8712 (042)

TISHUG NEWS DIGEST

Focusing on the TI-99/4A Home Computer

Volume 6, Number 11 **December 1987**

Registered by Australia Post - Publication No. NBH5933

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

TIsHUG NEWS DIGEST

TIsHUG(Australia) Ltd
TIsHUG NEWS DIGEST

DECEMBER 1987

Correspondence to:

PO Box 214
REDFERN NSW 2016

TIsHUG NEWS DIGEST ISSN 0819-1984

CONTENTS PAGE

The Editorial committee wishes to thank those members who have contributed, to the pages of the TND, this year. Without your support this newsletter may not have been as interesting as it has been. I would also like to thank those who regularly helped to do the paste-up.

This year has been somewhat of a challenge for the committee. A number of new ventures were undertaken. Probably the most successful of them was the Ramdisk. That is not to say that other projects were not equally successful. While the hardware side of the club has flourished there has been a dearth in the siftware area. The notable exception being PICASSO. The club has attempted to correct this imbalance by conducting a software competition. Make it your New Year's resolution to submit a program.

You should have found an insert in this issue with instructions of how to nominate for next year's directors. Take time over the holiday period to consider your position and nominate.

Note that the December meeting is the Christmas Party. Bring along a plate of party food to share with others.

Have a joyous Christmas and a prosperous New Year.

Bob Montgomery

CO-ORDINATOR'S REPORT

.... Chris Buttner

Our final full-day tutorial this year was successfully completed last month. I was pleasantly surprised to see so many of you attending the Assembly Language sessions and coming to grips with what has seemed to so many to be an impossible task. There is obviously a lesson to be learned and that must surely be that perseverance will win through. If you have not yet had a go, make a start and discuss your problems with other club members. It is a worthwhile learning experience and also a very goods means of making new friendships.

In a few weeks Christmas will be upon us and no doubt a number of you will put your computers away to get into full swing with the children during the vacation. The club will be doing much the same - "closing shop" until we gear up in February for another exciting year.

The annual general meeting is very important for the club because we are now bound by our Memorandum and Articles of Association and the various sections of the Companies Code (NSW).

Arrangements have been made for the Memorandum and Articles to be printed in booklet form and these will be distributed to all financial members. It is important you understand the club rules so please set aside time to read it.

In case you are wondering what the legal effect of the incorporation means I shall try to explain (I hope) in layman's terms.

Our old group TISHUG is no more. It has ceased and we have had to open new bank accounts and make new arrangements with suppliers. We are a public company and you are the members of the company although nobody owns any shares in the company. Our company structure makes it very important that you address correspondence to the Secretary.

As a company, we have five Directors. The first change you will probably notice is that the nominations for Directors for the 1988 year are as Directors - no more nominations for positions. Those who are successfully elected as Directors must decide who will be Chairman, (Co-Ordinator), Secretary, Treasurer and so on. As you can see, it is important you consider carefully who you vote for because the board of Directors needs a breadth of experience and expertise for the club to operate successfully.

If there are more nominations than vacancies for Directors, there will be a ballot. We do not have an option to elect directors by a show of hands as in the past.

The Directors are the people who are responsible for the running and good management of the club. However, the Board has the power to appoint committees. This allows us to be less centralised in our management and also give those who do the work more flexibility.

At the moment there are two committees: editorial and library: to handle the publishing of the club magazine and run the software and book/magazine library.

This month we will have a small Christmas Party. It is a bring-a-plate affair and I hope as many of you as possible will attend. In addition to the festivities, Cyril has assured me there will be some super-duper savings in the shop. 1987 has been a good year and despite a smaller membership, we are still in a sound financial position. One encouraging note is the number of new members slowly trickling in. TI'ers it seems are somewhat like old soldiers.

On behalf of all the old committee members and fellow directors, I wish each of you a safe and joyous time at Christmas and in the New Year. o

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. o

* Secretary's Notebook *

Despite a bit of rain around lunch time, the last meeting seemed to be enjoyed by all who attended. There was certainly a lot to do and see on the day and special thanks must go that willing band of "doers", Ross, John, Russell, Shane, Peter, George and Arto - sorry if I missed anyone - for their untiring efforts throughout the day. Not, of course, to forget Cyril and his band of shop-keepers, who toiled throughout the day making sure you were able to get what you wanted at the shop.

Speaking of the shop, Cyril tells me that there was certainly a rush on the Data Perfect disks, with him nearly selling out. And why not? At $12 a box of 10 these were a bargain. Watch for more disk specials in coming months.

Version 6.4 of the RAM operating system proved popular with most in attendance picking up a copy. If you missed out, make sure you pick up a copy, as this is excellent software.

The next meeting, to be held on Saturday, 5 December will be the last for this year and will take the form of a picnic/barbeque get together. The grounds of Woodstock are very spacious, with plenty of BBQ and other facilities, so bring along your eats and drinks and make a great afternoon of it. The shop will be open during the afternoon - at time of writing - we have the large upstairs room booked, and the publications library facility will also be available.

Tony McGovern has written to advise that Version 4.0 of Funnelweb is now available and a distribution copy has been received. Tony also advised, that his son, Will, has temporarily ceased work on his TI-MS/DOS file transfer program until after the HSC. You may have read of this in Micropendium.

One new member is welcomed this month. He is Colin McKay from Campbelltown. Hope you enjoy your stay with us Colin and that you can get to some of the meetings.

TIsHUG is now a company, and in accordance with its Articles of Association an election for 5 Directors will be held at the AGM on 6 February, 1988. Paragraph 16(1) of the Articles of Association states:-

"Nominations for the office of Director shall be delivered to the Secretary by 8.00pm on the twenty first (21) day prior to the day fixed by the Board for the annual election of Directors."

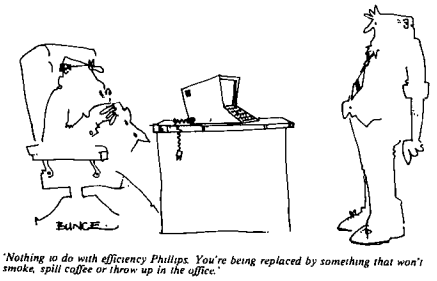
Paragraph 17(b) states:-

"Nominations for election of the Directors shall be made in writing and signed by two (2) members of the Club and by the nominee who shall signify his consent to the nomination."

In accordance with Paragraph 16(1), nominations for the office of Director will close with the Secretary, at 8pm on 16 January, 1988, while in accordance with Paragraph 17(b), a nomination form is contained elsewhere in this issue.

Please give this important matter some thought over the next few weeks.

That's it for now. Hope to see you at the next meeting. o



'Nothing to do with efficiency Philips. You're being replaced by something that won't smoke, spill coffee or throw up in the office.'

LETTERS TO THE EDITOR

This is just a few lines to express my gratitude to the club for the many benefits received by many members in the course of each year. Not enough is said in praise of this fact for very little cost to each member. The time and effort put in by many committee officers and lay members in the many activities needed to keep all the different things going, much of which is unknown to members and has to go on to enrich our lives, in the realm of computing.

I myself have been very conscious since joing the club of these many benefits but like so many others have taken things for granted until now, before writing some praise to those who have made all things possible.

With the help of the many articles in the newsdigest, like the 32K memory expansion, disk drives, TI controller, etc. etc, I have been able to expand my system beyond my wildest dreams, and at very moderate cost. I am now a retired "oldie" who took up computing by accident. After advising my nephew to have a look at computers as a career opportunity, I decided to have a look myself, and was advised to look at the TI99/4A by one of the members. Believe me, this was the best advice I have ever got!

Recently, for about a period of six weeks, I was without my computer. It was heart-breaking being unable to find the very serious fault - which I unknowing at this time - had contributed to.

What to do?

I live on the mid North Coast, near Taree, with a Regional Group of one - me! So you can see my plight, how am I going to get it up and running?

As I happened to be going to Sydney to visit my daughter, I rang John Paine, Technical Co-Ordinator, and he agreed to see me that night at his home with my non working TI etc. I arrived at John's at 7pm and left very close to 11pm after a long series of faults had been diagnosed, corrected, parts replaced and a complete test done on my system. Although this was all very time consuming, John made it all seem easy because of his expertise.

Another member also arrived at John's that night with a problem. That fault was also traced but because of unavailability of parts, repair had to be held over to another day. I also mention that this is the second time that John has been able to help me out with a problem. A fault in my home built disk drives was analysed and the faulty part able to be replaced.

Please let it be known to all concerned of my gratitude to John and to all officials, past and present, for the help that I have received, too many to detail, but which I humbly acknowledge.

Sincerely,
Bill Walker
Hallidays Point

★

Dear Editor,

Some time ago, a copy of Micro Pendium came out with what was suppose to be a comprehensive listing of TI User Groups around the world, but to my dismay, TIsHUG was not included in that listing. The reason for this letter, is to mention my discontent, for the lack of Promotional work, ADVERTISING being being done for our much loved repected Sydney based User Group.

If our group is to grow in numbers again, then it must be SEEN HEARD, but it would appear that the responsibility of that task has not been given to a Committee Member or deligated from one to a member of this group. This is a crying shame. There are hundreds of TI Users out there, who don't know our group exists, and don't know how to use their computer, and who, probably have bought their console second hand from someone who may even have originally been involved with this group.

continued on page 8

TECHO TIME

... John Paine

As Christmas is quickly approaching I'm afraid that other matters tend to distract me from playing with my beloved TI. So as a matter of course I will be be brief with this article on installing independant sound within the console.

This article was prompted by the fact that I was able to purchase a high resolution monitor for my son's system and get rid of the old black and white TV that he was using.

The first major flaw in using the monitor was the fact that there was no sound and this caused a little bit of heartache for my son. No more beep ....beep....honking noises from his bedroom and the shooting games were absolutely useless.

Two approaches to incorporating sound are viable.

1) Build an appropriate audio amplifier into the new monitor or include the amplifier in side the console.

2) Just put a speaker and isolating capacitor in the console and don't worry about the rather muted output from the speaker.

I chose to build an amplifier into the console and run it from the internal 5 volt supply. This allows plenty of sound and does not put a great load on the console power supply.

The heart of the amplifier is a National Semiconductor LM386 which is an 8 pin Audio Amplifier which is specified to run from a 4 to 12 volt power supply. With a small speaker mounted just behind the cartridge port and a potentiometer sticking up in the left hand corner just above the modulator connector this installation has been providing the necessary beeps and honks for a few weeks now.

The necessary parts are:

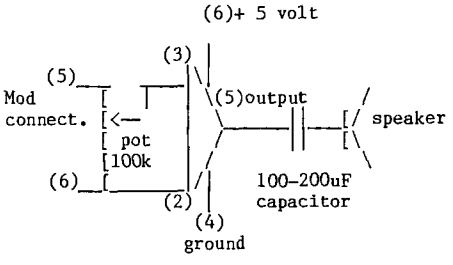
LM386 IC
100K ohm pot.
100 to 200 uF Electrolytic Capacitor
Small 8 to 16 ohm speaker

This minimum configuration will give a voltage gain of about 20 and only consume about 18mW of energy,

Audio signal from the console is picked up from pin 5 of the modulator socket and the 5 volts and ground can be picked up from the power board or motherboard (Consult your copy of the technical manual first). As stated above the pot is mounted immediately above the modulator connector and the speaker was mounted in the little speaker cage already provided by TI at the rear of the module port. Those of you who may have owned a TI 99/4 may remember that a speaker was provided. The current 99/4A's case still has the same enclosure (presumably a similar tool was used to injection mould all the cases for the family of computers).

Access to this compartment is via a pair of self tapping screws.

