300 IF P=9 THEN GOTO 820
310 IF P=7 THEN GOTO 960
320 IF P=1 THEN GOTO 1090
330 IF P=2 THEN GOTO 1260
340 IF P=4 THEN GOTO 1420
350 IF P=3 THEN GOTO 1570
360 IF P=10 THEN GOTO 1830
370 INPUT "WHICH PLANET?  ":
P
380 CALL CLEAR
390 PRINT "SATURN": : :
400 PRINT "SHEETS OF ICE OR
FROSTED       "
410 PRINT "GRAVEL WHIRL AROU
ND SATURN'S EQUATOR IN A RIN
G 42,000"
420 PRINT "MILES WIDE.THESE
BANDS ARE  ONLY FEW INCHES T
HICK;THE"
430 PRINT "INNERMOST ONE IS
TOO THIN TOBE SEEN.THE COLOR
ED STREAKS ON THE PLANET'S S
URFACE ARE"
440 PRINT "ATMOSPHERIC BELTS
 LIKE       JUPITER'S;THE WID
E SHADOW ISCAST BY THE RINGS
.A THOUSAND"
450 PRINT "TIMES AS BIG AS T
HE EARTH,   SATURN IS MADE O
F SUCH      LOOSELY PACKED MA
TTER THAT"
460 PRINT "IT COULD FLOAT IN
 WATER.": :
470 INPUT "PRESS ENTER TO CO
NTINUE":A$
480 GOTO 140
490 CALL CLEAR
500 PRINT "JUPITER": : :
510 PRINT "THE LARGEST PLANE
T,IS MORE  THAN 11 TIMES AS
BROAD AS    "
520 PRINT "THE EARTH.IT IS S
O COLD THAT"
530 PRINT "ITS ATMOSPHERE OF
 POISONOUS"
540 PRINT "GASES CONTAINS CL
OUDS OF    AMMONIA CRYSTALS.
THIS"
550 PRINT "ATMOSPHERE,WHICH
MAY BE A"
560 PRINT "VIOLENT ONE OF TO
RRENTIAL"
570 PRINT "RAINS AND TITANIC
 BLIZZARDS,"
580 PRINT "IS MARKED BY SHIF
TING"
590 PRINT "COLORED BANDS.THE
 MOST"
600 PRINT "FAMOUS MARKING IS
 THE GREAT"
610 PRINT "RED SPOT,THE SPOT
 CHANGES"
620 PRINT "COLOR FROM RED TO
 GRAY AND"
630 PRINT "SOMETIMES DISAPPE
ARS FOR"
640 PRINT "YEARS AT A TIME."
650 PRINT : :
660 INPUT "PRESS ENTER TO CO
NTINUE":A$
670 GOTO 140
680 CALL CLEAR
690 PRINT "NEPTUNE": : :
700 PRINT "THE PLANET BEYOND
 URANUS,IS"
710 PRINT "NEARLY 2.8 BILLIO
N MILES"
720 PRINT "FROM THE SUN.ASTR
ONOMERS"
730 PRINT "HAVE NOTICED IRRE
GULARITIES IN"
740 PRINT "THE PLANET'S ORBI
T."
750 PRINT "NEPTUNE IS A PALE
 GREEN ORB,CIRCLING THE"
760 PRINT "SUN ONCE EVERY 16
6 YEARS."
770 PRINT "TWO SATELLITES TR
AVEL WITH NEPTUNE, ONE OF WH
ICH"
780 PRINT "NAMED TRITON"
790 PRINT : :
800 INPUT "PRESS ENTER TO CO
NTINUE":A$
810 GOTO 140
820 CALL CLEAR
830 PRINT "PLUTO": : :
840 PRINT "THE NINTH PLANET,
FOUND IN"
850 PRINT "1930,KNOWN AS A T
INY SPHERE,"
860 PRINT "IT ORBIT ECCENTRI
CALLY"
870 PRINT "BETWEEN 4.6 AND 2
.7 BILLION"
880 PRINT "MILES FROM THE SU
N.ITS YEAR"
890 PRINT "IS 248 EARTH YEAR
S,ITS"
900 PRINT "TEMPERATURE IS -3
75'F."
910 PRINT "AND 35 MILLION MI
LES CLOSER"
920 PRINT "TO THE SUN THAN N
EPTUNE."
930 PRINT : :
940 INPUT "PRESS ENTER TO CO
NTINUE":A$
950 GOTO 140
960 CALL CLEAR
970 PRINT "URANUS": :
980 PRINT "WHOSE ATMOSPHERE
IS MADE"
990 PRINT "UP PRIMARILY OF M
ETHANE"
1000 PRINT "IS 14.5 TIMES AS
 MASSIVE AS"
1010 PRINT "THE EARTH.ITS TE
MPERATURE IS"
1020 PRINT "AT LEAST 270'F.
BELOW ZERO."
1030 PRINT "ITS YEAR IS 84 E
ARTH YEARS"
1040 PRINT "LONG,AND IT ROTA
TES EVERY 10"
1050 PRINT "HOURS AND 49 MIN
UTES."
1060 PRINT : :
1070 INPUT "PRESS ENTER TO C
ONTINUE":A$
1080 GOTO 140
1090 CALL CLEAR
1100 PRINT "MERCURY": :
1110 PRINT "THE SMALLEST PLA
NET AND THE"
1120 PRINT "CLOSEST TO THE S
UN,COMES"
1130 PRINT "WITHIN 28 MILLIO
N MILES OF"
1140 PRINT "THE SUN,THE PERI
OD OF EACH"
1150 PRINT "REVOLUTION IS EQ
UAL TO ONLY"
1160 PRINT "88 DAYS ON EARTH
.MERCURY'S"
1170 PRINT "EQUATOR RECEIVES
 SUNLIGHT"
1180 PRINT "FOR 90 CONSECUTI
VE EARTH"
1190 PRINT "DAYS,THE SIDE FA
CING THE SUN"
1200 PRINT "IS BEING COOKED
TO"
1210 PRINT "TEMPERATURES THA
T MAY"
1220 PRINT "APPROACH 1,000'F
."
1230 PRINT : :
1240 INPUT "PRESS ENTER TO C
ONTINUE":A$
1250 GOTO 140
1260 CALL CLEAR
1270 PRINT "VENUS": :
1280 PRINT "IT IS A FASCINAT
ING AND"
1290 PRINT "MYSTERIOUS WORLD
.VENUS"
1300 PRINT "COMPLETED A ROTA
TION ONLY"
1310 PRINT "ONCE IN EVERY 24
3 EARTH"
1320 PRINT "DAYS.IT TAKES VE
NUS 224.7"
1330 PRINT "EARTH DAYS TO GO
 AROUND THE SUN,"
1340 PRINT "ITS MASS IS 81 P
ERCENT,ITS"
1350 PRINT "VOLUME 88 PERCEN
T,ITS"
1360 PRINT "DENSITY 93 PERCE
NT AND ITS"
1370 PRINT "ESCAPE VELOCITY
92 PERCENT"
1380 PRINT "THAT OF THE EART
H."
1390 PRINT : :
1400 INPUT "PRESS ENTER TO C
ONTINUE":A$
1410 GOTO 140
1420 CALL CLEAR
1430 PRINT "MARS": :
1440 PRINT "141 MILLION MILE
S FROM THE"
1450 PRINT "SUN,TEMPERATURE
IS ABOUT"
1460 PRINT "-60'F.A YEAR ON
MARS IS 687 AND 1 HALF EARTH
 DAYS LONG"
1470 PRINT "AND ITS' DAY LAS
TS A LITTLE"
```

```
1480 PRINT "OVER 24 AND 1 HA
LF EARTH"
1490 PRINT "HOURS.MARS HAS T
WO TINY"
1500 PRINT "SATELLITES OR MO
ONS,PHOBOS"
1510 PRINT "AND DEIMOS,ONE 1
0 AND THE"
1520 PRINT "OTHER FIVE MILES
 IN"
1530 PRINT "DIAMETER."
1540 PRINT : :
1550 INPUT "PRESS ENTER TO C
ONTINUE":A$
1560 GOTO 140
1570 CALL CLEAR
1580 PRINT "EARTH": :
1590 PRINT "DIAMETER IN MILE
S IS 7,926"
1600 PRINT "MEAN DISTANCE FR
OM THE SUN"
1610 PRINT "IN MILLIONS OF M
ILES IS 93"
1620 PRINT "THE TIME TAKEN F
OR THE EARTH"
1630 PRINT "TO GO ONCE ROUND
 THE SUN,IS"
1640 PRINT "365 DAYS,THE EAR
THS AXIS"
1650 PRINT "POINTS TO THE NO
RTH"
1660 PRINT "CELESTIAL POLE,R
OUGHLY"
1670 PRINT "MARKED BY POLARI
S.EARTH:"
1680 PRINT "MAXIMUM 94,600,0
00 MILES;"
1690 PRINT "MEAN,93,000,000
MILES;"
1700 PRINT "MINIMUM,91,400,0
00 MILES."
1710 PRINT "THE MEAN DISTANC
E BETWEEN"
1720 PRINT "THE EARTH AND SU
N,KNOWN AS"
1730 PRINT "THE ASTRONOMICAL
 UNIT,HAS"
1740 PRINT "RECENTLY BEEN RE
MEASURED BY"
1750 PRINT "RADAR AND RADIO
ASTRONOMY"
1760 PRINT "METHODS;THE LATE
ST VALUE IS"
1770 PRINT "APPROXIMATELY 92
,956,000"
1780 PRINT "MILES."
1790 PRINT :
1800 INPUT "PRESS ENTER TO C
ONTINUE":A$
1810 GOTO 140
1820 CALL CLEAR
1830 CALL CLEAR
1840 CALL SCREEN(7)
1850 DISPLAY AT(2,12):"STARS
"
1860 INPUT "PRESS ENTER TO C
ONTINUE":A$
1870 CALL CLEAR
1880 DISPLAY AT(1,1):"WHICH
OPTION?"
1890 DISPLAY AT(3,1):"1  NUM
BERS OF STARS"
1900 DISPLAY AT(5,1):"2  DIS
TANCES OF STARS"
1910 DISPLAY AT(7,1):"3  STA
RLIGHT"
1920 DISPLAY AT(9,1):"4  STA
R BRIGHTNESS"
1930 DISPLAY AT(11,1):"5  ST
AR SIZE"
1940 DISPLAY AT(13,1):"6  DE
NSITY OF STARS"
1950 DISPLAY AT(15,1):"7  MO
TIONS OF STARS"
1960 DISPLAY AT(17,1):"8  CO
LOR OF STARS"
1970 DISPLAY AT(19,1):"9  ST
AR MAGNITUDES"
1980 DISPLAY AT(21,1):"10  E
ND PROGRAM"
1990 INPUT "WHICH OPTION?  "
:P
2000 IF P=1 THEN GOTO 2110
2010 IF P=2 THEN GOTO 2230
2020 IF P=3 THEN GOTO 2460
2030 IF P=4 THEN GOTO 2560
2040 IF P=5 THEN GOTO 2730
2050 IF P=6 THEN GOTO 2920
2060 IF P=7 THEN GOTO 3100
2070 IF P=8 THEN GOTO 3310
2080 IF P=9 THEN GOTO 3440
2090 IF P=10 THEN GOTO 3590
2100 INPUT "WHICH OPTION?  "
:P
2110 CALL CLEAR
2120 PRINT "NUMBER OF STARS"
: : :
2130 PRINT "ON THE CLEAREST
NIGHT YOU"
2140 PRINT "ARE NOT LIKELY T
O SEE"
2150 PRINT "MORE THAN 2,000
STARS."
2160 PRINT "WITH CHANGING SE
ASONS,"
2170 PRINT "NEW STARS APPEAR
,BRINGING"
2180 PRINT "THE TOTAL VISIBL
E DURING"
2190 PRINT "THE YEAR TO ABOU
T 6,000."
2200 PRINT : : :
2210 INPUT "PRESS ENTER TO C
ONTINUE":A$
2220 GOTO 1870
2230 CALL CLEAR
2240 PRINT "DISTANCES OF STA
RS": : :
2250 PRINT "THE NEAREST STAR
,OUR SUN,"
2260 PRINT "IS A MERE 93 MIL
LION MILES"
2270 PRINT "AWAY.THE NEXT NE
AREST STAR"
2280 PRINT "IS 26 MILLION MI
LLION"
2290 PRINT "MILES-NEARLY 300
,000 TIMES"
2300 PRINT "FARTHER THAN THE
 SUN."
2310 PRINT "FOR THESE GREAT
DISTANCES,"
2320 PRINT "MILES ARE NOT A
GOOD"
2330 PRINT "MEARSURE.INSTEAD
,THE LIGHT"
2340 PRINT "YEAR IS OFTEN US
ED."
2350 PRINT "THIS IS THE DIST
ANCE THAT"
2360 PRINT "LIGHT TRAVELS IN
 ONE YEAR,"
2370 PRINT "MOVING AT 186,00
0 MILES PER"
2380 PRINT "SECOND: NEARLY 6
 MILLION"
2390 PRINT "MILLION MILES.ON
 THIS SCALE"
2400 PRINT "THE NEAREST STAR
"
2410 PRINT "(EXCLUDING THE S
UN) IS"
2420 PRINT "4.3 LIGHT YEARS
AWAY."
2430 PRINT :
2440 INPUT "PRESS ENTER TO C
ONTINUE":A$
2450 GOTO 1870
2460 CALL CLEAR
2470 PRINT "STARLIGHT": :
2480 PRINT "ALL STARS SHINE
BY THEIR"
2490 PRINT "OWN LIGHT.THIS L
IGHT MAY"
2500 PRINT "BE PRODUCED BY N
UCLEAR"
2510 PRINT "REACTIONS SIMILA
R TO"
2520 PRINT "THOSE OF THE HYD
ROGEN BOMB."
2530 PRINT : :
2540 INPUT "PRESS ENTER TO C
ONTINUE":A$
2550 GOTO 1870
2560 CALL CLEAR
2570 PRINT "STAR BRIGHTNESS"
: :
2580 PRINT "THE SUN IS ABOUT
 AVERAGE IN"
2590 PRINT "SIZE AND BRIGHTN
ESS.SOME"
2600 PRINT "STARS ARE UP TO
600,000"
2610 PRINT "TIMES AS BRIGHT
AS THE SUN;"
2620 PRINT "OTHERS ARE ONLY
1/550,000;"
2630 PRINT "MOST ARE BETWEEN
 10,000 AND"
2640 PRINT "1/10,000 TIMES A
S BRIGHT"
2650 PRINT "AS THE SUN.THE B
RIGHTNESS"
2660 PRINT "OF A STAR YOU SE
E DEPENDS"
2670 PRINT "ON ITS DISTANCE
AND ON ITS"
2680 PRINT "REAL OR ABSOLUTE
"
2690 PRINT "BRIGHTNESS."
2700 PRINT : :
2710 INPUT "PRESS ENTER TO C
ONTINUE":A$
2720 GOTO 1870
2730 CALL CLEAR
2740 PRINT "STAR SIZE": :
2750 PRINT "MOST STARS ARE S
O DISTANT"
2760 PRINT "THAT THEIR SIZE
CAN ONLY"
2770 PRINT "BE MEARSURED IND
IRECTLY."
2780 PRINT "CERTIAN GIANT RE
D STARS"
2790 PRINT "ARE THE LARGEST.
ANTARES"
2800 PRINT "HAS A DIAMETER 3
90 TIMES"
2810 PRINT "THAT OF THE SUN,
OTHERS"
2820 PRINT "EVEN LARGER.AMON
G THE"
2830 PRINT "SMALL STARS ARE
WHITE"
2840 PRINT "DWARFS,NO LARGE
THAN"
2850 PRINT "PLANETS.THE SMAL
LEST ARE"
2860 PRINT "NEUTRON STARS TH
AT MAY"
2870 PRINT "BE NO MORE THAN
TEN MILES"
2880 PRINT "ACROSS."
2890 PRINT : :
2900 INPUT "PRESS ENTER CONT
INUE":A$
```