The circuit I used is indicated below.



pins 1,7,8 not connected

The final assembly was then taped to console mother board and tested. o

FAULT of the MONTH

Once again fault of the month is a misnomer. As our machines get older, more and more strange things are happening, and this month has been no exception.
One member recently bought over an original Stand Alone TI disk controller for repair. While he waited, a routine check showed that the power supply was shot and internally a 741s259 had evaporated. The diagnosis was fine but the repair was quite complicated. The power board in the disk controller had been replaced by a console power board some time ago and without too much thought a reconditioned power board was substituted.
The replacement board used was one of the newer switch mode types and I had to mount it in a different manner to the damaged board. Because the controller case was made of plastic, I just screwed the board down into the existing holes from the previous board. The chip problem was fixed and on power up I was promptly greeted with a dense cloud of smoke and a terrible stench of burnt capacitor.
On investigation, a dead short was placed on the board by the plastic case. Yes, conductive plastic was the culprit. Just shows that you must never assume anything around these machines. The black plastic case was manufactured using a metalised plastic formula, probably to help with RFI suppression.
Since then I have found that some of the black consoles use the same material.

Statistics compiled on console repairs still show that machines built after week 27, 1983 still represent the bulk of my workload. The common faults with these machines are corrupted roms and dead microprocessor chips. The most common faults in the black and silver consoles are the decoupling choke in the VDP oscillator and VDP ram chip failures. o

RAMCARD TIP

A problem recently cropped up with my HORIZON ramcard. As a result of a glitch in the system, whenever I first turned on the PEB, the ramcard light would come on and stay on. To make matters worse, the screen would blank and the keyboard became disabled.

It looked like a classic case of lock-up and would have meant taking out the RAMCARD, removing the batteries and letting the card discharge. Since I hadn't backed-up the ramcard onto a disk, this would have meant the total loss of all the files. No way!, I thought to myself and tried to come up with a less drastic solution.

I needed to disable the automatic powerup routine of the ramcard DSR, but I couldn't access either the ramcard or any of the diskdrives in order to load a diagnostic program. The only thing I could think of was to use EASYBUG in my MINIMEMORY module, and therein lies the solution....

To cut a long story short, here is what you need to do:

1) Turn everything off.
2) Turn on the console first with MINIMEMORY inserted and select option 2 (EASYBUG).
3) Turn on the PEB and TV.

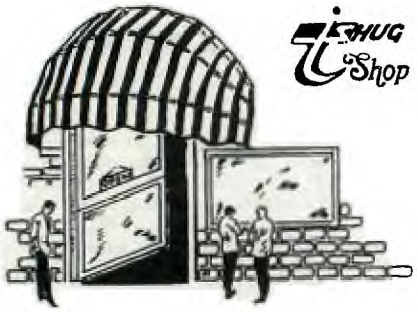
If everything worked out OK, the ramcard light won't be on and the keyboard will respond to input. If things didn't work out, go back and repeat steps 1) to 3) until they do.

4) Type C1000 and <enter>. (or C1200 or whatever)
5) Type 01 and <enter>.

At this point the ramdisk light should come on and stay on.

6) Escape from the C command by pressing the '.' then M4000 and <enter>.
7) Change the AA to a 00 and press <enter>.
8) Quit EASYBUG and return to the title screen.

You should now be able to load the CONFIG file and re-boot the ROS. In my case, I am using Version 6.3 of the CALL MENU ROS from the Miami TI Users Group. o



TIsHUG SHOP DECEMBER 1987

Christmas is once more upon us, and to help the members get the spirit of Christmas we will keep the price of blank disks at $12.00 per box till the end of December. Also we will be selling disk storage boxes (see listing below).

Also the price of multiple disk software will be reduced :- (ie single density software that won't fit on one disk)

2 disk set...$ 8.00
3 disk set...$10.00

MODULES:-

(a) VIATEL-1 (for serial port 1)
(b) VIATEL-2 (for serial port 2)
(c) CART WRITER. (similar to Console writer)
(d) DIAGNOSTICS (will test your TI peripherals)

The price of the above modules is $25.00 each
NOTE:-CART-WRITER will not work on the 1983 console.

RAM DISK CARD PARTS :-

Most of the remaining parts for this project were sold out at the November full day tutorial, the only item left is the battery set.

(a) Batteries AAA in stick form.....$ 14.00

THIRD RUN OF RAM CARD P.C.Bs.

If the response is large enought we will have a third run of the RAM card, this will require people putting up the $35.00 with their order. We will need at least 50 orders for boards to retain the same pricing as for the previous run.
At the time of writing I have only one order, so please get your order in quickly so we can get these under way.
If we only get orders for 25 P.C.Bs.the cost will be about $45.00 each, but if we can increase the order to 50 the price should be about $35.00 each.

PRINTER BUFFER PARTS :-

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

NOW THE STANDARD ITEMS:-

(a) New DS/DD 5 1/4" Disks (box).....$ 12.00
(b) Disk storage box (100 capacity)....$ 20.00
(c) " " " plus 10 disks.....$ 30.00
(d) " " " plus 20 disks.....$ 40.00
(e) T.I. Joystick handles.....$ .50
(f) 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.....$ 3.00
1986-June to Dec./1987-Jan.to Oct.

SOFTWARE:-

(a) Club Software Tapes.....$ 3.00
(b) Club Software Disks.....$ 5.00
(c) Picasso Publisher V2.0 (Arto Heino)$ 20.00

Bring or send your old original copy of Picasso to the shop for an updated copy.

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

FOR SALE

Stand alone PIO interface, with cable

$90

Contact Steven Carr
(02) 608 1968

PUBLICATIONS LIBRARY REPORT.
BY WARREN WELHAM.

Well I finally did it,after many tries.I finally wrote a report for the News Digest.

Firstly,I am happy to be able to report that the publications library is now fully catalogued. The library has 182 books and publications in it. 63 different overseas and local T.I. clubs newsletters(past and present).Ranging from America,Australia,Canada,England and other smaller countries.

A list is just in its final stges of completion after 11 months and should be ready this month(hopefully)of all the books in the library with their authors and catalogue numbers.

There is also a list of every issue of every oversea's and local newsletter the library has.I hope it will be avaiable at the club shop.Using this you can keep a record of what the library has,as I will list the new magazines we get each month.

I hope every one borrows big at the december meeting so you can read the publications the library has to offer over the christmas holidays and also because their will be no borrowing in February due to the library having a stocktake.

And finally If you are not yet a member of the Publications library,then I recommend you join as there is some excellent reading in the magazines as well as a host of entertaining games and useful routines for you to type in and enjoy over the christmas holidays.Remember it is only $5.00 to join the Publications library. o

I trust you found something to interest you in the software that was available last month.

During my recent holidays, I busied myself updating the complete software catalog. It is now available, in disk form only at this stage, in any of the following formats - double sided/double density (1 disk), double sided/single density (2 disks), single sided/double density (2 disks), single sided/single density (3 disks). In total, disk storage space, for the catalog, amounts to 970 sectors. All files on the catalog disk(s) are in D/V 80 format and contain the programs name, a brief description, its size in disk sectors and its language and peripheral requirements. In a change from past catalogs I have produced, this one shows the contents of each disk as it is in the library. The D/V 80 files on the catalog disk(s) each contain the contents of approximately 10 disks from the library. Be prepared to spend some time as you print it out as it runs to quite a number of pages.

New disks received in the past few weeks include:

DISK AID - a disk information, sector reading type utility from Col Christensen of Brisbane. The disk has 2 large D/V 80 text files which will give you all the information you need to get the most out of this package. Requires XB or E/A module, 32K, disk drive and printer, optional but recommended.

RAM DISK OP SYSTEM - Version 6.4 copy via John Paine and authored by members of the Miami User Group. For those with RAM cards this is the ultimate operating system.

CREATIVE FILING SYSTEM - Version 7.0 the latest update from Mark Beck. I understand that Version 8.0 may soon be available so I will hold of issueing this version until further word is heard.

FUNNELWEB - Version 4.0 the McGoverns classic software package.

3 DISKS OF GAMES, UTILITIES PICTURES from the Channel 99 user group in Canada. There is some really good material on these disks.

ASSEMBLY CONVERSIONS - a few more games and utilities from George Meldrum.

At the December meeting the following will be released:

ON DISK:

1. DISK AID as mentioned above.

2. FUNNELWEB V4.0 either on one double sided or double density disk or on 2 single sided disks. Make sure you get the right configuration for your system.

3. CHANNEL 99/1 containing the animal series of pictures, a maze race to the crown game, a RAM test utility plus source file, 3D Tic-Tac-Toe and an Airwolf fly helicopter game. All programs this disk require E/A module and 32K expansion.

4. CHRISTMAS MUSIC containing excellently programmed music with the words to sing along to. All your old favourite tunes will be on this disk.

ON TAPE:

TAPE 1987/12 - the same Christmas music as on the disk version.

TAPE 1987/12A - some games to play during the holiday period:

STAR PATROL - space game in XB
MAD DOGS - retrieve treasure and avoid dogs in XB
MOUSE - squish a mouse in XB
CONCENTRATION - concentration game in Basic
SEA POWER - well done battleship game in XB
NOMAD - a maze adventure in XB
BOOGENS 2 - shoot as many monsters as you can in XB
KILLER BEES - fight off the killer bees in XB

That's about all I will have time to do so hope you enjoy it.

On a final note, I have received one entry to the software competition. Make sure you work on yours over the Christmas period and remember, closing date for entries is at the February, 1988 meeting.

And almost forgot, new publications recently received and added to the publications library include October issues of Ottawa UG, Northern NJ UG, Bug Bytes, Edmonton UG, HV99er's, ATTIC and Spirit of 99. Some good articles in these issues so recommend you borrow if you are a member of the publications library. o

continued from page 16

```
2080 C=ASC(A$)! C=ascii valu
e of the first character of
the command

2090 IF C>96 THEN C=C-32 ! I
he ascii value is greater
an 90 (Z) decrease by 32
n and text for display

9020 SUB HELP

9030 DATA 8,5,"VALID COMMAND
S ARE:",10,5,"A = Add a Reco
rd",11,5,"D = Display a Reco
rd",12,5,"E = End Session"

9040 DATA 13,5,"H = Help (Th
is Menu)",14,5,"P = Print Da
ta",15,5,"Q = Query Data bas
e",16,5,"R = Remove a Record
"

9050 DATA 17,5,"U = Update a
Record",24,2,"Press SpaceBar
to Continue"

9060 RESTORE 9030 ! Ensures
data read starting at 9020

9070 FOR C=1 TO 10 :: READ A
,B,A$ :: DISPLAY AT(A,B):A$
:: NEXT C ! Reads the Data S
tatements and displays on sc
reen

9080 CALL KEY(0,K,S):: IF (K
<>32)+(S=-1)THEN 9080 ! Chec
ks for a spacebar press befo
re continuing

9090 SUBEND

9100 REM *** End of Help Sub
program ***

17000 SUB DISPLAY

17010 REM Take detail from t
he command line and get full
details of desired record

17020 REM Take the data stri
ng (record) appart and displ
ay the various parts on the
screen

17030 SUBEND

18000 SUB SEARCH

18010 REM Write a routine to
enter the file(already open)
and get record/s.

18020 REM Remember that it m
ust continue to search if th
e first character is not "~"

18030 REM Write routine to d
isplay Surname field and fir
st character of FName on scr
een with an index number
18040 SUBEND
```

o

THE COMMUNICATORS

Special Interest Group for Users
of the TEXPAC BBS.
by Ross Mudie, 9th November 1987.

CHRISTMAS GREETINGS.

I would like to take this opportunity to wish all members and their families a very safe and happy Christmas and New Year.

Don't forget the TIsHUG Software Competition whilst you have some spare time over the holiday period.

There is no TND magazine in January, however your committee is able to keep in touch with BBS members. The Group Co-ordinator, Secretary, TND Editor, Shop-keeper and Technical Co-ordinator are all capable of directly loading their own files into the NEWS menu. Watch out for these files which carry their user names. The first line of the file tells you when the file was uploaded into the BBS.