```
2910 GOTO 1870
2920 CALL CLEAR
2930 PRINT "DENSITY OF STARS
": :
2940 PRINT "THE DENSITIES OR
 RELATIVE"
2950 PRINT "WEIGHTS OF STARS
 VARY"
2960 PRINT "CONSIDERABLY.ACT
UALLY ALL"
2970 PRINT "STARS ARE MASSES
 OF GAS-"
2980 PRINT "BUT GAS UNDER VE
RY"
2990 PRINT "DIFFERENT CONDIT
IONS FROM"
3000 PRINT "THOSE WE USUALLY
 SEE.GIANT"
3010 PRINT "STARS SUCH AS AN
TARES HAVE"
3020 PRINT "A DENSITY AS LOW
 AS"
3030 PRINT "1/2000 OF THE DE
NSITY OF"
3040 PRINT "AIR.THE MORE USU
AL STARS"
3050 PRINT "HAVE A DENSITY F
AIRLY"
3060 PRINT "CLOSE TO THAT OF
 THE SUN."
3070 PRINT :
3080 INPUT "PRESS ENTER TO C
ONTINUE":A$
3090 GOTO 1870
3100 CALL CLEAR
3110 PRINT "MOTIONS OF STARS
": :
3120 PRINT "OUR SUN IS MOVIN
G ABOUT"
3130 PRINT "12 MILES PER SEC
OND"
3140 PRINT "TOWARD THE CONST
ELLATION"
3150 PRINT "HERCULES.OTHER S
TARS ARE"
3160 PRINT "MOVING TOO,AT SP
EEDS UP"
3170 PRINT "TO 30 MILES PER
SECOND OR"
3180 PRINT "FASTER.ARCTURUS
TRAVELS"
3190 PRINT "AT 84 MILES PER
SECOND."
3200 PRINT "MANY STARS ARE M
OVING AS"
3210 PRINT "PARTS OF SYSTEMS
 OR"
3220 PRINT "CLUSTERS.ONE SUC
H SYSTEM,"
3230 PRINT "INCLUDING STARS
IN TAURUS"
3240 PRINT "IS MOVING AWAY A
T ABOUT"
3250 PRINT "30 MILES PER SEC
OND.SOME"
3260 PRINT "STARS CONSIST OF
 TWO"
3270 PRINT "OR MORE COMPONEN
TS."
3280 PRINT :
3290 INPUT "PRESS ENTER TO C
ONTINUE":A$
3300 GOTO 1870
3310 CALL CLEAR
3320 PRINT "COLOR OF STARS":
 : :
3330 PRINT "VARIES FROM BRIL
LIANT"
3340 PRINT "BLUE-WHITE TO DU
LL"
3350 PRINT "REDDISH,INDICATI
NG STAR"
```

```
3360 PRINT "TEMPERATURE.A FA
CTOR IN"
3370 PRINT "STAR CLASSIFICAT
ION.CLOSE"
3380 PRINT "OBSERVATION IS N
EEDED TO"
3390 PRINT "SEE THE RANGE OF
 COLORS"
3400 PRINT "IN THE NIGHT SKY
."
3410 PRINT :
3420 INPUT "PRESS ENTER TO C
ONTINUE":A$
3430 GOTO 1870
3440 CALL CLEAR
3450 PRINT "STAR MAGNITUDES"
: :
3460 PRINT "BRIGHTNESS OF ST
ARS IS"
3470 PRINT "MEASURED IN TERM
S OF"
3480 PRINT "MAGNITUDE.A 2ND-
MAGNITUDE"
3490 PRINT "STAR IS  2.5 TIM
ES AS"
3500 PRINT "BRIGHT AS A 3RD,
AND SO ON"
3510 PRINT "THROUGHOUT THE S
CALE,SO"
3520 PRINT "THAT A 1ST-MAGNI
TUDE STAR"
3530 PRINT "IS 100 TIMES AS
BRIGHT"
3540 PRINT "AS A 6TH.STARS B
RIGHTER THAN 1ST MAGNITUDE H
AVE ZERO"
3550 PRINT "OR MINUS MAGNITU
DE."
3560 PRINT : :
3570 INPUT "PRESS ENTER TO C
ONTINUE":A$
3580 GOTO 1870
3590 CALL CLEAR
3600 PRINT "GOOD LUCK IN THE
 TWILIGHT"
3610 PRINT "ZONE.!!"
3620 END
```



```
100 CALL CLEAR
110 PRINT "          *RE-AC
T*"
120 PRINT : : : : :
130 PRINT "THE OBJECT IS TO
REACH THE":"INNER REACTOR DO
OR"
140 PRINT
150 PRINT "AVOIDING THE SECU
RITY ROBOT": :
160 PRINT "IF HE CATCHES YOU
 THEN YOU":"ARE DEAD": :
170 PRINT "WHEN HE RUNS EVER
YTHING":"ELSE SHUTS DOWN FOR
 A WHILE": :
180 CALL SOUND(100,1109,0)
190 FOR D=1 TO 2000
200 NEXT D
210 CALL CLEAR
220 PRINT "USE CURSOR KEYS T
O REACH":"REACTOR BEFORE TIM
E RUNS":"OUT AND IT EXPLODES
"
230 PRINT
240 PRINT "SECURITY HAS GONE
 WILD": :"ALL THE SYSTEMS AR
E OUT":"OF SYNCHRONIZATION A
ND THE":
250 PRINT "INNER DOORS CLOSE
 AND OPEN":"AT RANDOM"
260 PRINT : : :
270 PRINT "         *GOOD-LU
CK*"
280 CALL SOUND(200,1109,0)
290 FOR D=1 TO 2000
300 NEXT D
310 CALL CLEAR
320 R=24
330 C=16
340 RR=4
350 CC=4
360 REM SET SCREEN
370 CALL SCREEN(16)
380 CALL CHAR(40,"FFFFFFFFFF
FFFFFF")
390 CALL CHAR(97,"FFC3A59999
A5C3FF")
400 CALL CHAR(120,"FFFFFFFFF
FFFFFFF")
410 CALL CHAR(112,"0000081C2
41C1422")
420 CALL CHAR(104,"1C1C083E2
41C1422")
430 CALL CHAR(105,"002200999
90022")
440 CALL HCHAR(1,1,40,32)
450 CALL HCHAR(24,1,40,32)
460 CALL VCHAR(1,1,40,24)
470 CALL VCHAR(1,32,40,24)
480 CALL HCHAR(3,3,40,28)
490 CALL HCHAR(22,3,40,28)
500 CALL VCHAR(3,3,40,20)
510 CALL VCHAR(3,30,40,20)
520 CALL HCHAR(5,5,40,24)
530 CALL HCHAR(20,5,40,24)
540 CALL VCHAR(5,5,40,15)
550 CALL VCHAR(5,28,40,15)
560 CALL HCHAR(10,15,40,7)
570 CALL HCHAR(14,15,40,7)
580 CALL HCHAR(11,15,40,2)
590 CALL HCHAR(12,15,40,4)
600 CALL VCHAR(11,20,40,3)
610 CALL VCHAR(11,21,40,3)
620 CALL VCHAR(15,21,40,5)
630 CALL VCHAR(5,15,40,5)
640 CALL HCHAR(24,16,32)
650 CALL HCHAR(3,16,32)
660 CALL HCHAR(12,5,32)
670 CALL HCHAR(12,28,32)
680 CALL HCHAR(7,15,32)
690 CALL HCHAR(17,21,32)
700 CALL HCHAR(12,20,32,2)
```

```
710 CALL HCHAR(18,7,40,5)
720 CALL VCHAR(19,7,40)
730 CALL VCHAR(19,11,40)
740 FOR A=22 TO 25
750 CALL VCHAR(8,A,40,7)
760 NEXT A
770 CALL COLOR(9,2,12)
780 CALL COLOR(11,5,1)
790 CALL COLOR(10,7,1)
800 CALL VCHAR(9,23,120,3)
810 CALL VCHAR(9,24,120,3)
820 REM SET TIME
830 TIME=41
840 TIME=TIME-1
850 IF LEN(STR$(TIME))=1 THE
N 860 ELSE 870
860 CALL HCHAR(19,10,32)
870 FOR I=1 TO LEN(STR$(TIME
))
880 CALL HCHAR(19,I+8,ASC(SE
G$(STR$(TIME),I,1)))
890 NEXT I
900 REM
910 REM MAIN LOOP
920 RANDOMIZE
930 CALL HCHAR(R,C,112)
940 GOSUB 1160
950 GOSUB 1100
960 GOSUB 1160
970 GOSUB 2100
980 CALL GCHAR(12,21,E)
990 IF E=112 THEN 2400
1000 GOSUB 1160
1010 MX=INT(RND*10)+1
1020 IF MX=1 THEN 1130
1030 GOSUB 1160
1040 DC=INT(RND*10)+1
1050 ON DC GOSUB 1740,1810,1
880,1950,2020,1950,2020,1950
,2020,1740
1060 GOSUB 1160
1070 CALL HCHAR(RR,CC,104)
1080 GOTO 930
1090 REM CORE COLOR
1100 COLL=INT(RND*15)+1
1110 CALL COLOR(12,COLL,1)
1120 RETURN
1130 GOSUB 1440
1140 GOTO 930
1150 REM MAN MOVES
1160 CALL HCHAR(R,C,32)
1170 CALL KEY(0,K,S)
1180 IF K=69 THEN 1190 ELSE
1240
1190 R=R-1
1200 CALL GCHAR(R,C,G)
1210 IF G>32 THEN 1220 ELSE
1230
1220 R=R+1
1230 GOTO 1410
1240 IF K=88 THEN 1250 ELSE
1300
1250 R=R+1
1260 CALL GCHAR(R,C,G)
1270 IF G>32 THEN 1280 ELSE
1290
1280 R=R-1
1290 GOTO 1410
1300 IF K=83 THEN 1310 ELSE
1360
1310 C=C-1
1320 CALL GCHAR(R,C,G)
1330 IF G>32 THEN 1400 ELSE
1410
1340 C=C+1
1350 GOTO 1410
1360 IF K=68 THEN 1370 ELSE
1410
1370 C=C+1
1380 CALL GCHAR(R,C,G)
1390 IF G>32 THEN 1400 ELSE
1410
1400 C=C-1
1410 CALL HCHAR(R,C,112)
1420 RETURN
1430 REM ROBOT MOVES
1440 FOR T=4 TO 29
1450 CALL HCHAR(4,T,104)
1460 CALL GCHAR(4,T+1,GC)
1470 IF GC=112 THEN 2320
1480 CALL HCHAR(4,T,32)
1490 NEXT T
1500 GOSUB 2100
1510 FOR VV=5 TO 21
1520 CALL VCHAR(VV,29,104)
1530 CALL GCHAR(VV+1,29,GC)
1540 IF GC=112 THEN 2320
1550 CALL VCHAR(VV,29,32)
1560 NEXT VV
1570 GOSUB 2100
1580 FOR CO=28 TO 4 STEP -1
1590 CALL HCHAR(21,CO,104)
1600 CALL GCHAR(21,CO+1,GC)
1610 IF GC=112 THEN 2320
1620 CALL HCHAR(21,CO,32)
1630 NEXT CO
1640 GOSUB 2100
1650 FOR VC=20 TO 5 STEP -1
1660 CALL VCHAR(VC,4,104)
1670 CALL GCHAR(VC+1,4,GC)
1680 IF GC=112 THEN 2320
1690 CALL VCHAR(VC,4,32)
1700 NEXT VC
1710 GOSUB 2100
1720 RETURN
1730 REM DOOR CONTROL
1740 CALL HCHAR(3,16,97)
1750 CALL HCHAR(12,5,32)
1760 CALL HCHAR(12,28,32)
1770 CALL HCHAR(7,15,32)
1780 CALL HCHAR(17,21,32)
1790 CALL SOUND(100,110,10)
1800 RETURN
1810 CALL HCHAR(12,5,97)
1820 CALL HCHAR(12,28,32)
1830 CALL HCHAR(7,15,32)
1840 CALL HCHAR(17,21,32)
1850 CALL HCHAR(3,16,32)
1860 CALL SOUND(100,110,10)
1870 RETURN
1880 CALL HCHAR(12,28,97)
1890 CALL HCHAR(7,15,32)
1900 CALL HCHAR(17,21,32)
1910 CALL HCHAR(3,16,32)
1920 CALL HCHAR(12,5,32)
1930 CALL SOUND(100,110,10)
1940 RETURN
1950 CALL HCHAR(7,15,97)
1960 CALL HCHAR(17,21,32)
1970 CALL HCHAR(3,16,32)
1980 CALL HCHAR(12,5,32)
1990 CALL HCHAR(12,28,32)
2000 CALL SOUND(100,110,10)
2010 RETURN
2020 CALL HCHAR(17,21,97)
2030 CALL HCHAR(3,16,32)
2040 CALL HCHAR(12,5,32)
2050 CALL HCHAR(12,28,32)
2060 CALL HCHAR(7,15,32)
2070 CALL SOUND(100,110,10)
2080 RETURN
2090 REM TIME COUNT
2100 TIME=TIME-1
2110 IF LEN(STR$(TIME))=1 TH
EN 2120 ELSE 2130
2120 CALL HCHAR(19,10,32)
2130 FOR I=1 TO LEN(STR$(TIM
E))
2140 CALL HCHAR(19,I+8,ASC(S
EG$(STR$(TIME),I,1)))
2150 IF TIME=0 THEN 2190
2160 NEXT I
2170 CALL SOUND(100,1760,0)
2180 RETURN
2190 REM DESTRUCTION
2200 CALL COLOR(2,14,1)
2210 FOR S=30 TO 0 STEP -1
2220 CALL SOUND(50,110,S)
2230 NEXT S
2240 CALL SOUND(300,110,0)
2250 CALL CLEAR
2260 CALL SCREEN(14)
2270 PRINT "WHO BLEW THE REA
CTOR UP THEN": : : : : :
2280 FOR D=1 TO 2000
2290 NEXT D
2300 END
2310 REM CAUGHT BY ROBOT
2320 CALL HCHAR(R,C,105)
2330 CALL SOUND(300,-1,0)
2340 CALL HCHAR(R,C,32)
2350 FOR DE=1 TO 1000
2360 NEXT DE
2370 CALL CLEAR
2380 PRINT "OH DEAR!RUN OVER
 BY A ROBOT!": : : : : : : :
: :
2390 STOP
2400 FOR V=30 TO 0 STEP -1
2410 CALL SOUND(50,392,V)
2420 NEXT V
2430 CALL CLEAR
2440 PRINT "FANTASTIC YOU DI
D IT WITH";TIME:"SECONDS LEF
T"
2450 FOR D=1 TO 3000
2460 NEXT D
2470 END
```

```
100 ! MUSICAL KALEIDOSCOPE
      ADAPTED FROM HCM
110 CALL CHAR(43,RPT$("F",16
)):: CALL SCREEN(2):: FOR X=
1 TO 13 :: CALL SPRITE(#X,43
,X+2,192,24,-128,127,#28-X,4
3,17-X,192,224,-128,-127)::
NEXT X :: A=1
120 T=100*A
130 A$="DABCDCBCDABCDCBCDABC
DCBCDABCDCBCDABCDCBCDABCDCBC
DABCDCBCDABCDCBCACEACEBEABDA
BDCDACFAEADCAAAAAAA"
140 B$="IIIIIIIIHHHHHHHHGGGG
GGGGFFFFFFFFIIIIIIIIHHHHHHHH
GGGGGGGGFFFFFFFFIIKKKKKIIIKK
KKIIIIKKKKKKBBCCCCD"
150 C$="KKKKKKKKKKKKKKKKKKKK
KKKKKKKKKKKKKKKKKKKKKKKKKKKK
KKKKKKKKKKKKKKKKKIIKKKKKIIIKK
KKIIIIKKKKKKDDFFEEI"
160 DIM N(11)
170 READ N(0),N(1),N(2),N(3)
,N(4),N(5),N(6),N(7),N(8),N(
9),N(10),N(11)
180 DATA 110,262,311,349,392
,415,440,466,494,523,466,200
00
190 FOR Z=1 TO LEN(A$)
200 CALL SOUND(T,N(ASC(SEG$(
A$,Z,1))-64),5,N(ASC(SEG$(B$
,Z,1))-64),2,N(ASC(SEG$(C$,Z
,1))-64),0)
210 NEXT Z :: RESTORE 180 ::
 A=-A :: GOTO 120
```

A LOOK AT GPLLNK by R. A. Green.

Reprinted from The Ottawa T.I. 99/4 User's Group NEWSLETTER ...... September, 1985

The Operating System of the TI 99/4A consists of code in ROM and in GROM. The code in ROM is assembler language. The code in GROM is TI's proprietary Graphics Programming Language (GPL).
The ROM code has three main functions: interrupt processing, floating point arithemetic and GPL code interpretation. The GROM code has everything else !

There are, in all this GPL code, several very useful routines that can be used by Assembler language programs. The Editor/Assembler and the Mini Memory modules provide a means, called GPLLNK, to access these routines in GROM. The Extended Basic and TI Writer modules do not provide a link to GPL.

I have developed a GPL link routine that will work for all modules. The Assembler source listing is shown below. The code for this routine is a bit trickey, so a few notes for those who want to understand the code may be in order.

1. The workspace registers are alraedy loaded with some necessary values when RAGLNK is called.
2. The first, and only the first, time RAGLNK is called, it searches all GROMs until the hexadecimal value 0FFF is found.
3. The GPL operation code ›0F is a call to an assembler language routine. The byte following the ›0F, in our case, ›FF gives the table number and entry number in that table. Table number 15 (›0F) begins at ›8300 in the console CPU RAM, and entry 15 in this table is at address ›831E.
4. A GPL CALL stacks the current GROM address then branches to the routine to be called. A GPL RETURN unstacks a GROM address then resumes execution at that address. RAGLNK stacks the GROM address of the ›0FFF instruction, then goes to the GPL interpreter to begin execution of the GROM subroutine. When the GPL subroutine does a RETURN, the ›0FFF instruction is executed, causing GPL to exit to the assembler language routine whose address is at ›831E. This brings 1 back to RAGLNK who returns to his caller.

<u>ASSEMBLER SOURCE LISTING :</u>

```
*TITLE:     GPLLNK Subroutine
*AUTHOR:    R.A.Green
*FUNCTION:  Provides access to the GPL routines,
*           no matter which cartridge you are using.
*LINKAGE:   Same as described for GPLLNK in E/A or
*           MM manuals, except that the GPL STATUS
*           byte need not be reset before calling:
*                      BLWP  @RAGLNK
*                      DATA  GPL-routine-address
*NOTES:     This routine depends upon finding the
*           value ›0FFF somewhere in GROM. This
*           value occurs at least 3 times in the
*           console GROMS in my machine. ›0F is the
*           GPL opcode to call an assembler routine.
*
       DEF   RAGLNK
RAGLNK DATA  WSP,$+2    Linkage/Transfer Vector
       MOV   R0,R0      Do we have an address of ›0FFF ?
       JNE   STACK      Jump yes
*     Find an occurrence of  ›0FFF  somewhere in GROM
       MOVB  R0,*R3     Set the GROM address to zero
       MOVB  R0,*R3
       JMP   $+4
SRCH1  INC   R0         Increment our GROM address
       MOVB  *R4,R1     Get next GROM byte
SRCH2  CI    R1,›0F00   Is it the start of our value ?
       JNE   SCH1       Jump no, keep looking
       MOVB  *R4,R1     Get the byte after ›0F
       CI    R1, ›FF00  Do we have ›0FFF ?
       JEQ   STACK      Jump yes, EUREKA !
       INC   R0         Bump our GROM address past ›0F
       JMP   SRCH2      And keep looking.
*       Notice that the above loop will not end if we do
*   .   not find an occurrence of ›0FFF
* Put our GROM address on the GPL subroutine stack.
STACK  INCT  *R7        Bump GPL stack ptr at ›8373
       MOVB  *R7,@REG2+1  Get stack ptr into ›83xx
       MOV   R0,*R2      Our GROM address to the stack
       MOV   *R6,R9      Save contents of ›831E
       MOV   R5,*R6      Put address of BACK into entry
*                        ›F of table ›F.
* Get GPL routine address from CALLER
       MOVB  R10,@›837C   Reset GPL STATUS byte
       MOV   *R14+,R8     Fetch the GROM address
       LWPI  ›83E0        Switch to the GPL workspace
       MOV   @REG8,R6     R6=next GROM addr to interpret
       B     @›0060       Go to GPL interpreter
* Hopefully GPL will come back here
BACK   LWPI  WSP          Switch back to our workspace
       MOV   R9,*R6       Restore value in ›831E
       RTWP               Return to calling program
* Our workspace registers loaded with interestinmg stuff.
WSP    DATA  0        R0=our special GROM address
       DATA  0        R1 LSB is zero
REG2   DATA  ›8300    R2=GPL subroutine stack address
       DATA  ›9C02    R3=GROM write address address
       DATA  ›9800    R4=GROM raed data address
       DATA  BACK     R5=address for GPL to come back to
       DATA  ›831E    R6=address of entry ›F of table ›F
       DATA  ›8373    R7=pointer to GPL subroutine stack
REG8   BSS   2        R8=GROM addr of GPL routine
       BSS   2        R9=saved contents of ›831E
       DATA  0        R10=ZERO
       BSS   10       R11-R15
       END
```

○

ON ERROR from SUB ROUTINE or SUB PROGRAM
by Ross Mudie

Derived from a member's question.

How do you make ON ERROR work from a SUB-ROUTINE when an error is detected?

Answer...

You should set up error handling within the subroutine or subprogram so that your program always returns via the RETURN or SUBEND statement.

If you exit from SUBROUTINES without using the RETURN you build up return addresses on the return stack until the VDP RAM runs out of free space & crash for that reason.

With SUB PROGRAMS, the next time you call the subprogram from which you exited with ON ERROR the program will terminate in a RECURSIVE SUBPROGRAM CALL error.

Basically you need to handle errors fully in the subroutine or subprogram by setting the appropiate ON ERROR as entry is made to the routine and reset the error handling as you leave the routine with ON ERROR STOP.

✸ ✸ ✸

From P2

There are other committee members but quite frankly Cyril Bohlsen John Paine and Co. are "flat-chat" looking after the shop and other activities for the benefit of ALL members. Don't forget to check the club library which Russell brings to each meeting for magazines, local and overseas, and books. It is quite extensive and caters to virtually every need.

From: Wade Bowmer
45 Yanderra Ave
Bangor NSW 2234

To: All TIsHUG members who are interested in "illegal" things you can get away with in BASIC and other tidbits of information that will some day be useful, I present you with:

TIDBITS FIVE
TIDBITS FIVE

What have I got for you this month?

I will dispense with the greeting. A few things dealt with in this article include helpful (it may also be biased) information about, wait for it, BASIC's "dreaded" **GOTO** statement. The very fact that it could be biased should be an incentive to read it, because it would express someone else's view about the highly shunned "GOTO debate". Also, some interesting places where you can leave out spaces. (But BASIC will only put most of them back again, so it really only saves typing.)

And in my rush to get Tidbits IV printed in time, I didn't do much beyond providing a list of tokens. So rather than go on about it, I'm going to provide the rest of that section.

Beginners Tips.

I suppose really that I don't have to keep to my "framework" for my articles that I set down. So, since I'm combining the two Beginner's Tips together, and besides, they can be of use to more advanced programmers, a better subtitle would be:

Spaces: Leave them out!

Well, not all of them. But there are a lot you can leave out when typing in a program.

The one that comes most immediately to mind is after line numbers. Since every single statement (even assigning variables) starts with anything but a numeral, this is very feasible. (It makes you want to say "Why didn't I think of that?", doesn't it? Well, perhaps you were'nt looking for it, but have a look at a Commodore 64 BASIC listing, or VIC= 20, or MSX, or ...) I must say that I didn't discover this one, but came across it in some "Tigercub Tips".

Before I go any further, I must point out that most of the spaces I mention of leaving out refer to those found in any standard TI BASIC or Extended BASIC listing, output by BASIC itself. If you happen to be in the habit of placing a space before and after every number you type, well! that is certainly one good place to start leaving out spaces!

Another place which comes to mind is around the double colon (::) in Extended BASIC. The reason you can leave out the space after the it is the same reason why you can leave out the space after a line number. The reason you can leave out the space before it is because Extended BASIC will recognize a :: anywhere except in a string constant (a quoted string), or after !, **REM** or **DATA**.

Still on colons, sometimes people advise to use **PRINT** :;:;:;: as an alternative to **PRINT** : : : : in Extended BASIC (or even **PRINT** :::: in **TI-BASIC**!). **There really isn't** anything wrong with it- it does, of course, work. However, it does have these four faults: It is somewhat harder to type, since it is easier to hold the SHIFT key down and alternate your keypresses between the : key and the spacebar. (The spacebar always returns a space no matter whether SHIFT, FCTN, CTRL or nothing is pressed with it.) It is slower to RUN; however, this is rarely noticeable. And finally, it wastes memory. The ;'s actually take up room in memory between the :'s, wheras the spaces don't. So my advice is to use a space there.

While we're in **PRINT**, if the first item in the printlist is either a colon (:), a comma (,) or a string constant (quoted string, starts with a "), then you don't have to type a space after **PRINT**.

This trick works before a string constant in the following staements: **DATA**, **DELETE**, **DISPLAY** (without any options!), **IMAGE** (enclose the image in quotes), **INPUT** and **LINPUT** (with a string-prompt), **LIST** (with a device name, or with only an end line-number, eg. **LIST-999**), **MERGE** and **OLD** (enclose the device.filename in quotes), **PRINT** (already mentioned- see previous paragraph), **RUN** (when chaining to another program) and **USING**.

A similar situation exists with the **#** in file-related statements. Hence, you can leave out the space before the **#** when typing the following statements: **CLOSE**, **INPUT**, **LINPUT**, **OPEN**, **PRINT** and **RESTORE**. (With **INPUT**, **LINPUT**, **PRINT** and **RESTORE**, I mean **INPUT#**, **LINPUT#**, **PRINT#** and **RESTORE#**.)

A special situation for spaces occurs with the close paranthesis,). BASIC will not put a space after it. So you only waste a keystroke by putting a space there.

Which brings me to my favourite "space"-saver: **FOR**. The standard format used is **FOR** variable= number **TO** number. **Straight** away, I can see a place that doesn't need a space when typing in a program, before **TO**. But you can only omit (that's a fancy word! It means "leave out") that space if the starting value (which comes before **TO**) is either a number or a numeric-expression that ends with a number. An array element (such as B(5)) is an exeption to the rule, simply because of the). Another place which is in exactly the same situation is the space before **STEP**.

I hear you cry; "But there's no **STEP** in your example!" All right, I can soon fix that: **FOR** variable=start-value **TO** final-value **STEP** stepping- value. The situation over the space between **STEP** and the end-value is exactly the same as the space between TO and the start-value.

Incidently, there's another double situation like that! If the stepping value (also known as the increment) is negative, then a space between **STEP** and it is unnecessary! Likewise, for the end-value and **TO**.

Now, high time for some examples!