1. OPTIONAL SCREEN COLOURS FOR TEII AND 4A-TALK.

A number of members have asked for the BBS to periodically change their terminal screen colour to other than white characters on a blue background whilst others using non-TI99/4A computers on the BBS have had problems with the colour change string sending their terminal "off the deep end".

From version 3X of the BBS software users will be able to exercise some control over the screen colour when using TEII or to avoid the screen change character string.

To do this users will have to change their password making the last character a number. Passwords with a letter as the last character will remain with the current white text on a blue screen.

The following is the last character allocation.

Digit	Foreground	Background	Note.
0	no change	no change	LF/CR
1	White	Black	
2	White	Green	
3	White	Magenta	
4	Black	Cyan	
5	Black	Yellow	
6	Black	White	
7	Cyan	Blue	
8	Default		
9	Default		
Default			
Non Digit	White	Blue	

Digit 0 (zero) gives a LF/CR in place of the colour string.

Digits 1-9 and the default include a clear screen byte (CHR$(12)).

At the time that this item was written I requested that users advise me of their alternative preferences to those shown above. These preferences will be included as options 8-9. The available choices are listed on screen when a user is about to perform a password change.

Once you have changed your password the new colours will be given.

2. NEW REMOTE SYSOP FACILITY.

The SYSOP is now able to remotely delete or upload basic or extended basic programs which appear in the PROGRAMS menu. I am investigating the feasability of allowing all users to upload programs, possibly into the Discussion Rooms area. There will be more on this subject when something is worked out.

3. DOWNLOADING EXTENDED BASIC PROGRAMS WITH BASIC.

When using the TEII module it is much more convenient to use console basic to download prorams than x/b which for most users requires the changing of the command module.

Extended basic programs including those containing assembly can be downloaded using TI BASIC. Once the program has been downloaded it should be saved to disk or cassette tape using SAVE DSKx.PROGNAME or SAVE CS1 . Attempts to LIST or RUN will be fruitless but OLD RS232 and SAVE DSKx.PROGNAME or SAVE CS1 are valid in this case.

When saving multipart Extended Basic programs which contain assembly programs on disk, it is advisable to use the same name as the program carries on the BBS. These programs usually load the subsequent parts of the program from a name contained in the previous part, so if a different name is used then an error will occur as an attempt is made to load the incorrectly named part.

4. HELP FILES ON BBS.

I have observed that some people are unaware of some of the common control facilities for the BBS. The most common problems seem to be lack of understanding of CONTROL S which pauses, CONTROL Q to un-pause, the letter E once only to escape from a file which is being listed and CONTROL H which permits a back space. There are two files in the NEWS menu on the BBS which detail how to use the BBS. These are BBS_HELP and SENDMAIL. If in doubt about any facility on the BBS then users may leave their questions as mail to SYSOP on the BBS.

5. INTERMITTENT PROBLEMS AND LOCKUPS WITH TEII.

The major problem with TEII, extended basic or, in fact any other command cartridge, is contact problems on the cartridge edge connector. These contacts and the mating contacts in the console can be cleaned with many layers of clean, lint free cotton over the blade of a small screwdriver. Take extreme care not to scratch the contacts or the printed circuit tracks and always observe static electric precautions to prevent damage to static sensitive components. If an electronic grade FREON cleaner is available it may be used to further clean the edge connector and socket. Never use anything abrasive on the edge connector or the contacts as this will lead to future problems.

6. BBS ACCESS.

Access to the BBS is permitted for registered users only due to problems caused in the past by unregistered people. Requests for access should be sent to the Secretary TIsHUG, PO Box 214 Redfern, 2016. Subscription rates are shown on page 1 of the TND magazine. o

TEXPAC BBS, An Overview of the Facilities.
by Ross Mudie, 26th October 1987.

It is now a little over a year since I took on the task of SYSOP for the TEXPAC BBS. Regular readers of the TND and users of the BBS will have observed the steady effort to improve the BBS in both reliability & facilities offered. This BBS uses an almost standard TI99/4A home computer to provide a reasonably versatile Bulletin Board Service for the registered users.

When I took the task on the BBS was very unreliable. The previous SYSOP needed to provide constant supervision since there were many shortcomings in the program. My aim was to improve the BBS so that it could provide a reasonably reliable service, without the BBS ruling the life of the SYSOP. The BBS now operates unattended, sometimes for periods of over a month at a time, with with remote SYSOP supervision only being provided. The programming task that I have undertaken on this BBS has taught me a lot about assembly programming in the TI99/4A and there is still more to learn as I continue.

The basic structure of the BBS has been maintained so that it is easy to operate and to provide support for users with the simplest of TI99/4A configurations, (i.e., TEII, Cassette recorder, stand alone RS232 and modem). Many operational short cuts have been provided for experienced users and new features have been added. The major facility which has been removed is "the SYSOP keyboard chat" & "Page SYSOP". These features were virtually redundant due to the unattended nature of the operation and this made additional memory space free for other enhancements.

The following information summarises the major features of the TEXPAC BBS.

1. NORMAL USER FACILITIES.

a) News Menu - Information Files, Adds, Club News, BBS and program information.

b) Programs Menu - Downloadable basic and ext'd basic programs in memory image format. Some extended basic programs contain imbedded assembly programs.

c) Mail System - Mail can be sent to other registered users. All Mail is date & time stamped. Mail is presented at log on & users may save or delete mail.

d) Uncleared mail to ALL may be read prior to the SYSOP placing it in the appropiate file.

e) Password Change - Users can change their own password. If the last character of the password is a numeric digit then the colours of a terminal screen used with TEII may be selected from the option list.

f) User Log - Users may view the activity log which the BBS maintains.

g) Discussion Rooms - This facility is still being developed. Operational facilities so far include a mail room for the committee & committee file load and file delete. The committee files are loaded into the NEWS menu without SYSOP intervention.

h) Log On Again - This allows users to log on again to re-access the mail reading routine without the need to call the BBS again.

i) General Control - Files may be paused or escaped from. Abbreviated file or function selection is provided whilst full menus are easily obtained. The BBS times out after 9 minutes of non-activity and releases automatically in the event of the data carrier being lost by the modem.

2. REMOTE SYSOP FACILITIES.

These facilities allow the SYSOP to maintain all of the software based activities of the BBS without having to actually go to the BBS site.

a) Catalogueing of all disks.

b) Enabling new User Names and Passwords. (Also available to Secretary and Co-ordinator).

c) Send mail without the date/time stamp.

d) Read/Set Clock/calender in BBS.

e) Load/Delete News files.

f) Load/Delete Basic and Extended basic programs.

g) Place unfinancial users in a category which gives a special message to these users at log on.

3. ON SITE LOGGING.

All BBS operations such as files read, programs downloaded, log-ons and log offs are recorded with time on a dedicated 80 column printer. This assists in the diagnosis of problems.

4. BBS HARDWARE.

The BBS consists of the following hardware:

TI99/4A console, amber monitor and PE Box which contains:

a) 128K card (32K only used at present).

b) RS232 card (special EPROM).

c) Triple Tech Card (clock/calender & printer buffer).

d) Disk controller & 3 DSDD disk drives used as DSSD.

Additional equipment includes Auto Answer 300 baud modem, hardware watchdog timer and a MX80 printer.

5. SOFTWARE and OPERATING SYSTEM.

The main program of the BBS runs in TI console basic with links to assembly language provided by the Editor/Assembler cartridge.

The basic program is just over 10K bytes in length & with the necessary variables and arrays there is very little free space in the console RAM. The assembly program is a little over 6K in the high expansion RAM and with a buffer allocation of 14K, there is about 4K of free space for more assembly programming. The majority of text strings transmitted by the BBS reside in the low RAM and there is about 4K free here also. o

*

from page 3

May I suggest an URGENT review of this situation, and plan to Spead the word in in News Media, Computer Magazines, and other forms of promotion. We have the financial capabilities to promote this group in a big way, but we seem to be lacking in the enthusiasm we once had, in its promotion.

If this group is to survive another 6 to 10 years, then we need to rethink out how we are to tell others we exist.

I would like to believe that our past eforts have not been in vein, that our club and our computer will not die.

Kindest Regards
SHANE ANDERSEN

o

LINKING EXT'D BASIC-ASSEMBLY

WITH ROSS MUDIE.

TRANSFERING MEMORY IMAGE PROGRAMS
FROM TAPE TO DISK and DISK TO TAPE.

by Ross Mudie of TIsHUG, 12th November 1987.

This program is made up of a routine from an ILLAWARRA Regional Group disk and routines from the Smart Programmer to provide GPLLNK and DSRLNK which were not provided by TI in the INIT routines of extended basic. The source file is heavily documented to assist with your understanding of the operation of the program.

The program allows users to transfer memory image programs from tape to disk or disk to tape. The purpose of this article is to give programmers an application example of DSR linking to both the Cassette Tape system and the Disk system. Access to the cassette routines is by using GPLLNK whilst access to the disk system is via DSRLNK. The source file is in three distinct parts

(a) The Tape/Disk, Disk/Tape program.
(b) The GPLLNK routine.
(c) The DSRLNK routine.

These individual parts can be used in other programs.

This source file will appear on the TEXPAC BBS during December 1987 as will a downloadable program version of the extended basic program containing the object file in CALL LOAD format. This when downloaded will allow immediate use of the routines from extended basic.

The extended basic program which links into the assembly follows. This version of the extended basic program uses the conventional CALL LOAD from disk for the loading of the assembled Display Fixed 80 object file.

```
100 ! SAVE DSK1.TAPE/DISK
110 CALL INIT
120 CALL LOAD("DSK1.DTO")
125 CALL CLEAR
126 CALL KEY(3,K,S)
130 INPUT "DISKFILE TO SAVE/LOAD       eg. DSKx.PROGNAM
E              ":NAME$
140 IF LEN(NAME$)>15 THEN 130
190 PRINT :"PRESS  D. DISK TO TAPE":"OR     T. TAPE TO
DISK"
200 CALL KEY(0,K,S)
210 IF S=0 THEN 200
220 IF K=68 THEN 260
230 IF K<>84 THEN 200
240 CALL LINK("TAPDIS",NAME$)
250 GOTO 270
260 CALL LINK("DISTAP",NAME$)
270 PRINT :"DO ANOTHER? Y/N": : :
280 CALL KEY(0,K,S)
290 IF S=0 THEN 280
300 IF K=89 THEN 130
310 IF K<>78 THEN 280
320 STOP
```

```
*  LINK-IT 14     Ross Mudie.      Source=DTS  Object=DTO

       DEF  TAPDIS,DISTAP

STATUS EQU  >837C                          Status Byte
FAC    EQU  >834A              Floating Point Accumulator
PAB    EQU  >0F80     Address of Peripheral Access Block
PNTR   EQU  >8356         Name pointer to PAB for DSRLNK
VMBW   EQU  >2024               Video Multiple Byte Write
STRREF EQU  >2014      Gets a string from Extended Basic
VMBR   EQU  >202C               Video Multiple Byte Read

WS     BSS  32                          Register workspace
PABDSK DATA >0500,>1000,0,>2000     Disk PAB in CPU RAM
       BYTE 0                 Screen Display Offset byte
NLENBY BYTE 0                          Name length byte
       BSS  15       Space allocation for DSKx.PROGNAME
B15    BYTE 15        Byte size constant of 15, decimal
       EVEN

PABCS  DATA >0600,>1000,0,>2000,>6003      Cassette PAB
CS1    TEXT 'CS1'                         Cassette name
SAVE   BYTE >06  Read about this on page 297 of E/A man
LOAD   BYTE >05  Read about this on page 296 of E/A man
       EVEN
SAVRTN DATA 0       2 byte place to save return address

*  DISK ACCESS SUBROUTINE
DISK   LI   0,PAB          Address in VDP RAM to put PAB
       LI   1,PABDSK     Info on the PAB in the CPU RAM
       LI   2,25     How many bytes to write in VDP RAM
       BLWP @VMBW                              Do it
       LI   6,PAB+9      DSRLNK uses the VDP address of
       MOV  6,@PNTR        the name length byte in the
       BLWP @DSRLNK    PoiNTeR at >8356 to find the PAB
       DATA 8            The BLWP does the disk access
       RT                      Return after subprogram

*  SUBROUTINE TO TRANSFER LENGTH  OF  JUST READ PROGRAM
*  INTO SAVE PAB.
CHANGE LI   0,>1002 Location of length of file just read
       LI   2,2             2 bytes to be read from VDP
       BLWP @VMBR                              Get it
       RT                                      Return

*  TAPE ACCESS SUBROUTINE
*  See page 253 of e/a manual to assist with this part.

TAPE   LI   0,PAB          Address in VDP RAM to put PAB
       LI   1,PABCS       Source of PAB info in CPU RAM
       LI   2,13               Number of bytes to write
       BLWP @VMBW                              Do it
       LI   1,PAB+13   Addr of byte after CS PAB in VDP
       MOV  1,@PNTR                    Put it in PNTR
       LI   1,>0800  The byte >08 required for DSR call
       MOVB 1,@>836D             Put the >08 in >836D
       LI   0,PAB+10       Address of device name   CS1
       LI   1,FAC        Device name must be put in FAC
       LI   2,3                Number of bytes to read
       MOV  2,@PNTR-2     Put length of name in >8354-5
       BLWP @VMBR          Transfer copy of name to FAC
       CLR  @>83D0                        Must be zero
      MOVB @>83D0,@STATUS Clr status byte before GPLLNK
       BLWP @GPLLNK           Branch Load WSP to GPLLNK
       DATA >3D               & do the Tape DSR access.
       RT                                      Return

*  GET DISK NAME FROM EXTENDED BASIC SUBROUTINE
GETDNM CLR  R0                              Element 0
       LI   R1,1                           Argument 1
       MOVB @B15,@NLENBY                Maximum length
       LI   R2,NLENBY           Where to put DSKx.NAME
       BLWP @STRREF                            Do it
       RT                      Return from subprogram

*  ENTRY TO DISK to TAPE ROUTINE
DISTAP MOV  11,@SAVRTN          Remember return address
       LWPI WS          Load our own register work space
       BL   @GETDNM              Get disk name from x/b
       MOVB @LOAD,@PABDSK      Set up disk PAB for load
       MOVB @SAVE,@PABCS       Set up tape PAB for save
       BL   @DISK                    Do the disk access
       LI   1,PABCS+6    Load R1 with addr of size word
       BL   @CHANGE    To put loaded size in resave PAB
       BL   @TAPE                    Do the tape access
       JMP  RETURN
```