```
  Will Work              Won't Work
FOR A=1 TO 5            FOR B=2 TO3
                                  ^
FOR Z=2TO 10STEP 2      FOR Y=8 TO20 STEP5
                                  ^     ^
FOR P=H*4TO Q STEP-1    FORI=4*HTO JSTEP1
                           ^    ^  ^    ^
FOR I=A TO B+3STEP 2    FOR J=ATOB+3 STEP 2
                               ^ ^
FOR O=-1TO-8STEP-1      FORO=O-ZTO-JSTEP-1
                           ^    ^   ^
```

(The arrows indicate the mistakes.)

Going past the line length limit.

There's another purpose in discussing all the places you can leave out spaces. Because it's possible to type a line longer than the maximum size. How? Well, the TI only limits the length of the line when it is detokenized text. So if you type a line, leaving out all the "unnecessary" spaces, that touches the 5 line limit (Oh, I forgot, I'm talking about Extended BASIC, although the same technique can be used with very long expressions in TI-BASIC), then you'll find that it is longer than 5 lines when it is LISTed! And when you edit it, you'll find that you can make it up to 7 lines long!

You can make it longer, by using the same "space" saving method, but then you start getting *** LINE TOO LONG** messages.

A word of warning: if, when you delete part of the line, its length diminishes to below the 5-line limit, it's probably that it is suddenly back in force again! (Sometimes, yea, sometimes, nay...) When the TI lists a line for editing, it sets a limit and checks the line length. If the line length is greater, it increases the limit by two screen lines and tests it again, until the line's length is below the limit. The first limit is five (as you know), and when you delete part of the line, it checks to see whether it can reduce the line limit or not, and if so, does.

Oh yes, when you wish to change a line number of a line (effectively copying it), the usual method is with **REDO**, after pressing **ENTER**, after being in edit mode at that line. But if that line is longer than 5 lines, **REDO** only fetches only the first five lines! Fortunately, you can extend a line up to 10 screen lines in **REDO** mode, so it only means having to type the extra "illegal" portion of the line.

Playing with the Video Chip... From BASIC.

One way is with the PEEKV and POKEV subprograms found in the Editor/Assembler and Minimem modules or VPEEK, VPOKE and WRTRG subprograms found with the Corcomp Disk Manager. Another, more interesting way, is using !.

In my first **Tidbits** article, I mentioned that the **LIST** command detokenizes codes in ! and **REM** statements. Well, if you fill up an entire line with a high character code... Perhaps an example would be in order.

In either BASIC, enter **NEW**. Now type **1REM** and press and hold **CTRL-U** until the computer beeps at you. Press **ENTER**. Now type 1 and either **FCTN-E** or **FCTN-X**. Now, step through various video patterns with **FCTN-D**! When you've had enough, just press **ENTER**, or, if that doesn't work, press **FCTN-=**, which is **QUIT**.

The Great GOTO Debate.

I have a book I bought a few years ago called "BASIC Fun". It proports to teach BASIC, but it gives so many misideas about several computers (including the TI) that it's not at all worth buying.

(The authors think that most home computer versions of BASIC are still very much like the original Altair BASIC, which did not have string arrays!)

But that's beside the point. In it, there is a program that shows the "unlimited power" of **GOTO**.

I will include it only to show it to you.

```
100 PRINT "YOU ARE ON AN EXPEDITION":"
   SEARCHING FOR THE LOST ARK.":
110 GOTO 260
120 PRINT "YOUR CREW STARTS DIGGING.":
130 GOTO 220
140 PRINT "AND SECURE THE ROPES TO THE":"
   ARK. A HELICOPTER SAFELY.":
150 GOTO 280
160 PRINT "WHERE IT WAS BURIED.":
170 GOTO 240
180 PRINT "TOUGH LUCK!!":" THINK OF YOUR
   OWN SOLUTION.":
190 GOTO 340
200 PRINT "IS THERE!":
210 GOTO 300
220 PRINT "THEY UNCOVER A PIT FULL OF":"
   SNAKES, BUT THE ARK":
230 GOTO 200
240 PRINT "YOU SEE THE ""X"" ON THE
   MAP.":" IT IS BY THE TEMPLE OF THE":
250 GOTO 320
260 PRINT "A FRIEND SHOWS YOU A LONG":"
   LOST MAP. ""X"" MARKS THE":" SPOT.":
270 GOTO 160
280 PRINT "HOISTS IT OUT, BUT THERE ARE NO
   ROPES LEFT FOR YOU.":
290 GOTO 180
300 PRINT "YOU ENTER THE PIT WITH":"
   ROPES, AVOIDING THE SNAKES,":
310 GOTO 140
320 PRINT "ANCIENT SUN GOD.":
330 GOTO 120
340 END
```

(BASIC Fun, pg 40-41)

Can you see the story? Yes, you may type it in, but it is more educational to follow it through by hand, since it shows a very good example of what is possible because of **GOTO**.

In December '84 (that's a long way back!), Tony McGovern in his Extended Tutorial, presented some of his views of **GOTO** in conjunction with subprograms. (For those of you who haven't got Extended BASIC, I mean subprograms you define yourself, and yes I will try to bear TI-BASIC users in mind.)

My usage of subprograms has climaxed and is now at a level where I use my own subprograms as user-defined statements, or customised statements.

So that means that my use of **GOTO** has become important. As it is, I don't use **GOTO** to simply detour to a less crowded area of line numbers. (Well, I rarely do...) Why? Is it because I'm a good programmer? Well, yes, well, no. If you haven't got Extended BASIC then go out and buy it, now! Because its ability to place several statements on a single line is invaluable in avoiding the same **GOTO** bottlenecks that exist and are often difficult to manage in TI-BASIC.

Tony McGovern stated that "Backward **GOTO**'s over more than one or two lines of code, or any forward **GOTO**'s at all, should occur under the most regular of logical layouts..." While that doesn't conflict with my normal use of **GOTO**, Tony McGovern might find the number of **GOTO**'s questionable in bits of code that involve the user entering a list of values with the ability to "backtrack" to alter previos entries (similar to entering colours in Multiplan's™ Window Paint command).

So what have I got to say about **GOTO**? In BASIC, I find it perfectly indispensable, unless such statements as **REPEAT..UNTIL** or **DO..WHILE** loops are available. In Assembly (and I'm talking about **JMP**, here), I use it only as a secondary instruction, straight line coding is often nice- especially in Machine Language. In Logo, I don't even use it. In fact, I don't think Logo even needs such a statement.

As for using **GOTO**, well, the program I showed you earlier is an excellent example of how not to use **GOTO**!

Computer Codes, Second Attempt.

I'll go through the tokens, one by one. ≡ means a space.

128 is not used.	158 is **OPTION≡**
129 is **≡ELSE≡**	159 is **OPEN≡**
130 is **≡::≡**	160 is **SUB≡**
131 is **!**	161 is **CLOSE≡**
132 is **IF≡**	162 is **DISPLAY≡**
133 is **GO≡**	163 is **IMAGE≡**
134 is **GOTO≡**	164 is **ACCEPT≡**
135 is **GOSUB≡**	165 is **ERROR≡**
136 is **RETURN≡**	166 is **WARNING≡**
137 is **DEF≡**	167 is **SUBEXIT≡**
138 is **DIM≡**	168 is **SUBEND≡**
139 is **END≡**	169 is **RUN≡**
140 is **FOR≡**	170 is **LINPUT≡**
141 is **LET≡**	171
142 is **BREAK≡**	172
143 is **UNBREAK≡**	173 not used.
144 is **TRACE≡**	174
145 is **UNTRACE≡**	175
146 is **INPUT≡**	176 is **THEN≡**
147 is **DATA≡**	177 is **TO≡**
148 is **RESTORE**	178 is **STEP≡**
149 is **RANDOMIZE≡**	
150 is **NEXT≡**	
151 is **READ≡**	
152 is **STOP≡**	
153 is **DELETE≡**	
154 is **REM≡**	
155 is **ON≡**	
156 is **PRINT≡**	
157 is **CALL≡**	

179 is parameter separator (comma ,).
180 is PRINT-list separator (semi-colon ;).
181 is a major separator (colon :).
182 is the end of a parameter list (close parenthesis)).
183 is the start of a parameter list (open parenthesis ().
184 is the string concatenator (ampersand &).
185 is not used.
186 is **OR**
187 is **AND**
188 is **XOR**
189 is **NOT**
190 is the equals sign, =. Used in **LET**, also.
191 is the logical less-than sign, <.
192 is the logical greater-than sign, >.
193 is the plus sign, +.
194 is the minus sign, -.
195 is the multiply sign, *.
196 is the division sign, /.
197 is the caret, or circumflex, ^.
198 is not used.
199 means that a quoted string (a string constant) follows.
200 means that an unquoted string (a literal constant, eg. a subprogram name) follows.

For tokens 199 and 200, the next byte signifies the length of the constant (in bytes) and that many bytes follow, making up the constant. For example, 199,3,77,69,33 would LIST under BASIC as "ME!", and 157,200,5,67,76,69,65,82 means **CALL CLEAR**.

201 indicates that a line number follows, in high byte, low byte format. For example, 134,201,0,100 is **GOTO 100**, tokenised.

202 is **EOF** (End of File).
203 is **ABS** (Absolute).
204 is **ATN** (Arctangent).
205 is **COS** (Cosine).
206 is **EXP** (Exponential).
207 is **INT** (Greatest Integer).
208 is **LOG** (Logarithm).
209 is **SGN** (Signum).
210 is **SIN** (Sine).
211 is **SQR** (Square Root).
212 is **TAN** (Tangent).
213 is **LEN** (Length).
214 is **CHR$** (Character String).
215 is **RND** (Random).
216 is **SEG$** (Segment String).
217 is **POS** (Position).
218 is **VAL** (Value).
219 is **STR$** (String).
220 is **ASC** (ASCII).
221 is **PI** (π).
222 is **REC** (Record).
223 is **MAX** (Maximum).
224 is **MIN** (Minimum).
225 is **RPT$** (Repeat String).
226
227
228 } not used.
229
230
231

232 is NUMERIC	243 is VARIABLE
233 is DIGIT	244 is RELATIVE
234 is UALPHA	245 is INTERNAL
235 is SIZE	246 is SEQUENTIAL
236 is ALL	247 is OUTPUT
237 is USING	248 is UPDATE
238 is BEEP	249 is APPEND
239 is ERASE	250 is FIXED
240 is AT	251 is PERMANENT
241 is BASE	252 is TAB
242 is not used.	253 is #
	254 is VALIDATE

Well, that about wraps it up for TIDBITS FIVE! Next month, I'll have something to say about the prescan commands. Enjoy............!

≈ ≈

TIsHUG SOFTWARE COMPETITION

How would you like to win a great prize merely by putting your programming skills to good use? You would? Then read on because a great software competition has been organised and there are some really good prizes to be won.

First, here are the rules:

1. The submitted program can be in any language capable of running on the TI99/4A computer using common peripherals.

2. The completed program must be submitted on cassette or disk and must contain operating instructions either within the main program or as a separate program.

3. TIsHUG committee members, and their families, are ineligible to enter.

4. All programs submitted become the property of TIsHUG.

5. Entries may be submitted either by post to the Secretary or handed in at monthly meetings. Closing date for entries will be the 6th February, 1988.

6. Winner(s) will be announced at the March 1988 meeting.

7. The judges decision will be final and no correspondence will be entered into.

8. Entries will be accepted from non TIsHUG members, however such entries will not be eligible for any prizes.

9. The submitted program must be the original work of the author or joint authors as family participation in the competition is encouraged.

10. The TIsHUG committee, reserve the right not to present the major prize if entries submitted fail to meet an adjudged adequate standard.

THE PRIZES

The overall winning author will be awarded a $200 voucher which can be used to purchase any goods then available from the TIsHUG shop. Other minor awards, of $20 TIsHUG Shop vouchers, will be presented as encouragement to other authors whose programs are adjudged to have sufficient merit.

WHY RUN A SOFTWARE COMPETITION?

Your Committee feels that TIsHUG must stimulate interest in programming. For years we have ran software competitions and some excellent material has been submitted and distributed to the membership. But the point now is that, apart from a handful of active authors, the local supply is drying up, and of course, you do not need me to tell you, that without software, there is zero you can do with your TI.

What this boils down to is that we are relying, in the main, on the generosity of overseas groups to supply us with new software, with precious little to give them in return. Sooner or later they will say, enough is enough, as no doubt you and I would, if it was all one way traffic.

Hence this competition. Hopefully a new base of software will be established which we will be proud to exchange with our friends around this country and overseas.

So go to it. You have several months to complete your program, and who knows, maybe win the big prize.

Text Manipulation

by D.N.Harris