continued on page 10

from page 9

```
*  ENTRY TO TAPE to DISK ROUTINE
TAPDIS MOV  11,@SAVRTN          Remember return address
       LWPI WS                  Load register work space
       BL   @GETDNM               Get disk name from x/b
       MOVB @LOAD,@PABCS            Set up tape for load
       MOVB @SAVE,@PABDSK           Set up disk for save
       BL   @TAPE                        Load from Tape
       LI   1,PABDSK+6          Address for the file size
       BL   @CHANGE        Take file size from tape-disk
       BL   @DISK                   Save program to disk

RETURN CLR  0                   Prevent false indication
       MOVB 0,@STATUS                          of errors
       LWPI GPLWS                 Reload GPLWS for X/B
       MOV  @SAVRTN,11          Remember where to return
       RT                                   Return to x/b
* ---------------------------------------------------------
*  GPLLNK routine for extended basic.

UTLWS  EQU  >2038                        Utility workspace
SUBST  EQU  >8373                Subroutine Stack Pointer
GRMRA  EQU  >9802                        GRoM Read Address
GPLWS  EQU  >83E0                          GPL Work Space

GPLLNK DATA UTLWS     BLWP @ GPLLNK, vectors: Workspace
       DATA GPLLN1                              Entry point

GPLLN1 MOVB @GRMRA,R0   {  Save Grom read address in R0
       SWPB R0          {
       MOVB @GRMRA,R0   {
       SWPB R0          {

       AI   R0,-3         Back up to the XML instruction

       MOVB @SUBST,R1   {          Get the stack pointer
       SRL  R1,8        {
       AI   R1,>8300    {

       INCT R1          {   Push XML address for return
       MOV  R0,*R1      {
       SWPB R1          {
       MOVB R1,@SUBST   {

       LI   R3,>2000   Load R3 with address of XML link
       MOV  *R3,R2       Save current XML link address

       LI   R0,GPLLN2   {        Load new XML address at
       MOV  R0,*R3      {                  address >2000

       MOV  *R14+,@>83EC        Place GPLLNK data value
*                 at >83CE & advance saved PC past DATA
       LWPI GPLWS                  Load GPL/xb workspace
       B    @>0060              Go to routine in console
*                                          GROM chip 0

GPLLN2 LWPI UTLWS      Should return here, reload UTLWS
       MOV  R2,*R3 Reload original XML address in >2000
       RTWP        Return to caller after GPLLNK & DATA
*  DSRLNK routine for XB

VSBR   EQU  >2028
SCLEN  EQU  >8355
CRULST EQU  >83D0
SADDR  EQU  >83D2
SAVCRU DATA 0                  CRU address of peripheral
SAVENT DATA 0                       Entry address of DSR
SAVLEN DATA 0                 Save device name length
SAVPAB DATA 0               Pointer to device name in PAB
SAVVER DATA 0                       Version number of DSR
DLNKWS DATA 0,0,0,0,0,0
TYPE   DATA 0,0,0,0,0,0,0,0,0,0,0,0
NAMBUF DATA 0,0,0,0
H20    DATA >2000
DECMAL TEXT '.'
HAA    BYTE >AA
       EVEN
*              DSRLNK - Workspace, PC for BLWP @DSRLNK

DSRLNK DATA DLNKWS,DLENTR
DLENTR MOV  *R14+,R5        Fetch program type for link
       SZCB @H20,R15                      Reset equal bit
       MOV  @PNTR,R0            Fetch pointer into PAB
       MOV  R0,R9                           Save pointer
       AI   R9,-8           Adjust pointer to flag byte
       BLWP @VSBR             Read device name length
       MOVB R1,R3                              Store it
       SRL  R3,8                 Make it a word value
       SETO R4                     Initialise counter
       LI   R2,NAMBUF                 Point to NAMBUF
LNK$LP INC  R0             Point to next char of name
       INC  R4               Increment char counter
       C    R4,R3                        End of name?
       JEQ  LNK$LN                               Yes
       BLWP @VSBR                  Read current char
       MOVB R1,*R2+                Move it to NAMBUF
       CB   R1,@DECMAL          Is it a decimal point?
       JNE  LNK$LP                                No
LNK$LN MOV  R4,R4                 Is name length zero?
       JEQ  LNKERR                          Yes - Error
       CI   R4,7                  Is name length > 7 ?
       JGT  LNKERR                          Yes - Error
       CLR  @CRULST
       MOV  R4,@SCLEN-1    Store name length for search
       MOV  R4,@SAVLEN        Save device name length
       INC  R4                              Adjust it
       A    R4,@PNTR       Point to position after name
       MOV  @PNTR,@SAVPAB              Save pointer into
*                                          device name
* SEARCH ROM FOR DSR
SROM   LWPI GPLWS          Use GPL Workspace to search
       CLR  R1                    Version found of DSR
       LI   R12,>0F00                Start over again
NOROM  MOV  R12,R12           Anything to turn off ?
       JEQ  NOOFF                                  No
       SBZ  0                        Yes, turn it off!
NOOFF  AI   R12,>0100          Next DSR ROM's turn on
       CLR  @CRULST            Clear in case finished
       CI   R12,>2000                      At the end?
       JEQ  NODSR           No more ROM's to turn on
       MOV  R12,@CRULST     Save address of next CRU
       SBO  0                            Turn on ROM
       LI   R2,>4000          Start at the beginning
       CB   *R2,@HAA                Is it a valid ROM
       JNE  NOROM                                  No!
       A    @TYPE,R2          Go to the first pointer
       JMP  SGO2
SGO    MOV  @SADDR,R2        Continue where we left off
       SBO  0                          Turn ROM back on
SGO2   MOV  *R2,R2                  Is address a zero?
       JEQ  NOROM           Yes, no program to look at
       MOV  R2,@SADDR        Remember where to go next
       INCT R2                        Go to entry point
       MOV  *R2+,R9                   Get entry address
* CHECK AND SEE IF NAME MATCHES
       MOVB @SCLEN,R5             Get length as counter
       JEQ  NAME2           Zero length? Don't do match
       CB   R5,*R2+                 Does length match ?
       JNE  SGO                                    No
       SRL  R5,8                  Move to right place
       LI   R6,NAMBUF                 Point to NAMBUF
NAME1  CB   *R6+,*R2+           Is character correct?
       JNE  SGO                                    No
       DEC  R5                        More to look at?
       JNE  NAME1                                 Yes
NAME2  INC  R1                      Next version found
       MOV  R1,@SAVVER              Save version number
       MOV  R9,@SAVENT               Save entry address
       MOV  R12,@SAVCRU                Save CRU address
       BL   *R9                          EXECUTE ROUTINE
       JMP  SGO                      Not right version
       SBZ  0                              Turn off ROM
       LWPI DLNKWS                Select DSRLNK workspace
       MOV  R9,R0                   Point to flag in PAB
       BLWP @VSBR                         Read flag byte
       SRL  R1,13               Just want the error flags
       JNE  IOERR                                 Error !
       RTWP
* ERROR HANDLING
NODSR  LWPI DLNKWS                 Select DSRLNK workspace
LNKERR CLR  R1                       Clear the error flags
IOERR  SWPB R1
       MOVB R1,*R13      Store error flags in calling R0
       SOCB @H20,R15           Indicate an error occured
       RTWP                              Return to caller
       END
```

Hardware News

By P. Schubert

The new MULTI-FUNCTION CARD was displayed and used to run a system at the November Tutorial day. I now have two prototype MULTI-CARD's, both of which have functioned reliably,and one has been used in different PE boxes. This MULTI-CARD was designed for the following reasons:-

1) To provide an advanced double density controller more suited to todays modern disk drives and software, developed to run at faster speeds, and including EXTRA CALLS provided by Corcomp and Myarc designs as well as our own enhancements.

2) To provide a source of local RS232 CARDS which have become scarce (many have become faulty and are uneconomical to repare due to use of special IC's), and also to enhance it to meet todays extra needs. Additional commands include VIATEL (PRESTEL) 1200/75 baud, 75/1200, MIDI (31250 baud), RTTY 50 baud,and extra speeds of 50 baud and 19200 baud. The 'PIO' port can also be called 'PRINT'.

3) Memory expansion of 32K eliminates need for separate 32K card in PE box.

4) Only one power supply is required for all functions to work, although the NEG supply is used for RS232. This allows the MULTI-CARD to be used in home-made PE box systems, including the Poormans Disk System featured in earlier TND.

5) It is a buffered card for extra reliability and compatability with other commercial PE boards. Also the Ports use TI standard Connectors.

6) All these functions can be provided on one card to save space in your PE box, and keep cost to a minimum.

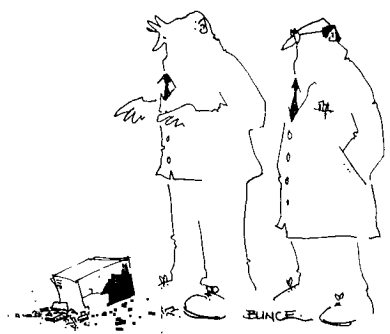
This card represents 2 years of part-time design and development work on my part, and it has resulted in a number of other peripherals being developed for our little orphan TI in the process, notably the MINI-PE SYSTEM, which is a stand-alone version of this card, and the "single" chip 32K design now used as a kit for mounting within console.