```
10 REM ************************
20 REM **CENTRING by D.N.Harris
30 REM **SENT ON 29/8/1987*****
40 REM **XBAS VERSION**********
50 REM ************************
100 PRINT TAB(28);
110 LINPUT A$
120 CALL CENTRE(A$)
130 GOTO 100
140 SUB CENTRE(A$)
150 A=LEN(A$)
160 B=28-A
170 C=B/2
180 PRINT TAB(C);A$
190 SUBEND
```

```
100 REM *************************
110 REM D.N.HARRIS 29/8/1987*****
130 REM CENTRING*****************
140 REM TI BAS VERSION***********
150 DEF FNCENTRE=.5*(28-LEN(A$))
160 PRINT TAB(28);
170 USE CALL KEY ROUTINE INSTEAD
OF INPUT
180 FOR EXAMPLE IN TI BAS THE INPUT
CAN BE USED BUT THE WHOLE STRING MUST
BE IN QUOTES IF COMMAS OCCUR.
190 INPUT A$
200 PRINT TAB(FNCENTRE);A$
```

```
90 REM OTHER TEXT ROUTINES
100 RANDOMIZE
110 DEF FNRIGHT=28-LEN(A$)
120 DEF FNCENTRE=.5*FNRIGHT
130 DEF FNWEIRD=FNRIGHT*RND*29/16+1
140 PRINT TAB(28);
150 LINPUT A$
160 PRINT TAB(FNRIGHT);A$
170 PRINT TAB(FNCENTRE);A$
180 PRINT TAB(FNWEIRD);A$
190 RANDOMIZE
200 CALL SCREEN(FNWEIRD)
210 PRINT FNWEIRD
220 GOTO 150
```

This last version is a hybrid, since the XBAS command LINPUT is involved. It took a lot of trial and error to find a way to define CENTRE as it crosses strings and numerics in a function, and it is not illustrated in the manuals on programming. I feel it will simplify a lot of programme documentation to be able to write a simple command.

The rule seems to be that TI BASIC requires that the function defined on the left hand side must be a numeric variable if the function on the right hand side generates a numeric constant, as does LEN. It follows that any string expression such as ASC or VAL that generates a number, and this would mean GCHAR and POS as well, must be a numeric function, whereas if the function on the right will have a string OUTPUT, such as CHR$(), then the function defined on the left must be a string function. It took a lot of pecking at the keyboard to find that out, so what I am passing on to you is the fruit of a bit of Research Programming.

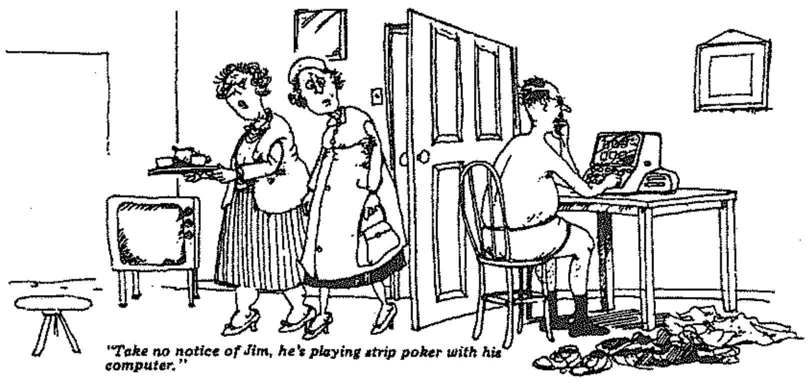
I should put in a plug for myself at this point, by mentioning that I am on the job market, and have training in Chemistry and Computing, and am a non-driver, and have good typing skills, experience in editing, typesetting, data entry, chemical analysis, flowcharting, spellchecking, interviewing, teaching, styrenecutting,spotwelding,some electrical work including assembly of electric motors, and have a kind of above average if not perfect attendance and punctuality record, but am not the right man for Augean stable cleaning. I do, however, lend a psychological boost as well as a helping hand to my co-workers anywhere, and have some impact on morale as well as efficiency, and have added valuable ideas to virtually everything I have touched.

Contact ADVANCED via TEXPAC or ring 502-2267 if interested.

I NOTICED THE ARTICLE BY R.N.HARRIS, LOOKING CLOSER IT SEEMS TO BE AN ARTICLE I WROTE, BUT MY INITIAL LETTER IS NOT R AND I WONDER AT WHICH END THE ERROR OCCURRED. IT WOULD BE IMPROBABLE THAT I WOUL WOULD MISPELL MY OWN NAME. THE NEXT THING POSSIBLE IS THAT THE NAME WAS GARBLED IN TRANSMISSION, AS DOES SOMETIMES HAPPEN WITH TELECOMMUNICATION.

IT IS SAD TO TAKE SOME INTEREST AND EFFORT ONLY TO PASS THE CREDIT TO A MYTHICAL PLAGIARIST WITH A SIMILAR NAME! PERCHANCE I SHOULD TRY TO GET AN EX GRATIA COURT AWARD FROM R.N.HARRIS OVER COPYRIGHT, FOR ABSURD EXAMPLE.

Respectfully,
D.N.Harris


"Take no notice of Jim, he's playing strip poker with his computer."

Tigercub Tips

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

Over 130 original programs in Basic and Extended Basic, available on cassette or disk, now reduced to just $2.00 each, plus $1.50 per order for cassette or disk and PP&M. Cassette programs will not be available after my present stock of blanks is exhausted.
Descriptive catalogs, while they last, $1.00 which is deductable from your first order.

Tigercub Full Disk Collections, reduced to $10 postpaid. Each of these contains either 5 or 6 of my regular $2 catalog programs, and the remaining disk space has been filled with some of the best public domain programs of the same category. I am NOT selling public domain programs - they are a free bonus!
TIGERCUB'S BEST, PROGRAMMING TUTOR, PROGRAMMER'S UTILITIES, BRAIN GAMES, BRAIN TEASERS, BRAIN BUSTERS!, MANEUVERING GAMES, ACTION REFLEX AND CONCENTRATION, TWO-PLAYER GAMES, KID'S GAMES, MORE GAMES, WORD GAMES, ELEMENTARY MATH, MIDDLE/HIGH SCHOOL MATH, VOCABULARY AND READING, MUSICAL EDUCATION, KALEIDOSCOPES AND DISPLAYS

NUTS & BOLTS (No. 1), a full disk of 100 Extended Basic utility subprograms in merge format, ready to merge into your own programs. Plus the Tigercub Menuloader, a tutorial on using subprograms, and 5 pages of documentation with an example of the use of each subprogram. Reduced to $15.00 postpaid.
NUTS & BOLTS NO. 2, another full disk of 108 utility subprograms in merge format, all new and fully compatible with the last, and with 10 pages of documentation and examples. Also $15 postpaid.

```
****************************
* NUTS & BOLTS #3 is now   *
* ready, another full disk *
* of 140 new merge-format  *
* utility subprograms, all *
* compatible with the pre- *
* vious. With 11 pages of  *
* documentation, $15 ppd.  *
****************************
```

TIPS FROM THE TIGERCUB, a full disk containing the complete contents of this newsletter Nos. 1 through 14, 50 original programs and files, reduced to $10 ppd.
TIPS FROM THE TIGERCUB VOL. 2, another diskfull, complete contents of Nos. 15 through 24, over 60 files and programs, also just $10
TIPS FROM THE TIGERCUB VOL. 3, another 62 programs, tips and routines from Nos. 25 through 32, $10 postpaid.
TIPS FROM THE TIGERCUB VOL. 4, another 48 programs and files from issues 33 through 41, also $10 postpaid.

If you have as much trouble as I do, trying to get the strip labels lined up in the printer, you'll like this one -

```
100 DISPLAY AT(4,7)ERASE ALL
:"TIGERCUB LABELER": : : :"
This label maker will allow"
:"you to specify different":
"printer codes for each line
"
110 DISPLAY AT(11,1):"of a 5
-line label.": : :" You may
stop the program":"while lab
els are printing":"by pressi
ng any key, turn"
120 DISPLAY AT(17,1):"off th
e printer to adjust":"the la
bels, turn it back on,":"and
 press any key to con-":"tin
ue printing."
130 DISPLAY AT(23,1):"Printe
r designation?":"PIO" :: ACC
EPT AT(24,1)SIZE(-28)BEEP:PR
$ :: OPEN #1:PR$ :: P$,E$,DS
$,CEN$="Y" :: DW$,I$,SS$,U$=
"N" :: P=1
140 CALL CHAR(95,"FF")
150 FOR J=1 TO 5 :: CALL KEY
(3,K,S)
160 DISPLAY AT(2,1)ERASE ALL
:"Line #";J;" - PRINT? "&P$
:: CALL QUERY(2,20,P$):: IF
P$="N" THEN L$(J)="" :: GOTO
 360
170 IF J>1 THEN DISPLAY AT(4
,1):"Change codes? N" :: CAL
L QUERY(4,15,Q$):: IF Q$="N"
 THEN 300
180 DISPLAY AT(4,1):"Print p
itch? ";P:" (1)pica":" (2)el
ite":" (3)condensed" :: ACCE
PT AT(4,15)SIZE(-1)VALIDATE(
"123"):P
190 CI=(P=1)*-10+(P=2)*-12+(
P=3)*-17 :: L$(J)=CHR$(27)&"
B"&CHR$(P):: DISPLAY AT(5,1)
:"":"":""
200 DISPLAY AT(6,1):"Double
width? "&DW$ :: CALL QUERY(6
,15,DW$):: IF DW$="Y" THEN C
I=CI/2 :: L$(J)=L$(J)&CHR$(1
4)ELSE L$(J)=L$(J)&CHR$(20)
210 DISPLAY AT(8,1):"Italics
? "&I$ :: CALL QUERY(8,10,I$
):: IF I$="Y" THEN L$(J)=L$(
J)&CHR$(27)&"4" ELSE L$(J)=L
$(J)&CHR$(27)&"5"
220 DISPLAY AT(10,1):"Supers
cript? "&SS$ :: CALL QUERY(1
0,14,SS$):: IF SS$="Y" THEN
L$(J)=L$(J)&CHR$(27)&CHR$(83
)&CHR$(0)ELSE L$(J)=L$(J)&CH
R$(27)&CHR$(84)
230 IF SS$="Y" THEN 250
240 DISPLAY AT(12,1):"Double
-strike? "&DS$ :: CALL QUERY
(12,16,DS$):: IF DS$="Y" THE
N L$(J)=L$(J)&CHR$(27)&"G" E
LSE L$(J)=L$(J)&CHR$(27)&"H"
250 IF P<>1 OR SS$="Y" THEN
270 :: DISPLAY AT(14,1):"Emp
hasized? "&E$ :: CALL QUERY(
14,13,E$)
260 IF E$="Y" THEN L$(J)=L$(
J)&CHR$(27)&"E" ELSE L$(J)=L
$(J)&CHR$(27)&"F"
270 DISPLAY AT(16,1):"Underl
ine? "&U$ :: CALL QUERY(16,1
2,U$)
280 IF U$="N" THEN L$(J)=L$(
J)&CHR$(27)&CHR$(45)&CHR$(0)
290 DISPLAY AT(18,1):"Center
 text? Y" :: CALL QUERY(18,1
4,CEN$)
300 DISPLAY AT(18,1):"Type l
ine";J;". Enter each":"scree
n line, enter again":"when d
one." :: DISPLAY AT(22,1):RP
T$("_",INT(CI*3.5)):: R=21 :
: CALL KEY(5,K,S)
310 ACCEPT AT(R,1):M$ :: IF
M$="" THEN 320 :: A$=A$&M$ :
: R=R+1 :: GOTO 310
320 IF LEN(A$)>INT(CI*3.5)TH
EN DISPLAY AT(16,1):"LINE TO
O LONG!" :: CALL SOUND(300,1
10,0,-4,0):: A$="" :: R=21 :
: GOTO 310
330 L=LEN(A$):: IF U$="Y" TH
EN A$=CHR$(27)&CHR$(45)&CHR$
(1)&A$&CHR$(27)&CHR$(45)&CHR
$(0)
340 IF CEN$="Y" THEN A$=RPT$
(" ",(INT(CI*3.5)-L)/2)&A$
350 L$(J)=L$(J)&A$ :: A$=""
360 NEXT J
370 DISPLAY AT(12,1)ERASE AL
L:"Print how many?" :: ACCEP
T AT(12,17):N
380 FOR J=1 TO N :: FOR K=1
TO 6 :: PRINT #1:L$(K):: NEX
T K
390 CALL KEY(0,K,S):: IF S=0
 THEN 410 ELSE CLOSE #1
400 CALL KEY(0,K1,S1):: IF S
1<1 THEN 400 ELSE OPEN #1:PR
$
410 NEXT J
420 DISPLAY AT(12,8)ERASE AL
L:"Another?" :: CALL QUERY(1
2,17,Q$):: IF Q$="N" THEN ST
OP ELSE 150
430 SUB QUERY(R,C,Q$):: ACCE
PT AT(R,C)SIZE(-1)VALIDATE("
YN")BEEP:Q$ :: SUBEND
```

More peculiarities of the TI computer -

```
90 CALL CLEAR :: PRINT TAB(7
);"SPRITE PUZZLE #1":"
 from Tigercub"
100 PRINT "A non-existent sp
rite can be":"created by CAL
L MOTION.": :"It apparently
```

```
starts in"
110 PRINT "dot-row 1, dot-co
lumn 1, and":"has color 1, b
ut its pattern":"is not that
 of any ASCII!"
120 !by Jim Peterson
130 FOR CH=0 TO 255 :: PRINT
 CHR$(CH);:: NEXT CH
135 PRINT "CALL MOTION(#1,5,
5):: CALL COLOR(#1,16):: CAL
L MAGNIFY(4)"
140 CALL MOTION(#1,5,5):: CA
LL COLOR(#1,16):: CALL MAGNI
FY(4)
150 GOTO 150
```

And another -

```
100 DISPLAY AT(3,5)ERASE ALL
:"SPRITE PUZZLE #2": :"
 from Tigercub"
110 DISPLAY AT(7,1):"Non-exi
stent sprites can be":"creat
ed by CALL COLOR.": :"Their
existence can be con-"
120 DISPLAY AT(11,1):"firmed
 by CALL COINC, but":"CALL P
OSITION reports that":"they
have no position!"
130 CALL COLOR(#1,16):: CALL
 COLOR(#2,16)
140 CALL COINC(#1,#2,1,X)::
DISPLAY AT(15,1):"COINC #1,#
2=";X :: CALL POSITION(#1,X,
Y)
150 CALL POSITION(#1,X,Y)::
DISPLAY AT(17,1):"POSITION #
1=";X;Y
160 CALL POSITION(#2,X,Y)::
DISPLAY AT(19,1):"POSITION #
2=";X;Y
170 IF FLAG=1 THEN 140 :: FL
AG=1
180 DISPLAY AT(21,1):"PRESS
ANY KEY"
190 CALL KEY(0,K,S):: IF S=0
 THEN DISPLAY AT(21,1):"pres
s any key" :: GOTO 180
200 DISPLAY AT(21,1):"Until
they're set in motion!"
210 CALL MOTION(#1,5,5):: CA
LL MOTION(#2,-5,-5):: GOTO 1
50
```

If you have the Terminal Emulator II, Speech Synthesizer, and a pre-schooler in the house, this will help him to grasp the idea of spelling as well as letter recognition and keyboard familiarization-

```
100 REM  PRE-SPELLER BY JIM
PETERSON
110 REM  TI BASIC WITH TERMI
NAL EMULATOR II AND SPEECH S
YNTHESIZER
120 CALL CLEAR
130 DIM M$(100),S$(100)
140 OPEN #1:"SPEECH",OUTPUT
150 PRINT "         PRE-SPELL
ER"::::::
160 PRINT "TYPE WORDS TO PRA
CTICE"::"TYPE 'END' WHEN FIN
ISHED"
170 X=X+1
180 INPUT M$(X)
190 IF M$(X)="END" THEN 380
200 PRINT #1:M$(X)
210 PRINT "PRONUNCIATION OK?
 (Y/N)"
220 CALL KEY(3,K,S)
230 IF S<1 THEN 220
240 IF K=78 THEN 280
250 IF K<>89 THEN 220
260 S$(X)=M$(X)
270 GOTO 170
280 PRINT "TRY SPELLING PHON
ETICALLY"
290 INPUT S$(X)
300 PRINT #1:S$(X)
310 PRINT "PRONUNCIATION OK?
 (Y/N)"
320 CALL KEY(3,K,S)
330 IF S<1 THEN 320
340 IF K=89 THEN 170
350 IF K<>78 THEN 320
360 PRINT "TRY AGAIN"
370 GOTO 290
380 CALL CLEAR
390 FOR J=1 TO X-1
400 PRINT #1:"CAN YOU SPELL
THIS?"
410 FOR A=1 TO LEN(M$(J))
420 CALL HCHAR(12,8+A,ASC(SE
G$(M$(J),A,1)))
430 NEXT A
440 FOR B=1 TO LEN(M$(J))
450 CALL KEY(3,K,S)
460 IF (S<1)+(K=32)THEN 450
470 IF K=ASC(SEG$(M$(J),B,1)
)THEN 500
480 GOSUB 640
490 GOTO 450
500 C$=C$&CHR$(K)
510 CALL HCHAR(14,8+B,K)
520 NEXT B
530 IF C$<>M$(J)THEN 640
540 PRINT #1:S$(J)
550 FOR D=1 TO 500
560 NEXT D
570 PRINT #1:"VEREE GOOD"
580 FOR D=1 TO 500
590 NEXT D
600 C$=""
610 CALL HCHAR(12,1,32,100)
620 NEXT J
630 GOTO 390
640 PRINT #1:"NO THAT IS NOT
 RIGHT"
650 PRINT #1:"TRY AGAIN"
660 RETURN
```

And, a simple little game that is a bit different than any I've seen -

```
100 !FORMATION by Jim Peters
on - use the S and D keys
110 CALL CLEAR :: CALL CHAR(
100,"381010FEFE383810103838F
EFE10103838"):: CALL SCREEN(
5):: CALL MAGNIFY(2):: RANDO
MIZE
120 V,W,P=0 :: FOR J=1 TO 7
:: CALL SPRITE(#J,100,7,1,25
0*RND+1,10,4):: FOR D=1 TO 1
00 :: NEXT D :: NEXT J :: CA
LL SPRITE(#11,101,16,160,128
)
130 CALL KEY(3,K,S):: W=W+1
:: IF W=150 THEN 170 ELSE IF
 W=300 THEN 180 ELSE IF K=68
 THEN V=V+2+(V>125)*2 ELSE I
F K=83 THEN V=V-2-(V<-125)*2
140 IF P=0 THEN CALL MOTION(
#11,0,V)ELSE IF P=1 THEN CAL
L MOTION(#11,0,V,#12,0,V)ELS
E CALL MOTION(#11,0,V,#12,0,
V,#13,0,V)
150 CALL COINC(ALL,A):: IF A
=0 THEN 130
160 CALL SOUND(1000,-4,0)::
H=MAX(H,W):: DISPLAY AT(23,1
):"SCORE";W:"HIGH SCORE";H :
: CALL DELSPRITE(ALL):: GOTO
 120
170 P=1 :: CALL POSITION(#11
,R,C):: CALL SPRITE(#12,101,
16,160,C-40-(C<40)*256):: GO
TO 140
180 P=2 :: CALL POSITION(#11
,R,C):: CALL SPRITE(#13,101,
16,160,C+40+(C>216)*256):: G
OTO 140
```

If you can't figure out where all the money goes, this may be an eye-opener -

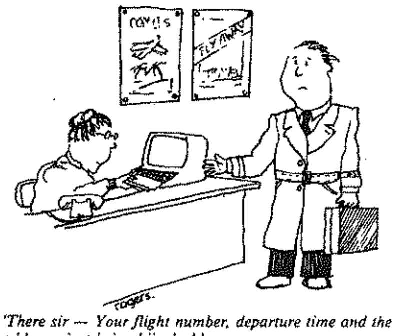
```
100 DISPLAY ERASE ALL AT(3,5
):"THE COST OF CREDIT" ! by
Jim Peterson
110 S,T,X=0 :: DISPLAY AT(8,
1):"AMOUNT OF PURCHASE?" ::
ACCEPT AT(8,21):A :: B,T=A :
: DISPLAY AT(10,1):"CREDIT C
ARD INTEREST RATE?" :: ACCEP
T AT(11,1):R
120 DISPLAY AT(13,1):"SAVING
S ACCOUNT INT. RATE?" :: ACC
EPT AT(14,1):SR
130 X=X+1 :: I=B*R/100/12 ::
 B=B+I :: T=T+I :: P=B/10 ::
 B=B-P :: S=S+P+S*SR/100/12
:: IF S<A THEN 130
140 D$="$"&STR$(INT((T-A+S-A
+.5)*100)/100)
150 DISPLAY AT(17,1):"If you
 had saved the amount":"of y
our minimum 10% of the":"bal
ance credit card payment":"e
ach month for";X;"months,"
160 DISPLAY AT(21,1):"and us
ed it to pay cash, you":"wou
ld have saved ";D$ :: GOTO 1
10
```

And this is one of the handiest routines I've seen in a long time -

```
10 !TURNS ALL NUMERALS AND P
UNCTUATION WHITE! BY HARRY W
ILHELM IN TWIN TIers UG NEWS
LETTER
20 !TURN IT OFF BY CALL LOAD
(-31804,0)::TURN IT ON BY CA
LL LOAD(-31804,63)
100 CALL INIT
110 CALL LOAD(16128,2,224,38
,0,2,0,8,17,2,1,63,36,2,2,0,
3,4,32,32,36,2,224,131,192,3
,128)
120 CALL LOAD(16164,240,240,
240)
130 CALL LOAD(-31804,63)
```

Memory full

Jim Peterson



'There sir — Your flight number, departure time and the odds against being hijacked.'

NUMBER SPEAKER