I am in the process of ordering PCBs now so that production can start early next year, and I would like to hear from anyone interested in either the full MULTI- CARD or a partial RS232 or Disk only card. Cost is expected to be around $230 for Disk Control only, $170 for RS232 only, and a maximum of $390 for all options including 32K.

A MULTI-CARD is also available for loan to Regional Groups. Contact me on (02)3585602 or write to

P. Schubert
P.O.Box 28
Kings Cross 2011

o



'Congratulations, Brian. You've just invented the world's first 2000 bit micro'

JOYSTICK CONVERTOR.
by Laurie Marsh of TIsHUG.

Just recently my 2 boys had worn out their third joystick, which I can't complain about considering the 2 years of service we got out of them, averaging about an hour a day.

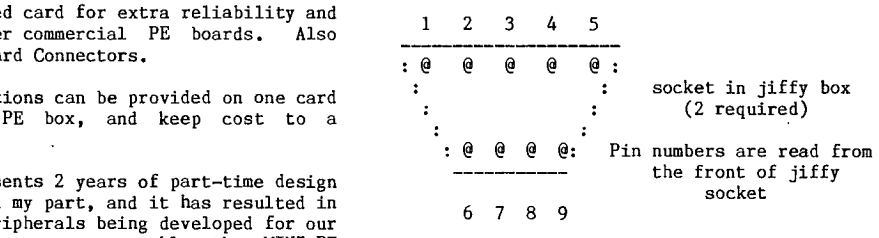
It would be great if we could buy a stick to go straight into the TI port with out modifications. Thus if you bought cheaply as I did ("Quick shot 1, $12.99 at TARGET") then, if you received a faulty unit, you could at least return it to the supplier and get it replaced as I had to !!!

This brings us to this very simple project, with not much more work or cost then converting your new joystick.

I used a small plastic jiffy box with 2 male 9 pin sockets mounted to allow both sticks to be plugged in. Then using the original TI "Y" cable with plug left attached and the old joysticks cut from the cables. The 6 inner wires are then terminated to the back of the 9 pin sockets via small signal type diodes ("1N4001, 1N914 etc").

Once you have mounted the sockets in the case, wire the 10 diodes with the black band end ("cathode") soldered to the back of the socket (except the white "live" which should be soldered directly without a diode).

The 5 wires from each cable can then be soldered to the free end of the diodes, but cover with spaghetti first so no cross shorts can develop.



TI "Y" cable colours to sockets, with pin numbers to suit "QUICKSHOT1"
(one cable to one socket)

Pin 1.....orange up
Pin 2.....Green down
Pin 3.....brown left
Pin 4.....Blue right
Pin 6.....black fire
Pin 8.....white live (no diode)

PARTS REQUIRED

1 * SMALL plastic jiffy box
10 * IN4001 OR EQUIV
YOUR OLD TI JOYSTICK "Y"CABLE
2 * 9 PIN MALE SOCKETS
Length of spaghetti to suit diode size
And of course 2 JOYSTICKS SUITABLE FOR ATARI VIDEO GAME SYSTEM, ATARI 400/800, COMMODORE VIC-20-C64, NEC PC 6001.

o

'It used to be notes up the chimney asking for train sets — now it's all printouts requesting expansion interfaces and floppy disk drives.'

```
100 REM CHANNEL 99 HAMILTON
USER'S GROUP
110 REM THE BOOGENS
120 REM A1-BOOO3-JGU
130 REM ****************
140 REM * W.M. JOHNSON *
150 REM ****************
160 CALL CHAR(128,"147687612
6")
170 CALL COLOR(12,7,7)
180 CALL CLEAR
190 FOR L=5 TO 8
200 CALL COLOR(L,7,7)
210 NEXT L
220 FOR I=24 TO 1 STEP -1
230 CALL HCHAR(I,1,120,32)
240 CALL SOUND(-200,350-10*I
,5,220-10*1,5)
250 NEXT I
260 CALL SCREEN(7)
270 PRINT TAB(9);"THE BOOGEN
S":TAB(13);"ARE":TAB(11);"CO
MING": : : : : : : :
280 FOR L=5 TO 8
290 CALL COLOR(L,2,1)
300 RANDOMIZE
310 NEXT L
320 CALL SOUND(750,333,6,444
,6,555,6)
330 CALL SOUND(10,110,0,120,
0)
340 CALL CLEAR
350 CALL CHAR(40,"0808081818
")
360 CALL CHAR(41,"0002041818
")
370 CALL CHAR(42,"000000181F
")
380 CALL CHAR(43,"0000001818
0402")
390 CALL CHAR(44,"0000001818
10101")
400 CALL CHAR(45,"0000001818
204")
410 CALL CHAR(46,"000000F818
")
420 CALL CHAR(47,"0040201818
")
430 CALL CHAR(59,"081C7F5D49
14")
440 CALL CHAR(58,"080849495D
3E1C08")
450 CALL CHAR(64,"4F7F0F1F0A
0203")
460 CALL CHAR(65,"40224488F0
F2F5F8")
470 CALL CHAR(74,"001C042E1F
2E041C")
480 CALL CHAR(73,"001C2070FF
70201C")
490 CALL CHAR(80,"0003020A1F
0F7F4F")
500 CALL CHAR(81,"F8F5F2F088
44224")
510 CALL CHAR(88,"00002892BA
FE381")
520 CALL CHAR(89,"10387CBA92
92101")
530 CALL CHAR(96,"00C04050F8
F0FEF2")
540 CALL CHAR(97,"1FAF4F0F11
22442")
550 CALL CHAR(104,"382074F87
42038")
560 CALL CHAR(105,"38040EFF0
E0438")
570 CALL CHAR(112,"F2FEF0F85
040C")
580 CALL CHAR(113,"024422110
F4FAF1F")
590 SC=0
600 HARD=5
610 BOOG1=1
620 BOOG2=1
630 BOOG3=32
640 BOOG4=24
650 BOOG5=24
660 BOOG6=24
670 BOOG7=1
680 BOOG8=0
690 CALL COLOR(12,8,8)
700 CALL SCREEN(15)
710 REM **************
720 REM * MOVE LOOP *
730 REM **************
740 FOR MOVE=1 TO HARD
750 CALL JOYST(1,ROUND,Y)
760 POINT=POINT+ROUND/4
770 IF POINT>8 THEN 790
780 IF POINT<1 THEN 810 ELSE
 820
790 POINT=1
800 GOTO 820
810 POINT=8
820 CALL HCHAR(12,16,39+POIN
T)
830 CALL KEY(1,FIRE,ST)
840 IF FIRE=18 THEN 1950
850 NEXT MOVE
860 REM ****************
870 REM * PICK A BOOGEN *
880 REM ****************
890 CALL COLOR(BOOGEN+3,1,1)
900 BOOGEN=INT(RND*8)+1
910 DEAD=0
920 ON BOOGEN GOTO 960,1080,
1200,1320,1440,1560,1680,180
0
930 REM ****************
940 REM * BOOGEN MOVER *
950 REM ****************
960 IF DEAD=1 THEN 1040
970 BOOG1=BOOG1+1
980 IF BOOG1=10 THEN 2050
990 CALL HCHAR(BOOG1+1,16,58
)
1000 CALL HCHAR(BOOG1+2,16,5
9)
1010 CALL COLOR(BOOGEN+3,7,1
)
1020 CALL HCHAR(BOOG1,16,32)
1030 GOTO 2030
1040 CALL HCHAR(BOOG1+2,16,3
2)
1050 CALL HCHAR(BOOG1+1,16,1
28)
1060 BOOG1=1
1070 GOTO 900
1080 IF DEAD=1 THEN 1160
1090 BOOG2=BOOG2+1
1100 IF BOOG2=10 THEN 2050
1110 CALL HCHAR(BOOG2,29-BOO
G2,32)
1120 CALL HCHAR(BOOG2+1,28-B
OOG2,65)
1130 CALL HCHAR(BOOG2+2,27-B
OOG2,64)
1140 CALL COLOR(BOOGEN+3,7,1
)
1150 GOTO 2030
1160 CALL HCHAR(BOOG2+1,28-B
OOG2,32)
1170 CALL HCHAR(BOOG2+2,27-B
OOG2,128)
1180 BOOG2=1
1190 GOTO 900
1200 IF DEAD=1 THEN 1280
1210 BOOG3=BOOG3-1
1220 IF BOOG3=19 THEN 2050
1230 CALL HCHAR(12,BOOG3,32)
1240 CALL HCHAR(12,BOOG3-1,7
3)
1250 CALL HCHAR(12,BOOG3-2,7
4)
1260 CALL COLOR(BOOGEN+3,7,1
)
1270 GOTO 2030
1280 CALL HCHAR(12,BOOG3-1,3
2)
1290 CALL HCHAR(12,BOOG3-2,1
28)
1300 BOOG3=32
1310 GOTO 900
1320 IF DEAD=1 THEN 1400
1330 BOOG4=BOOG4-1
1340 IF BOOG4=15 THEN 2050
1350 CALL HCHAR(BOOG4,BOOG4+
4,32)
1360 CALL HCHAR(BOOG4-1,BOOG
4+3,81)
1370 CALL HCHAR(BOOG4-2,BOOG
4+2,80)
1380 CALL COLOR(BOOGEN+3,7,1
)
1390 GOTO 2030
1400 CALL HCHAR(BOOG4-1,BOOG
4+3,32)
1410 CALL HCHAR(BOOG4-2,BOOG
4+2,128)
1420 BOOG4=24
1430 GOTO 900
1440 IF DEAD=1 THEN 1520
1450 BOOG5=BOOG5-1
1460 IF BOOG5=14 THEN 2050
1470 CALL HCHAR(BOOG5-1,16,8
9)
1480 CALL HCHAR(BOOG5-2,16,8
8)
1490 CALL COLOR(BOOGEN+3,7,1
)
1500 CALL HCHAR(BOOG5,16,32)
1510 GOTO 2030
1520 CALL HCHAR(BOOG5-1,16,3
2)
1530 CALL HCHAR(BOOG5-2,16,1
28)
1540 BOOG5=24
1550 GOTO 900
1560 IF DEAD=1 THEN 1640
1570 BOOG6=BOOG6-1
1580 IF BOOG6=15 THEN 2050
1590 CALL HCHAR(BOOG6,28-BOO
G6,32)
1600 CALL HCHAR(BOOG6-1,29-B
OOG6,97)
1610 CALL HCHAR(BOOG6-2,30-B
OOG6,96)
1620 CALL COLOR(BOOGEN+3,7,1
)
1630 GOTO 2030
1640 CALL HCHAR(BOOG6-1,29-B
OOG6,32)
1650 CALL HCHAR(BOOG6-2,30-B
OOG6,128)
1660 BOOG6=24
1670 GOTO 900
1680 IF DEAD=1 THEN 1760
1690 BOOG7=BOOG7+1
1700 IF BOOG7=14 THEN 2050
1710 CALL HCHAR(12,BOOG7,32)
1720 CALL HCHAR(12,BOOG7+1,1
05)
1730 CALL HCHAR(12,BOOG7+2,1
04)
1740 CALL COLOR(BOOGEN+3,7,1
)
1750 GOTO 2030
1760 CALL HCHAR(12,BOOG7+1,3
2)
1770 CALL HCHAR(12,BOOG7+2,1
28)
1780 BOOG7=1
1790 GOTO 900
1800 IF DEAD=1 THEN 1880
1810 BOOG8=BOOG8+1
1820 IF BOOG8=10 THEN 2050
```