```
100 CALL CLEAR
110 PRINT TAB(7);"NUMBER SPE
AKER": : :"by Jim Peterson":
"      of Tigercub Software"
: : :
120 PRINT " This program wil
l print any":" number of les
s than 67":"digits in number
s and in"
130 PRINT "words, and will s
peak the":"words.": : : :" R
equires Terminal Emulator":"
II and Speech Synthesizer.":
 : :
140 CALL CHAR(39,"0000000000
301020")
150 OPEN #1:"SPEECH",OUTPUT
160 DIM HIGH$(21),NN$(23)
170 DATA ONE,TWO,THREE,FOUR,
FIVE,SIX,SEVEN,EIGHT,NINE
180 DATA TEN,ELEVEN,TWELVE,T
HIRTEEN,FOURTEEN,FIFTEEN,SIX
TEEN,SEVENTEEN,EIGHTEEN,NINE
TEEN
190 DATA TWENTY,THIRTY,FORTY
,FIFTY,SIXTY,SEVENTY,EIGHTY,
NINETY
200 DATA THOUSAND,MILLION,BI
LLION,TRILLION,QUADRILLION,Q
UINTILLION,SEXTILLION,SEPTIL
LION,OCTILLION,NONILLION
210 DATA DECILLION,UNDECILLI
ON,DUODECILLION,TREDECILLION
,QUATTUORDECILLION,QUINDECIL
LION,SEXTEDECILLION
220 DATA SEPTENDECILLION,OCT
ODECILLION,NOVEMDECILLION,VI
GINTILLION
230 FOR J=1 TO 9
240 READ ONE$(J)
250 NEXT J
260 FOR J=1 TO 10
270 READ TEEN$(J)
280 NEXT J
290 FOR J=1 TO 8
300 READ TEN$(J)
310 NEXT J
320 FOR J=1 TO 21
330 READ HIGH$(J)
340 NEXT J
350 PRINT : : :
360 PRINT #1:"NUMBER"
370 INPUT "NUMBER? ":N$
380 L=LEN(N$)
390 FOR J=1 TO L
400 IF POS("0123456789",SEG$
(N$,J,1),1)=0 THEN 360
410 NEXT J
420 IF (VAL(N$)<1)+(VAL(N$)<
>INT(VAL(N$)))THEN 360
430 IF L<67 THEN 470
440 PRINT "HEY! I CAN ONLY C
OUNT TO A":"VIGINTILLION!":
:
450 PRINT #1:"HAY I CAN ONLY
 COUNT TO A VIGINTILLION"
460 GOTO 360
470 IF VAL(N$)>0 THEN 510
480 PRINT : :"ZERO": :
490 PRINT #1:"ZERO"
500 GOTO 360
510 IF L/3=INT(L/3)THEN 540
520 N$="0"&N$
530 GOTO 380
540 X=L/3
550 FOR J=1 TO L STEP 3
560 JJ=JJ+1
570 NN$(JJ)=SEG$(N$,J,3)
580 IF J>1 THEN 610
590 P$=STR$(VAL(NN$(JJ)))
600 GOTO 620
610 P$=P$&"'"&NN$(JJ)
620 NEXT J
630 PRINT : : :P$: : :
640 FOR J=1 TO X
650 GOSUB 670
660 GOTO 1150
670 IF VAL(NN$(J))<>0 THEN 7
10
680 A$=""
690 FLAG=1
700 GOTO 1140
710 FLAG=0
720 H=VAL(SEG$(NN$(J),1,1))
730 T=VAL(SEG$(NN$(J),2,2))
740 TT=VAL(SEG$(NN$(J),2,1))
-1
750 VV=VAL(SEG$(NN$(J),3,1))
760 IF T=0 THEN 1000
770 IF T>9 THEN 810
780 A$=ONE$(T)
790 SP$=A$
800 GOTO 1000
810 IF T>19 THEN 880
820 A$=TEEN$(T-9)
830 IF T<>19 THEN 860
840 SP$="NINE TEEN"
850 GOTO 1000
860 SP$=A$
870 GOTO 1000
880 IF VV<>0 THEN 950
890 A$=TEN$(TT)
900 IF TT<>8 THEN 930
910 SP$="NINE TEE"
920 GOTO 1000
930 SP$=A$
940 GOTO 1000
950 A$=TEN$(TT)&"-"&ONE$(VV)
960 IF TT<>8 THEN 990
970 SP$="NINE TEE"&ONE$(VV)
980 GOTO 1000
990 SP$=A$
1000 IF H=0 THEN 1080
1010 IF T=0 THEN 1050
1020 A$=ONE$(H)&" HUNDRED &
"&A$
1030 SP$=ONE$(H)&" HUNDRED &
"&SP$
1040 GOTO 1140
1050 A$=ONE$(H)&" HUNDRED"
1060 SP$=A$
1070 GOTO 1140
1080 IF (J<X)+(T=0)+(VAL(N$)
<100)THEN 1140
1090 A$=" & "&A$
1100 IF (TT<>8)*(T<>19)THEN
1130
1110 SP$=" & "&SP$
1120 GOTO 1140
1130 SP$=A$
1140 RETURN
1150 PRINT A$
1160 IF FLAG=1 THEN 1200
1170 PRINT #1:SP$
1180 PRINT HIGH$(X-J)
1190 PRINT #1:HIGH$(X-J)
1200 GOSUB 670
1210 NEXT J
1230 A$=""
1240 JJ=0
1260 P$=""
1270 FOR D=1 TO 500
1280 NEXT D
1290 GOTO 350
```

EXPLANATION OF PROGRAM LINES

Note how the ":" is used to get 3 lines of text into one PRINT statement.
In line 140, character ASCII 39, the apostrophe, is redefined as a comma because a true comma could not be used in the string P$ in line 610.
Line 150 activates the Speech Synthesizer
The variable names HIGH$ and NN$ will contain more than 11 subscripts, so they must be dimensioned in advance.
Lines 230-250 define ONE$(1) through ONE$(9) as being the words ONE through NINE from the DATA statement in 170.
Similarly, The words in DATA line 180 are read into TEEN$, those in DATA lines 190 into TEN$, and those in DATA lines 200-220 into HIGH$.
Line 360 speaks the word "NUMBER" and 370 requests INPUT of the user's no. in the form of a string instead of a numeric value so that a large number won't print out in exponential notation. However, if any non-numeric characters are mistakenly entered in N$, taking the VAL of it in 420 would cause the program to crash. So, lines 380-410 check thru N$ character by character. If a character is not found in the string "0123456789", the value of POS is 0 and the program requests another number.
Line 420 edits for invalid negative number or a number containing a decimal, and line 430 edits for a number more than 66 digits long.
If N$ is input as 0, or a string of 0's, lines 470-500 print and speak "ZERO" and go back for another. Otherwise, line 510 checks whether N$ can be evenly divided into sets of 3 digits; if not, 520-530 add a 0 in front of it and go back to measure its length again - and again, if necessary. In 540, X is the number of sets of 3 digits in N$.
The loop 550-610 goes through the length of N$ in steps of 3, using JJ as a counter to assign each 3-digit segment to a subscript of NN$. At the same time, the string P$ is built up as a representation of the number having each 3-digit set of numerals, except the first set, preceded by a comma (as redefined in line 140), as large numbers are usually written. It is then printed by line 630.
Now, the loop 640-1210 goes through the 3-digit sets in sequence, each time going to the subroutine 670-1140 to compute what should be printed and spoken, then jumping to 1150 to do so. A$ is the word to be printed, SP$ is the word to be spoken.
In line 670, if the set consists of all 0's the program drops through to 680, A$ will be a blank, a flag is set to 1 and we jump to 1140 which returns us to 660 and thence to 1150 where a blank is printed and the FLAG value of 1 causes us to jump over the speech routine and the printing/speaking of HIGH$.
But if the value of the set is more than 0, line 670 goes to 710 which makes sure that the FLAG is reset to 0, and then determines the values of the numerals in the set. H is the 1st numeral, T is the 2nd-3rd numerals, TT is the 2nd numeral and VV is the 3rd.
In line 760, if T (2nd-3rd numerals) is 00 the set must be an even hundred so we can skip over the checking of TEEN$ and TEN$ to line 1000 (which will drop us through to 1010 because we have already determined in 670 that all 3 digits are not 0).

Continued on P24

PUTTING IT ALL TOGETHER #1

by Jim Peterson

The hardest part of learning to program is not in learning what the various commands do - it is in learning how to put them all together to do what you want them to do!

Key in this simple routine and run it, to see what it does. Then read the explanations of each line and see how they do what they do!

Clear the screen and insure that selection of random numbers will be different each time. RND gives a random number between 0 and .999... Therefore RND*5 gives a random number between 0 and 4.99999.... INT drops the decimal part of a number, so INT(RND*5) gives a random whole number between 0 and 4, and INT(RND*5+2) gives a whole number between 2 and 6.

The first time the program is run, B2 has never been given a value, so it equals 0. Since B is between 2 and 6, it does not equal B2; the program continues, B2 is given the value of B.

When the next random problem is selected, if the same value happens to be selected again for B, B2 will equal B and the program will go back to make another selection. This prevents the "stupid computer syndrome" of the same question being asked twice in a row.

B was the number of boys in the first question. In the same way, F is selected to be the number of frogs that one boy can catch in one day, and D is selected to be the number of days in the first question.

In line 140, F is multiplied by B by D to find the total number of frogs in the first question. This method insures that all calculations will be in whole numbers.

In lines 150 and 160, BB and DD are randomly selected as the numbers of boys and days in the second question. These values are rejected if they are the same as the previous time or if they are the same as were selected for the first question.

Line 170 then multiplies the number of frogs that one boy can catch in one day by the number of boys and days in the second question. The rest is merely a matter of screen formatting. Note that numeric variables can be incorporated in string text, by separating them with semicolons; they will print out their value with a blank space before and after. Note also that numeric calculations can be performed within the DISPLAY AT statements, and will print the numeric result of the calculation preceded and followed by a blank space. o

```
100 CALL CLEAR :: RANDOMIZE
110 B=INT(5*RND+2):: IF B=B2
 THEN 110 ELSE B2=B
120 F=INT(5*RND+2):: IF F=F2
 THEN 120 ELSE F2=F
130 D=INT(5*RND+2):: IF D=D2
 THEN 130 ELSE D2=D
140 X=F*B*D
150 BB=INT(5*RND+2):: IF BB=
BB2 OR BB=B THEN 150 ELSE BB
2=BB
160 DD=INT(5*RND+2):: IF DD=
DD2 OR DD=D THEN 160 ELSE DD
2=DD
170 F=F*BB*DD
180 DISPLAY AT(3,1)ERASE ALL
:"IF";B;"BOYS CAN CATCH";X;"
FROGS IN";D;"DAYS,"
190 DISPLAY AT(6,1):"HOW MAN
Y FROGS CAN";BB;"BOYS":"CATC
H IN";DD;"DAYS?"
210 ACCEPT AT(7,19):Q
220 IF Q=F THEN DISPLAY AT(9
,1):"THAT'S RIGHT!" :: GOTO
110
230 DISPLAY AT(9,1):"NO, THA
T'S WRONG."
240 DISPLAY AT(11,1):"IF";B;
"BOYS CAN CATCH";X;"FROGS IN
";D;"DAYS"
250 DISPLAY AT(13,1):"THEN O
NE BOY CAN CATCH";X/B;"FROGS
 IN";D;"DAYS"
260 DISPLAY AT(15,1):"AND ON
E BOY CAN CATCH";X/B/D;"FROG
S IN ONE DAY."
270 DISPLAY AT(17,1):"SO, IF
 ONE BOY CAN CATCH";X/B/D;"F
ROGS IN ONE DAY,"
280 DISPLAY AT(19,1):"THEN";
BB;"BOYS CAN CATCH";X/B/D*BB
;"FROGS IN ONE DAY"
290 DISPLAY AT(21,1):"AND";B
B;"BOYS CAN CATCH";X/B/D*BB*
DD;"FROGS IN";DD;"DAYS."300
DISPLAY AT(24,1):"PRESS ANY
KEY" :: CALL KEY(0,K,S):: IF
 S=0 THEN 300 ELSE 110
```

From P23

In line 770, if T is more than 9 we can skip over the ONE$; otherwise, the value of T will pick out the correct subscript of ONE$ (from 170 and 230-250) to be printed and, in line 790, to be spoken. In 810, from 770, if T is more than 19 we can similarly skip over TEEN$; otherwise, the value of T picks out the proper subscript of TEEN$ (see 180 and 260-280).
In most cases SP$, the word to be spoken, can be defined as the same as A$, the word to be printed, but the words NINETEEN and NINETY would be mispronounced (that E in the middle confuses the computer!) so we must define them separately.
In line 880, from 810, if VV (the 3rd digit) is 0 we don't need ONE$, so in 890 the value of TT (the 2nd digit) gives us the correct subscript of TEN$ (lines 190 and 290-310); else, 880 takes us to 950 where TT again picks out the right word for TEN$, a dash (-) is added after it, and then the value of VV picks out the correct word for ONE$.
In all cases, the program jumps to line 1000. If H (the 1st digit of the set) is 0, we do not need the word HUNDRED so we skip to 1080. If T in 760 was 0 we need only the HUNDRED, so go to 1050. Else, H gives us the subscript of ONE$ to be placed in front of the word HUNDRED in lines 1020-1030 and both of these words are placed in front of whatever A$ and SP$ already consist of from 780, 820 or 950.
Line 1080, from 1000, checks to see if we are on the last set of 3. If so, and if T was more than 0 and N$ was more than 99, line 1090 places the symbol & before the printed number; SP$ in 1110 pronounces & as "AND".
In all cases, the routine goes to 1140 which returns it to 660 and thence to 1150, which prints out the words. We have already described what line 1160 does. Line 1170 speaks the words that were just printed.
Line 1180 finds the correct subscript for HIGH$ by subtracting the current value of J (the 550-1210 loop we are in) from the value of X (the total number of loops to be made), and then prints and speaks the correct subscript of HIGH$ (from 200-220 and 320-340).
After the first pass through the loop we enter the 670-1140 subroutine from 1200 instead of from 650.
When the loop has been completed, and the entire number has been spoken and and printed, we must cancel out the values of A$, JJ and P$, which were formed by adding onto themselves, before we pause briefly and then go back to ask for the next number. o

Programs That Write Programs

Part 2

by Jim Peterson

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

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

Oh - that puzzle in last month's article? Try creating those DATA statements with this LINEWRITER program!
Now, let's get down to business and learn how to do all this.
First, let's write a program that will write a program to list the token codes that you need to use to write a program that will write a program -

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

Key that in, RUN it, then enter NEW, then MERGE DSK1.TOKENLIST. Now LIST it and you will see a list of ASCII codes 129 through 254 and their token meanings. Delete lines 171 through 175, 185, 198, 226 through 231, and 242. Change the definition of 199 to QUOTED STRING, of 200 to UNQUOTED STRING, and 201 to LINE NUMBER, and add line 255 !END OF FILE. You don't need all those exclamation points, so change the program to a DIS/VAR 80 file by LIST "DSK1.TOKENLIST". Then key in this little routine.

```
100 OPEN #1:"DSK1.TOKENLIST"
,INPUT :: OPEN #2:"PIO" !or
whatever
110 PRINT #2:CHR$(27);"N";CH
R$(6)
120 LINPUT #1:A$ :: PRINT #2
:TAB(10);SEG$(A$,1,4)&SEG$(A
$,6,255):: IF EOF(1)<>1 THEN
 120 ELSE CLOSE #1 :: END
```

RUN it, and print out a list of all the token codes. Keep it handy, you'll be needing it.
Notice that every Extended Basic statement has its own ASCII token code - even the ones you perhaps never heard of, such as LET and GO. Notice also that every keyboard symbol which affects program execution, such as + and =, has its own ASCII token code which is NOT the same as its keyboard ASCII code. And notice that the double colon, used as a separator in Extended Basic multi-statement lines, has its own token.
Now, let's take a look at how a MERGE format program is put together. This routine will do that for you - and you will also find it very useful in debugging the MERGE programs you are going to write.