continued on page 15

```
100 CALL COLOR(14,9,9)
110 CALL CHAR(136,"FFFFFFFFF
FFFFFFF")
120 CALL MAGNIFY(4):: CALL S
CREEN(2)
130 NIV=1 :: CHA=5 :: DENT=1
 :: C=8 :: SCORE=0 :: SPEED=
12 :: CS=14
140 CALL CLEAR
150 FOR I=2 TO 8 :: CALL COL
OR(I,16,1):: NEXT I
160 DISPLAY AT(10,12):"1983"
 :: DISPLAY AT(12,5):"J A W
B R E A K E R"
170 DISPLAY AT(14,5):" PAR
 P.PELLETIER"
180 FOR I=1 TO 400 :: NEXT I
190 DISPLAY AT(10,7):" " ::
DISPLAY AT(12,5):" "
200 DISPLAY AT(14,5):" "
210 CALL CHAR(104,"FFFFFFFFF
FFF000000000000000000000FFFFF
FFFFFFF000000000000000000000"
)
220 CALL CHAR(112,"070F1F333
77F7EFFFFE773393C0E0F03E0F0F
8CCECFE7EFFFFE7CE9C3C70F0C0"
)
230 CALL CHAR(116,"3F7FFFFFF
F000000000000FFFFFF7F3FF0F8F
CFCFC000000000000FCFCFCF8F0"
)
240 UP$="000000DBDBDB0000000
0DBDBDB0000000000006C6C6C000
000006C6C6C000000"
250 CALL CHAR(120,UP$)
260 OUT$="C0C0D8181B030300C0
C0D8181B0303000C0C6C60600000
000C0C6C6C6060000000"
270 CALL CHAR(128,OUT$)
280 DN$="00000000DBDBDBDB00D
BDBDBDB0000000000000006C6C6C6
C006C6C6C00000000"
290 CALL CHAR(124,DN$)
300 CALL CHAR(98,"0000000000
000010")
310 CALL CHAR(97,"0000000000
3C3C3C")
320 CALL CHAR(96,"0000FFFFFF
FF0000"):: CALL COLOR(9,C,1)
330 GOSUB 350
340 GOTO 580
350 FOR I=1 TO 24 STEP 5
360 CALL HCHAR(I,1,96,32)
370 NEXT I
380 FOR I=3 TO 18 STEP 5
390 FOR J=3 TO 30 STEP 3
400 CALL HCHAR(I,J,98)
410 NEXT J :: NEXT I
420 CALL HCHAR(3,3,97)
430 CALL HCHAR(3,30,97)
440 CALL HCHAR(18,3,97)
450 CALL HCHAR(18,30,97)
460 DISPLAY AT(23,1):"J A W
B R E A K E R   "
470 CALL SPRITE(#10,104,2,(5
*8)-5,1,0,SPEED)
480 CALL SPRITE(#11,104,2,(1
0*8)-5,1,0,-SPEED-8)
490 CALL SPRITE(#12,104,2,(1
5*8)-5,200,0,SPEED+8)
500 CALL SPRITE(#3,112,CS-2,
(1*8)+1,125,0,INT(SPEED/2))
510 CALL SPRITE(#4,112,CS-4,
(6*8)+1,25,0,-INT(SPEED-8/2)
)
520 CALL SPRITE(#5,112,CS-10
,(16*8)+1,225,0,INT(SPEED+8/
2))
530 DISPLAY AT(7,5):"YOU HAV
E";CHA;" CHANCES!" :: DISPLA
Y AT(2,10):"LEVEL ";NIV :: F
OR T=1 TO 300 :: NEXT T :: D
ISPLAY AT(7,3):" " :: DISPLA
Y AT(2,1):"" :: DISPLAY AT(7
,5):" "
540 CALL SPRITE(#1,116,9,(11
*8)+1,125)
550 CALL SPRITE(#2,120,16,(1
1*8)+1,125)
560 LH=89 :: LV=125 :: PO=11
570 RETURN
580 LH=89 :: LV=125 :: PO=11
590 CALL JOYST(1,V,H)
600 IF H=-4 AND V=0 THEN LH=
LH+40 :: GOTO 780
610 IF H=4 AND V=0 THEN LH=L
H-40 :: GOTO 860
620 IF V=-4 THEN LV=LV-24 ::
 GOTO 960
630 IF V=4 THEN LV=LV+24 ::
GOTO 980
640 IF DENT=1 THEN LET DENT=
2 :: GOSUB 760 :: GOTO 690
650 IF COL=1 THEN LET TIM=TI
M+1
660 IF TIM>=20 THEN COL=0 ::
 TIM=0
670 IF COL=0 THEN CALL COLOR
(9,8,1):: CALL COLOR(#3,12,#
4,10,#5,4)
680 IF DENT=2 THEN LET DENT=
1 :: GOSUB 760 :: GOTO 690
690 IF PO=9 OR PO=10 THEN CA
LL COINC(#1,#3,24,SS):: IF S
S=-1 THEN GOTO 1090
700 IF PO=10 OR PO=11 THEN C
ALL COINC(#1,#4,24,SS):: IF
SS=-1 THEN GOTO 1090
710 IF PO=11 OR PO=12 THEN C
ALL COINC(#1,#5,24,SS):: IF
SS=-1 THEN GOTO 1090
720 IF CHA<=0 THEN 1400
730 GOTO 590
740 GOSUB 1000
750 GOTO 590
760 IF DENT=1 THEN CALL PATT
ERN(#2,120):: RETURN
770 IF DENT=2 THEN CALL PATT
ERN(#2,124):: RETURN
780 IF LH>=161 THEN LH=LH-40
 :: GOTO 640
790 CALL POSITION(#1,X,Y)
800 IF X=9 THEN PO=10 :: GOS
UB 950 :: GOTO 840
810 IF X=49 THEN PO=11 :: GO
SUB 950 :: GOTO 840
820 IF X=89 THEN PO=12 :: GO
SUB 950 :: GOTO 840
830 IF X=129 THEN PO=13 :: G
OSUB 950 :: GOTO 840
840 IF XXX=-1 THEN GOSUB 100
0
850 GOTO 640
860 IF LH<=1 THEN LH=LH+40 :
: GOTO 640
870 CALL POSITION(#1,X,Y)
880 IF X=9 THEN PO=9 :: GOSU
B 940 :: GOTO 920
890 IF X=49 THEN PO=10 :: GO
SUB 940 :: GOTO 920
900 IF X=89 THEN PO=11 :: GO
SUB 940 :: GOTO 920
910 IF X=129 THEN PO=12 :: G
OSUB 940 :: GOTO 920
920 IF XXX=-1 THEN GOSUB 100
0
930 GOTO 640
940 CALL COINC(#1,#PO,24,XXX
):: RETURN
950 CALL COINC(#1,#PO,32,XXX
):: RETURN
960 IF LV<=1 THEN LV=LV+24 :
: GOTO 640
970 GOSUB 1000 :: GOTO 640
980 IF LV>=232 THEN LV=LV-24
 :: GOTO 640
990 GOSUB 1000 :: GOTO 640
1000 CALL LOCATE(#1,LH,LV,#2
,LH,LV)
1010 GOSUB 1200
1020 IF COL=1 THEN CALL COLO
R(#3,14,#4,14,#5,14):: CALL
COLOR(9,14,1):: LET TIM=TIM+
1
1030 IF TIM>20 THEN COL=0 ::
 TIM=0 :: CALL COLOR(#3,12,#
4,10,#5,4):: CALL COLOR(9,8,
1)
1040 IF PO=9 OR PO=10 THEN C
ALL COINC(#1,#3,24,SS):: IF
SS=-1 THEN GOTO 1090
1050 IF PO=10 OR PO=11 THEN
CALL COINC(#1,#4,24,SS):: IF
 SS=-1 THEN GOTO 1090
1060 IF PO=11 OR PO=12 THEN
CALL COINC(#1,#5,24,SS):: IF
 SS=-1 THEN GOTO 1090
1070 IF COUNT>=40 THEN SPEED
=SPEED+4 :: NIV=NIV+1 :: COU
NT=0 :: GOTO 380
1080 RETURN
1090 IF COL=1 THEN 1180
1100 CALL PATTERN(#2,128)
1110 FOR I=1 TO 20
1120 FOR T=1 TO 10 :: NEXT T
1130 CALL MOTION(#2,16,0)
1140 CALL SOUND(-1,200*I,0,-
2,0):: NEXT I
1150 CALL MOTION(#2,0,0)
1160 CHA=CHA-1 :: IF CHA<=0
THEN 1400
1170 GOSUB 470
1180 CALL COLOR(#PO-7,1):: S
CORE=SCORE+25
1190 GOTO 590 :: GOTO 590
1200 CALL POSITION(#1,X,Y)
1210 IF X=9 THEN LET HO=3 ::
 GOTO 1260
1220 IF X=49 THEN LET HO=8 :
: GOTO 1260
1230 IF X=89 THEN LET HO=13
:: GOTO 1260
1240 IF X=129 THEN LET HO=18
 :: GOTO 1260
1250 END
1260 IF Y=5 THEN LET VO=3 ::
 GOTO 1360
1270 IF Y=29 THEN LET VO=6 :
: GOTO 1360
1280 IF Y=53 THEN LET VO=9 :
: GOTO 1360
1290 IF Y=77 THEN LET VO=12
:: GOTO 1360
1300 IF Y=101 THEN LET VO=15
 :: GOTO 1360
1310 IF Y=125 THEN LET VO=18
 :: GOTO 1360
1320 IF Y=149 THEN LET VO=21
 :: GOTO 1360
1330 IF Y=173 THEN LET VO=24
 :: GOTO 1360
1340 IF Y=197 THEN LET VO=27
 :: GOTO 1360
1350 IF Y=221 THEN LET VO=30
 :: GOTO 1360
1360 CALL GCHAR(HO,VO,CH)
1370 IF CH=97 THEN COL=1 ::
SCORE=SCORE+10 :: COUNT=COUN
T+1 :: CALL HCHAR(HO,VO,32):
: GOTO 1390
```

continued on page 14

```
100 REM CHANNEL 99 HAMILTON
USER'S GROUP
110 REM ZAPPER ZONE
120 REM A7-E0050-JGU
130 REM ****************
140 REM * IAIN JOHNSON *
150 REM ****************
160 CALL CLEAR :: RANDOMIZE
170 DIM D$(14),V(2,2),R(50),
Q(50),F(14)
180 CALL CHAR(104,"10547C7C7
C7C7C44",105,"0000FE7C7F7CFE
",106,"447C7C7C7C7C541",107,
"00007F3EFE3E7F")
190 CALL CHAR(96,"0",108,"40
E04",109,"56AA55AA55AA552A",
110,"4904A04D00224904",111,"
4100108200200441")
200 CALL CHAR(112,"183C7EFFF
F7E3C18",120,"0",136,"54AA41
824182552A")
210 CALL SCREEN(13):: CALL C
OLOR(9,4,4,10,4,4,11,5,4,12,
13,13,14,7,4)
220 DATA `ppp````ppp````pp``
````````,8,``pp``p``p`pp`pp
`ppp```ppp``,14,``p```p`````
``pp````````p```,5,````p````
````````p````p``p``,4
230 DATA ```pp`ppppp``p``pp`
pp```ppp`,15,``pp`pp`p`pp```
pp`pp```pp`p`,14,`pp`````p``
p```pp``````p`p`,8,`p``p```
pp``p```ppp``````p``,9
240 DATA `p```````p```````p
``p```p``,5,`pppp````ppp`ppp
``p`ppppppp`,18,````pp``````
`ppp`p`````ppp``,9,`pp````pp
`````p``p```pp``````,8
250 DATA ````p```````p```p``
```p``````,4,``pp```````pp```
````pp``````,6,````````````
````````````,0
260 DATA 0,0,-1,-1,-1,0,-1,1
,0,-1,0,1,1,-1,1,0,1,1
270 FOR H=0 TO 14 :: READ M$
,M :: F(H)=M :: D$(H)=M$ ::
NEXT H
280 FOR H=0 TO 8 :: READ L,M
 :: AY(H)=L :: AX(H)=M :: NE
XT H
290 V(1,2)=104 :: V(2,1)=105
 :: V(1,0)=106 :: V(0,1)=107
300 NM=0 :: DISPLAY AT(3,1):
D$(14):: FOR H=4 TO 20 STEP
2 :: I=INT(RND*14):: DISPLAY
 AT(H,1):D$(I):D$(I+1):: NM=
NM+F(I)+F(I+1):: NEXT H
310 I=INT(RND*14):: NM=NM+F(
I):: DISPLAY AT(22,1):D$(I):
D$(14):: CALL HCHAR(24,1,120
,96):: CALL VCHAR(1,31,120,9
6):: K,SC=0
320 NT=1 :: T=105 :: YV,XV=0
 :: CALL SPRITE(#1,105,2,17,
17,0,0)
330 CALL JOYST(1,A1,B1):: K=
K+1 :: IF ABS(A1)+ABS(B1)=4
THEN T=V(SGN(A1)+1,SGN(B1)+1
)ELSE 350
340 YV=-(B1+B1):: XV=A1+A1 :
: CALL PATTERN(#1,T):: CALL
MOTION(#1,YV,XV)
350 CALL POSITION(#1,DY,DX):
: IF DX<16 OR DX>234 OR DY<1
6 OR DY>176 THEN GOSUB 810 :
: GOTO 320
360 CALL KEY(1,KY,ST):: IF K
Y=18 THEN CALL MOTION(#1,0,0
):: ON T-103 GOTO 450,480,51
0,540 ELSE ON T-103 GOTO 370
,380,390,400
370 Y1=INT((DY-7)/8)+1 :: X1
=INT((DX-1)/8)+1 :: Y2=INT((
DY-7)/8)+1 :: X2=INT((DX+5)/
8)+1 :: GOTO 410
380 Y1=INT(DY/8)+1 :: X1=INT
((DX+14)/8)+1 :: Y2=INT((DY+
6)/8)+1 :: X2=INT((DX+14)/8)
+1 :: GOTO 410
390 Y1=INT((DY+14)/8)+1 :: X
1=INT((DX-1)/8)+1 :: Y2=INT(
(DY+14)/8)+1 :: X2=INT((DX+5
)/8)+1 :: GOTO 410
400 Y1=INT((DY+6)/8)+1 :: X1
=INT((DX-6)/8)+1 :: Y2=INT(D
Y/8)+1 :: X2=INT((DX-6)/8)+1
410 CALL GCHAR(Y1,X1,Z1):: C
ALL GCHAR(Y2,X2,Z2):: IF Z1=
112 THEN 420 ELSE IF Z2=112
THEN 430 ELSE 330
420 CALL HCHAR(Y1,X1,136)::
GOSUB 810 :: CALL HCHAR(Y1,X
1,96):: GOTO 440
430 CALL HCHAR(Y2,X2,136)::
GOSUB 810 :: CALL HCHAR(Y2,X
2,96)
440 NM=NM-1 :: IF NM THEN 32
0 ELSE 850
450 Y1=INT((DY-1)/8)+1 :: X=
INT((DX+1)/8)+1
460 CALL SPRITE(#2,108,2,DY+
2,DX+2,-56,0):: FOR Y=Y1 TO
1 STEP -1 :: CALL GCHAR(Y,X,
Z1):: IF Z1=112 THEN 580
470 NEXT Y :: GOTO 570
480 Y=INT((DY+2)/8)+1 :: X1=
INT((DX-1)/8)+1
490 CALL SPRITE(#2,108,2,DY+
3,DX+3,0,56):: FOR X=X1 TO 3
2 :: CALL GCHAR(Y,X,Z1):: IF
 Z1=112 THEN 580
500 NEXT X :: GOTO 570
510 Y1=INT((DY-1)/8)+1 :: X=
INT((DX+1)/8)+1
520 CALL SPRITE(#2,108,2,DY+
3,DX+2,56,0):: FOR Y=Y1 TO 2
4 :: CALL GCHAR(Y,X,Z1):: IF
 Z1=112 THEN 580
530 NEXT Y :: GOTO 570
540 Y=INT((DY+2)/8)+1 :: X1=
INT((DX-1)/8)+1
550 CALL SPRITE(#2,108,2,DY+
3,DX+2,0,-56):: FOR X=X1 TO
1 STEP -1 :: CALL GCHAR(Y,X,
Z1):: IF Z1=112 THEN 580
560 NEXT X
570 CALL MOTION(#1,YV,XV)::
CALL DELSPRITE(#2):: GOTO 33
0
580 CALL DELSPRITE(#2):: C=1
 :: I=0 :: GOSUB 790
590 Y1=INT((DY-1)/8)+1 :: X1
=INT((DX-1)/8)+1 :: Y2=INT((
DY+6)/8)+1 :: X2=INT((DX+6)/
8)+1
600 CALL GCHAR(Y1,X1,Z1):: C
ALL GCHAR(Y1,X2,Z2):: CALL G
CHAR(Y2,X1,Z3):: CALL GCHAR(
Y2,X2,Z4)
610 IF Z1=96 THEN CALL HCHAR
(Y1,X1,104)
620 IF Z2=96 THEN CALL HCHAR
(Y1,X2,104)
630 IF Z3=96 THEN CALL HCHAR
(Y2,X1,104)
640 IF Z4=96 THEN CALL HCHAR
(Y2,X2,104)
650 CALL GCHAR(Y-1,X-1,Z(1))
:: CALL GCHAR(Y-1,X,Z(2))::
CALL GCHAR(Y-1,X+1,Z(3)):: C
ALL GCHAR(Y,X-1,Z(4))
660 CALL GCHAR(Y,X+1,Z(5))::
 CALL GCHAR(Y+1,X-1,Z(6))::
CALL GCHAR(Y+1,X,Z(7)):: CAL
L GCHAR(Y+1,X+1,Z(8))
670 IF Z(1)+Z(2)+Z(3)+Z(4)+Z
(5)+Z(6)+Z(7)+Z(8)=768 THEN
CALL HCHAR(Y,X,96):: GOTO 73
0
680 FOR I=1 TO 8 :: ON INT(Z
(I)/8)-11 GOTO 710,690,700,7
10
690 GOSUB 810 :: GOTO 710
700 GOSUB 790
710 NEXT I
720 FOR H=-1 TO 1 :: CALL HC
HAR(Y-H,X-1,96,3):: NEXT H
730 Y=R(C):: X=Q(C):: C=C-1
:: IF C THEN 650
740 IF Z1=96 THEN CALL HCHAR
(Y1,X1,96)
750 IF Z2=96 THEN CALL HCHAR
(Y1,X2,96)
760 IF Z3=96 THEN CALL HCHAR
(Y2,X1,96)
770 IF Z4=96 THEN CALL HCHAR
(Y2,X2,96)
780 CALL MOTION(#1,YV,XV)::
IF NM THEN 330 ELSE 850
790 CALL HCHAR(Y+AY(I),X+AX(
I),136):: CALL SOUND(-750,11
0,6,-5,0):: C=C+1 :: R(C)=Y+
AY(I):: Q(C)=X+AX(I):: NM=NM
-1
800 SC=SC+100 :: DISPLAY AT(
1,12):SC :: RETURN
810 IF NT=0 THEN RETURN
820 FOR H=109 TO 111 :: CALL
 PATTERN(#1,H):: CALL SOUND(
-400,110,(H-109)*5+5,-5,(H-1
09)*5):: NEXT H :: SC=SC-100
0 :: NT=0
830 CALL DELSPRITE(#1):: IF
SC<0 THEN SC=0
840 DISPLAY AT(1,12):SC :: R
ETURN
850 FOR H=1 TO 500 :: NEXT H
 :: DISPLAY AT(4,1)ERASE ALL
:"THE CLOCK STOPPED AT;";K;"
YOUR FINAL SCORE IS; ";SC+(2
00-K)*10
860 CALL DELSPRITE(ALL):: DI
SPLAY AT(12,2)BEEP:"DO YOU W
ISH TO PLAY AGAIN?":TAB(12);
"Y/N?"
870 CALL KEY(3,KY,ST):: IF S
T=0 THEN 870 ELSE IF KY<>78
THEN 300
880 DISPLAY AT(12,6)ERASE AL
L:"HAVE A NICE DAY!" :: FOR
H=1 TO 800 :: NEXT H
890 END                    o
```