```
100 DISPLAY AT(3,5)ERASE ALL
:"D/V 163 FILE READER": :"
    by Jim Peterson": : :" T
o edit a file saved or":"cre
ated in MERGE format."
110 DISPLAY AT(12,1):"Output
 to? (S/P)S":" (S)creen":" (
P)rinter" :: ACCEPT AT(12,17
)SIZE(-1)VALIDATE("SP"):Q$
120 IF Q$="P" THEN DISPLAY A
T(14,1):"PRINTER? PIO" :: AC
CEPT AT(14,10)SIZE(-18):P$ :
: D=2 :: OPEN #2:P$
130 DATA ELSE,"::",!,IF,GO,G
OTO,GOSUB,RETURN,DEF,DIM,END
,FOR,LET,BREAK,UNBREAK,TRACE
140 DATA UNTRACE,INPUT,DATA,
RESTORE,RANDOMIZE,NEXT,READ,
STOP,DELETE,REM,ON,PRINT,CAL
L
150 DATA OPTION,OPEN,CLOSE,S
UB,DISPLAY,IMAGE,ACCEPT,ERRO
R,WARNING,SUBEXIT,SUBEND,RUN
,LINPUT
160 DATA ,,,,,THEN,TO,STEP,"
,",";",":",),(,&,,OR,AND,XOR
,NOT,=,<,>,+,-,*,/,^,
170 DATA QUOTED STRING,UNQUO
TED STRING,LINE NUMBER,EOF,A
BS,ATN,COS,EXP,INT,LOG,SGN,S
IN
180 DATA SQR,TAN,LEN,CHR$,RN
D,SEG$,POS,VAL,STR$,ASC,PI,R
EC,MAX,MIN,RPT$,,,,,,,NUMERI
C,DIGIT
190 DATA UALPHA,SIZE,ALL,USI
NG,BEEP,ERASE,AT,BASE,,VARIA
BLE,RELATIVE,INTERNAL,SEQUEN
TIAL,OUTPUT,UPDATE,APPEND
200 DATA FIXED,PERMANENT,TAB
,#,VALIDATE
210 DIM T$(126):: FOR J=1 TO
 126 :: READ T$(J):: NEXT J
:: E$(1)="LINE NOT CLOSED WI
TH CHR$(0)"
220 DISPLAY AT(16,1):"FILENA
ME? DSK" :: ACCEPT AT(16,14)
:F$
230 ON ERROR 240 :: OPEN #1:
"DSK"&F$,VARIABLE 163,INPUT
:: GOTO 250
240 DISPLAY AT(20,1):"I/O ER
ROR" :: ON ERROR STOP :: RET
URN 220
250 ON ERROR 260 :: LINPUT #
1:A$ :: X=ASC(SEG$(A$,1,1)):
: Y=ASC(SEG$(A$,2,1)):: IF X
=255 AND Y=255 THEN 410 ELSE
 270
260 PRINT #D:"FILE NOT CLOSE
D PROPERLY":"WITH CHR$(255),
CHR$(255) ?" :: STOP
270 PRINT #D:"LINE NUMBER":X
;"TIMES 256=";X*256:Y;"PLUS"
;Y;"=";X*256+Y
280 FOR J=3 TO LEN(A$)-1 ::
X=ASC(SEG$(A$,J,1))
290 IF X=201 THEN PRINT #D:X
;"LINE NUMBER" :: X=ASC(SEG$
(A$,J+1,1)):: Y=ASC(SEG$(A$,
J+2,1)):: J=J+2 :: PRINT #D:
X;"TIMES 256=";X*256:Y;"PLUS
";Y;"=";X*256+Y
300 IF X=199 THEN PRINT #D:X
;"QUOTED STRING" ELSE IF X=2
00 THEN PRINT #D:X;"UNQUOTED
 STRING" ELSE GOTO 360
310 J=J+1 :: X=ASC(SEG$(A$,J
,1)):: PRINT #D:X;"OF";X;"CH
ARACTERS"
320 ON ERROR 340 :: FOR L=1
TO X :: Y=ASC(SEG$(A$,J+L,1)
):: PRINT #D:Y;CHR$(Y):: IF
Y<32 OR Y>126 THEN PRINT #D:
"UNPRINTABLE CHAR - ERROR?"
330 NEXT L :: J=J+X :: GOTO
370
340 PRINT #D:"ERROR! INSUFFI
CIENT BYTES IN":"STRING" ::
IF ASC(SEG$(A$,LEN(A$),1))<>
0 THEN PRINT #D:E$(1)
350 ON ERROR STOP :: RETURN
250
360 IF X<129 THEN PRINT #D:X
;CHR$(X);" VARIABLE NAME" EL
SE PRINT #D:X;T$(X-128)
370 CALL KEY(0,K,S):: IF S=0
 THEN 390
380 CALL KEY(0,K2,S2):: IF S
2<1 THEN 380
390 NEXT J :: IF ASC(SEG$(A$
,J,1))=0 THEN PRINT #D:"0 EN
D OF LINE" ELSE PRINT #D:E$(
1)
400 GOTO 250
410 PRINT #D:X:X;"END OF FIL
E" :: CLOSE #1 :: STOP
```

Next month - how to do it!

WORD WRAP AND FILL ROUTINES

Below is a short program that illustrates two very useful routines for handling displays of string data. The first routine automatically right justifies each line of text to give a neat appearance to things like instructions, etc. without having to "count out the spaces."

The second routine will automatically calculate the longest possible portion of the string that can be presented on one line and breaks the line up accordingly. This prevents that annoying break up of words that sometimes occurs when printing long strings.

The actual routines themselves appear first in both BASIC and X-BASIC. A short TI-BASIC demo program follows.

FSUBROUTINES

```
10298 REM  *FILL/B*
10299 REM
10300 FOR XL=1 TO LEN(M$)
10301 IF LEN(M$)=28-XI THEN
10307
10302 IF SEG$(M$,XL,1)<>" "
THEN 10305
10303 M$=SEG$(M$,1,XL)&" "&
SEG$(M$,XL+1,LEN(M$)-XL)
10304 XL=XL+1
10305 NEXT XL
10306 GOTO 10300
10307 RETURN
10308 REM

10800 SUB FILL(R,I,M$)
10801 FOR X=1 TO LEN(M$)
10802 IF LEN(M$)=28-I THEN
10804 ELSE IF SEG$(M$,X,1
)=" " THEN M$=SEG$(M$,1,X)
&" "&SE G$(M$,X+1,LEN(M$
)-X):: X=X+1
10803 NEXT X :: GOTO 10801
10804 DISPLAY AT(R,29-LEN(M
$)):M$
10805 SUBEND
```

WRAP SUBROUTINES

```
12598 REM *WRAP/B*
12599 REM
12600 X1=0
12601 M$=M$&" "
12602 X2=POS(M$," ",X1+1)
12603 PRINT SEG$(M$,X1+1,X2
-X1);:: IF X2=LEN(M$)THEN
SUBEXIT
12604 IF X2=LEN(M$)THEN 126
07
12605 X1=X2
12606 GOTO 12602
12607 RETURN

12598 ! *WRAP/X*
12599 !
12600 SUB WRAP(M$)
12601 M$=M$&" " :: X1=0
12602 X2=POS(M$," ",X1+1)
12603 PRINT SEG$(M$,X1+1,X2
-X1);:: IF X2=LEN(M$)THEN
SUBEXIT
12604 X1=X2 :: GOTO 12602
12605 SUBEND
```

DEMO PROGRAM

```
100 REM ***************
110 REM * *
120 REM * FILL & WRAP *
130 REM * *
140 REM * SUB DEMOS *
150 REM * *
160 REM ***************
170 REM
180 REM SUBFILE99
190 REM 04/85
200 REM
210 REM *HOUSEKEEPING*
220 REM
230 L$="____________________
__________"
240 CALL CLEAR
250 RESTORE 970
260 FOR L=1 TO 17
270 READ M$
280 PRINT TAB(7);M$
290 NEXT L
300 REM
310 INPUT "   PRESS ENTER T
O START":A$
320 REM
330 REM *SELECT DEMO*
340 REM
350 CALL CLEAR
360 PRINT "SELECT DEMO:"
370 PRINT "____________"
380 PRINT
390 PRINT
400 INPUT "<W>RAP, <F>ILL O
R <Q>UIT? ":A$
410 IF (A$<>"W")*(A$<>"F")T
HEN 920
420 IF A$="W" THEN 690
430 REM
440 REM *FILL DEMO*
450 REM
460 CALL CLEAR
470 PRINT "FILL DEMO"
480 PRINT "_________"
490 PRINT
500 PRINT
510 PRINT L$
520 PRINT
530 RESTORE 1100
540 FOR L=1 TO 10
550 READ M$
560 GOSUB 1290
570 PRINT M$
580 NEXT L
590 REM
600 PRINT
610 PRINT L$
620 PRINT
630 PRINT
640 INPUT "HIT <CR> KEY":A$
650 GOTO 350
660 REM
670 REM *WRAP DEMO*
680 REM
690 CALL CLEAR
700 PRINT "WRAP DEMO"
710 PRINT "_________"
720 PRINT
730 PRINT
740 PRINT L$
750 PRINT
760 RESTORE 1240
770 FOR L=1 TO 2
780 READ M$
790 GOSUB 1400
800 NEXT L
810 REM
820 PRINT
830 PRINT
840 PRINT L$
850 PRINT
860 PRINT
870 INPUT "HIT <CR> KEY":A$
880 GOTO 350
890 REM
900 REM *QUIT PROGRAM*
910 REM
920 CALL CLEAR
930 END
940 REM
950 REM *TITLE DATA*
960 REM
970 DATA "***************"
980 DATA "*             *"
990 DATA "* FILL & WRAP *"
1000 DATA "*             *"
1010 DATA "*  SUB DEMOS  *"
1020 DATA "*             *"
1030 DATA "***************"
1040 DATA ,,,"   SUBFILE99"
1050 DATA "      04/85",,,,,
1060 REM
1070 REM
1080 REM *FILL DATA*
1090 REM
1100 DATA THIS IS AN EXAMPL
E OF THE
1110 DATA FILL ROUTINE IN T
I-BASIC.
1120 DATA AS YOU CAN SEE -
IT IS NOT
1130 DATA THE FASTEST ROUTI
NE AROUND
1140 DATA BUT IT GETS THE J
OB DONE!
1150 DATA IT COMES IN VERY
HANDY WHEN
1160 DATA YOU WANT TO CREAT
E A NEAT
1170 DATA LOOKING SCREEN LA
YOUT W/O
1180 DATA ALL THE HASSLE OF
COUNTING
1190 DATA SPACES WHEN ENTER
ING DATA!
1200 REM
1210 REM
1220 REM *WRAP DATA*
1230 REM
1240 DATA THIS LINE WAS ORI
GINALLY VERY LONG AND IT
HAS BEEN SHORTENED SO THAT
IT WILL APPEAR ON THE SCRE
EN WITHOUT CUTTING ANY
WORDS OFF.
1250 DATA IT'S REALLY AMAZI
NG WHAT CAN BE DONE WITH A
LITTLE PATIENCE AND PERSERV
ERANCE!
1260 REM
1270 REM  *FILL/B*
1280 REM
1290 FOR XL=1 TO LEN(M$)
1300 IF LEN(M$)=28-XI THEN
1360
1310 IF SEG$(M$,XL,1)<>" "
THEN 1340
1320 M$=SEG$(M$,1,XL)&" "&S
EG$(M$,XL+1,LEN(M$)-XL)
1330 XL=XL+1
1340 NEXT XL
1350 GOTO 1290
1360 RETURN
1370 REM
1380 REM *WRAP/B*
1390 REM
1400 X1=0
1410 M$=M$&" "
1420 X2=POS(M$," ",X1+1)
1430 PRINT SEG$(M$,X1+1,X2-
X1);:: IF X2=LEN(M$)THEN
RETURN
1440 IF X2=LEN(M$)THEN 1470
1450 X1=X2
1460 GOTO 1420
1470 RETURN
```

REGIONAL GROUP NEWS

GLEBE Regional Group.
8th October 1987, 8pm, 43 Boyce St, Glebe. Contact Mike Slattery, 692 0559.

Regular meetings on the Thursday evening following the first Saturday of the month.

LIVERPOOL REGIONAL GROUP - Contact Arto Heino 603-8956 for more info.

LIVERPOOL REGIONAL GROUP MEETING 9/10/87 WILL BE AT LARRY'S PLACE, 34 Colechin St Yagoona West at 7.30pm.

Will be the NEW SUPER EXT-BASIC DEMO. Plus a lot of other products from the U.S.A.

Regular meeting date is the Friday following the TIsHUG general meeting (first Saturday) at 7.30pm.

CENTRAL COAST Regional Group.

Meetings are normally held on Second Saturday of each month at 6.30 pm at
Toukley Tennis Club hall,
Header St, Toukley.
Next meeting 10th October 1987.

Contact Russell Welham (043 92 4000)

CARLINGFORD Regional Group.

The next Carlingford Regional Group meeting will be 21st October 1987. Commencing Time 7.30 pm

The meeting will be held at the home of Percy Harrison, 3 Storey St, Ryde. Phone 808 3181.

Regular meetings are third Wednesday of each month.

ILLAWARRA Regional Group.

Next meeting 19/10/87 7.30pm, Keiraville Public School, Gipps Rd, Keiraville. Opposite Keiraville Shopping centre.

Regular meetings are third Monday of each month except January.

NORTHERN SUBURBS Regional Group.

Contact Dennis Norman on 452 3920 or Dick Warburton on 918 8132 for further information.

15/10/87 8pm.

Regular meetings are third or fourth Thursday of the month.

BANKSTOWN Regional Group.

Next meeting date unknown to EDITOR but if interested contact Peter Pedersen, (02) 772 2396. Meeting are held at 11 Bastille Close, Padstow Heights.

Banana Coast (Coffs Harbour area) Regional Group.

For information on meetings of the Banana Coast group contact Keir Wells at 9 Tamarind Drive, Bellingen, phone 066 55 1487.

ILLAWARRA REGIONAL GROUP

The Illawarra Regional Group has its headquarters in Wollongong, some 80km south of Sydney. With a population in excess of 250,000 it vies with Newcastle as the second city of New South Wales.

The major industries are coal mining, steel mills and copper refining. The largest steel mills in Australia are in Pt Kembla. Economic downturns in the steel and coal industries has seen Wollongong become tourist oriented. So much so that it is expected to be the biggest employer by the turn of the century.

The regional group formed about 4 years ago in a local distibutor's shop, just before the TI pullout in 1983. Five of the original members still form part of the group.

It has been a self help group right from the start with members contributing financially, in addition to the normal club fees,to purchase software and the like. Number have generally been so many that the shop was outgrown and the need arose to use rented premises. Again members of the group bearing the cost. It has also served the purposes of the TI community in Wollongong reducing the need to travel to Sydney.

The most popular attraction the group developed has been its libraries. There are 62 modules, 140 cassettes, 50 disks, 35 periodicals, a full set of Micropendiums, 100 books and miscellaneous hardware. Modules, cassettes and books are the order of popularity.

Over the years members have lead various technical and programming skills discussions.

Members who form the organising committee are:- Rolf Schreiber, Lou Amadio, George Meldrum, Geoff Trott, Phil Thompson and Bob Montgomery, the current editor of the TIsHUG TND.

Meetings are held the third Monday of the month at Keiraville Public School, except January, starting at 7:30pm.

SUTHERLAND REGIONAL USERS GROUP

Although it has been a couple of months between Regional Reports the Sutherland Group has been far from idle.

A further two ramdisks have now been completed, thanks to the technical assistance provided by Derek Wilkinson, who persevered with us two amateur techos for two nights. Thanks also to Beverley for the loverly supper.

Both systems are now being put to good use.

Like most ramdisk owners we had difficulty in finding suitable battery holders for the three AAA batteries. A stock was eventually located at Radiospares Components P/L, Rosebery.

Part No. 489-891 $1.95 ea. retail.

The group is arranging for a Ti-Writer tutorial to be conducted by new member Jack Krupski. This will now commence at the October meeting due to a number of absences, on holidays, during September.

Meeting nights are the third Friday of each month starting at 7.30pm.

Regards

Peter Young.