from page 13

```
1380 IF CH=98 THEN CALL HCHA
R(HO,VO,32):: SCORE=SCORE+1
:: COUNT=COUNT+1 :: GOTO 139
0
1390 DISPLAY AT(23,21)SIZE(6
):SCORE :: RETURN
1400 GOSUB 470
1410 DISPLAY AT(7,8):"  GAM
E OVER  "
1420 DISPLAY AT(20,6):"PLAY
AGAIN (Y/N)? "
1430 CALL KEY(0,KK,ST):: IF
ST=0 THEN 1430
1440 IF KK=ASC("Y")OR KK=ASC
("y")THEN CALL DELSPRITE(ALL
):: GOTO 100
1450 END
```

```
100 ! MENU MAKER produces a
MENU of programs on a disk
by reading the directory and
then writing the information
back to disk as a MERGE file
110 ! Type 'NEW' and MERGE
the output file into MEMORY,
 then save to disk as 'DSK1.
LOAD'
120 CALL CLEAR :: PRINT "PRO
GRAM STATUS.......WORKING" :
: CL$="CLEAR" :: DIM A$(20):
: OPEN #1:"DSK1.",INPUT ,REL
ATIVE,INTERNAL
130 DEF LN$(N)=CHR$(0)&CHR$(
N)
140 DEF DI$(R)=CHR$(162)&CHR
$(240)&CHR$(183)&CHR$(200)&C
HR$(LEN(STR$(R)))&STR$(R)&CH
R$(179)&CHR$(200)&CHR$(1)&ST
R$(COL)&CHR$(182)&CHR$(181)
150 DEF IF$(N)=CHR$(132)&"K@
"&CHR$(190)&CHR$(200)&CHR$(2
)&STR$(N)&CHR$(176)&CHR$(169
)&CHR$(199)&CHR$(LEN(A$(I-64
))+5)&"DSK1."&A$(I-64)
160 FOR I=0 TO 20
170 J=J+1 :: INPUT #1:A$(I),
B,C,D :: IF I=0 THEN 180 ELS
E IF J>=127 OR LEN(A$(I))=0
THEN 190 ELSE IF ABS(B)<>5 0
R A$(I)="LOADER" THEN 170
180 NEXT I
190 CLOSE #1 :: EN$=CHR$(181
)&CHR$(199)&CHR$(28)&"PRESS
<ERASE> TO END PROGRAM"&CHR$
(0):: COL=1 :: L=I-1 :: OPEN
 #2:"DSK1.CAT",VARIABLE 163
200 PRINT #2:LN$(1)&CHR$(157
)&CHR$(200)&CHR$(5)&CL$&CHR$
(0)
210 PRINT #2:LN$(2)&DI$(1)&C
HR$(199)&CHR$(28)&"CATALOG"&
RPT$(" ",12-LEN(A$(0)))&"DIS
KNAME-"&A$(0)&CHR$(0)
220 COL=8 :: FOR I=1 TO L ::
 PRINT #2:LN$(I+2)&DI$(12+I-
INT(L/2))&CHR$(199)&CHR$(3+L
EN(A$(I)))&CHR$(I+64)&"--"&A
$(I)&CHR$(0):: NEXT I
230 PRINT #2:LN$(L+3)&CHR$(1
62)&CHR$(240)&CHR$(183)&CHR$
(200)&CHR$(2)&"24"&CHR$(179)
&CHR$(200)&CHR$(1)&"1"&CHR$(
182)&CHR$(238)&EN$
240 PRINT #2:LN$(L+4)&CHR$(1
57)&CHR$(200)&CHR$(3)&"KEY"&
CHR$(183)&CHR$(200)&CHR$(1)&
"O"&CHR$(179)&"K@"&CHR$(179)
&"S@"&CHR$(182)&CHR$(0)
250 PRINT #2:LN$(L+5)&CHR$(1
32)&"S@"&CHR$(190)&CHR$(200)
&CHR$(1)&"O"&CHR$(176)&CHR$(
201)&LN$(L+4)&CHR$(0)
260 FOR I=64 TO L+64 :: PRIN
T #2:LN$(L+I-59)&IF$(I)&CHR$
(0):: NEXT I
270 PRINT #2:LN$(2*L+6)&CHR$
(132)&"K@"&CHR$(190)&CHR$(20
0)&CHR$(1)&"7"&CHR$(176)&CHR
$(157)&CHR$(200)&CHR$(5)&CL$
&CHR$(130)&CHR$(139)&CHR$(0)
280 PRINT #2:LN$(2*L+7)&CHR$
(134)&CHR$(201)&LN$(L+4)&CHR
$(0):CHR$(255)&CHR$(255):: C
LOSE #2 :: DISPLAY AT(23,21)
BEEP:"COMPLETE" :: END
```

```
100 REM DISK LABELS
        PRINTER OPTIONAL
110 IM$(1)="########## ### #
###### ### #  |  "
120 IM$(2)=IM$(1)&SEG$(IM$(1
),1,28)
130 DIM FILE$(50),TYP(50),SI
Z(50),CHARLEN(50)
140 CALL CLEAR
150 DIM TYPE$(5)
160 TYPE$(1)="DIS/FIX"
170 TYPE$(2)="DIS/VAR"
180 TYPE$(3)="INT/FIX"
190 TYPE$(4)="INT/VAR"
200 TYPE$(5)="PROGRAM"
210 INPUT "MASTER DISK (1-3)
?":I
220 I=INT(I)
230 IF (I<1)+(I>3)=-1 THEN 2
10
240 OPEN #1:"DSK"&STR$(I)&".
",INPUT ,RELATIVE,INTERNAL
250 INPUT #1:A$,J,J,K
260 DISPLAY "DSK";STR$(I);"
-DISKNAME= ";A$:"AVAILABLE="
;K;" USED= ";J-K
270 DISPLAY :"FILENAME   SIZ
E    TYPE    P";"---------
---- --------- -";
280 COUNT=1
290 INPUT #1:FILE$(COUNT),TY
P(COUNT),SIZ(COUNT),CHARLEN(
COUNT)
300 IF LEN(FILE$(COUNT))=0 T
HEN COUNT=COUNT-1 :: GOTO 38
0
310 DISPLAY :FILE$(COUNT);TA
B(12);SIZ(COUNT);TAB(17);TYP
E$(ABS(TYP(COUNT)));
320 IF ABS(TYP(COUNT))=5 THE
N 350
330 L$=" "&STR$(CHARLEN(COUN
T))
340 DISPLAY SEG$(L$,LEN(L$)-
2,3);
350 IF I>0 THEN 370
360 DISPLAY TAB(28);"Y";
370 COUNT=COUNT+1 :: GOTO 29
0
380 CLOSE #1
390 DISPLAY AT(24,1):"PRINT
LABEL?(Y/N) Y" :: ACCEPT AT(
24,19)BEEP SIZE(-1)VALIDATE(
"YN"):RE$
400 IF RE$="Y" THEN GOSUB 44
0
410 DISPLAY AT(24,1):"MORE
DISKS?(Y/N) Y" :: ACCEPT AT(
24,19)BEEP SIZE(-1)VALIDATE(
"YN"):RE$
420 IF RE$="Y" THEN 240
430 END
440 OPEN #2:"PIO.CR" :: PRIN
T #2:CHR$(27)&"O"&CHR$(15)&C
HR$(27)&CHR$(71)
450 CLOSE #2 :: OPEN #2:"PIO
"
460 PRINT #2:CHR$(14);A$&" A
VAIL="&STR$(K)&" USED="&STR$
(J-K)
470 PASS=1
480 FOR X=1+(PASS-1)*18 TO M
IN(COUNT,9+(PASS-1)*18)
490 Z1$="  "&STR$(CHARLEN(X)
):: IF ABS(TYP(X))=5 THEN RL
1$="" ELSE RL1$=SEG$(Z1$,LEN
(Z1$)-2,3)
500 IF TYP(X)<0 THEN P1$="P"
 ELSE P1$=""
510 IF X+9>COUNT THEN GOSUB
590 ELSE GOSUB 550
520 LCT=LCT+1 :: NEXT X :: G
OSUB 610 :: IF COUNT<=PASS*1
8 THEN CLOSE #2 :: RETURN
530 PASS=PASS+1 :: PRINT #2:
CHR$(14);A$&" (Cont.) LABEL
#"&STR$(PASS)
540 GOTO 480
550 Z2$="  "&STR$(CHARLEN(X+
9)):: IF ABS(TYP(X+9))=5 THE
N RL2$="" ELSE RL2$=SEG$(Z2$
,LEN(Z2$)-2,3)
560 IF TYP(X+9)<0 THEN P2$="
P",ELSE P2$=""
570 PRINT #2,USING IM$(2):FI
LE$(X),SIZ(X),TYPE$(ABS(TYP(
X))),RL1$,P1$,FILE$(X+9),SIZ
(X+9),TYPE$(ABS(TYP(X+9))),R
L2$,P2$
580 RETURN
590 PRINT #2,USING IM$(1):FI
LE$(X),SIZ(X),TYPE$(ABS(TYP(
X))),RL1$,P1$
600 RETURN
610 FOR Y=1 TO 11-LCT :: PRI
NT #2:"" :: NEXT Y :: LCT=0
:: RETURN
```

*

continued from page 12

```
1830 CALL HCHAR(BOOG8,BOOG8+
3,32)
1840 CALL HCHAR(BOOG8+1,BOOG
8+4,113)
1850 CALL HCHAR(BOOG8+2,BOOG
8+5,112)
1860 CALL COLOR(BOOGEN+3,7,1
)
1870 GOTO 2030
1880 CALL HCHAR(BOOG8+1,BOOG
8+4,32)
1890 CALL HCHAR(BOOG8+2,BOOG
8+5,128)
1900 BOOG8=1
1910 GOTO 900
1920 REM *****************
1930 REM * BOOGEN KILLER *
1940 REM *****************
1950 CALL SOUND(-50,110,0,-6
,0)
1960 IF POINT=BOOGEN THEN 19
80
1970 GOTO 890
1980 DEAD=1
1990 SC=SC+1
2000 IF SC<>70-HARD*10 THEN
2020
2010 HARD=HARD-1
2020 GOTO 920
2030 CALL SOUND(-600,-5,8)
2040 GOTO 740
2050 CALL CLEAR
2060 CALL CHAR(89,"004444442
8101010")
2070 CALL CHAR(65,"000814222
2372222")
2080 CALL CHAR(73,"001010101
010101")
2090 PRINT TAB(4);"SORRY YOU
'RE DEAD"
2100 PRINT "YOU KILLED";SC;"
BOOGENS"
2110 PRINT : : :" AGAIN Y-N"
2120 FOR I=3 TO 8
2130 CALL COLOR(I,2,1)
2140 NEXT I
2150 CALL KEY(3,K,S)
2160 IF S=0 THEN 2150
2170 IF K=89 THEN 170
2180 END
```

# Starting a Database from Scratch.

PART 2

with CHRIS BUTTNER

In the September issue I started out by giving you a program to create the data file which would be used by the main database program. This month, I have some of the simpler "modules" which will be incorporated in the main program.

I hope by now you have some understanding of how the large program is broken down into smaller, more manageable parts. If you are to get the most out of our little exercise you should try to write some of the modules yourself. It doesn't matter if your version does not work at first. At least you will have tried to do something and you can check your efforts against the published routines. That way, you will see where you went wrong and the learning experience will be more rewarding.

Now down to business. One of the first things needed will be an on-line help facility. A simple approach is to accept the command letter "H" as input and on receipt, display a summary of the commands. It is important that you understand how the modules are constructed initially so I will use this one as an example. Since the subprogram is going to provide help, let's call it HELP, and start with a line number of 9000 like so:

```
9000 SUB HELP
```

From here, I have inserted data statements which provide the information to be displayed and the co-ordinates at which the various lines of text appear. This if followed by the RESTORE command to ensure data is read starting at the correct line number. A simple loop is used to read and display all information. The CALL KEY command is then used to check for a particular key-press which is the trigger to continue. Until then, the information is displayed on screen and the user can take as long as he likes to read the help screen before proceeding. The last command is SUBEND.

Now that the subprogram is written, it is saved in the pure subprogram form. Now comes the moment of truth when the debugging process is started. Type into the program the following line:

```
1 CALL HELP
```

Now in command mode type the following commands each followed by ENTER:

```
CALL CLEAR
RUN
```

If all is well, your subprogram should do exactly as you want. It not, examine the screen and see what is missing/wrong. Edit your program and run it again. Repeat this process until you have it right then delete line 1 and resave your subprogram. At this stage I recommend you save in two ways: firstly as a conventional program and secondly in MERGE format which you will require eventually when the various subprograms are merged into the final work. To distinguish between the two, save the merge version with the suffix M.

We also need to know when our input to the program is wrong so there is another subprogram this time called ILLEGAL. It is developed in the same manner. In the published version at line 3020 it flashes the warning message on the screen then clears before re-displaying the warning message. If you find this flashing disconcerting, delete line 3020. (The purpose of the flashing is to attract your attention. Other methods could be used such as making a sound).

The WIPE subprogram which I mentioned in the previous article is also included. It's purpose is to clear the screen from line 4 down to the bottom.

I have also included a larger subprogram called at this stage, COMMAND. It takes the input from line 1, analyses it and then calls the necessary subprogram. There are a number of parts to this subprogram which perform the following functions:

(a) Stripping space characters from the end of the command so that out input is always consistent;

(b) Converting the first character of each command to upper-case so command letters are not case dependant. This has one side effect - you must be consistent in the way to enter your data and how you query the database. For example, if you entered a name as McShane, the program would not retrieve your entry if you listed it MCSHANE.

(c) Checking for the single letter commands H for help and E for exit;

(d) Checking for a space between the command letter and command information;

(e) Giving the warning message if a valid command letter is not used.

All of these modules are commented so you should not have any problem to seeing exactly what is happening.

Other subprograms are needed to allow you to Query the database, display information for a particular record on the screen remove and update various records. To get you started in programming, I have provided the Query and Display modules in outline form only. Try to work out the various steps then have a go with your own version. I will have fully written and commented versions in part 3.

If you make a genuine effort and still have problems, send me a not or button-hole me at the December meeting. Remember, no matter how lucid this or any other tutorial may be, personal experience will be your greatest teacher.

END OF NARRATIVE

START OF VARIOUS SUBPROGRAMS.

```
2000 REM *** Start of main c
ommand Subprogram.

2010 SUB COMMAND

2020 ACCEPT AT(1,1)SIZE(-28)
:A$ ! Get input from line 1
of screen allowing 28 charac
ters total

2030 L=LEN(A$)! Calculate th
e length of the input string

2040 IF SEG$(A$,L,1)<>" " TH
 2080 ! If the last charac
ter is NOT a blank space, sk
ip the next two lines.

2050 A$=SEG$(A$,1,L-1)! Stri
p of the last blank space

2060 GOTO 2030 ! Loop back a
nd check for more blank spac
es to be removed

2070 REM ** The next few lin
es convert the first charact
er to upper case if necessar
y.
```

continued on page 6

# REMOVING PROTECTION

*I found it annoying being unable to copy other peoples programs...*

by George Meldrum

The Extended BASIC language provides a protection for programs by including the PROTECTED option when saving to cassette or disk, e.g. SAVE CS1,PROTECTED. Personally I found it annoying being unable to list, edit, or copy other peoples programs. Even more aggravating was the knowledge discovered later that those with expanded systems could get around this problem with a simple CALL LOAD command. Well now I have an expanded system too so I can just sit back. Whenever a protected program pops up a quick CALL LOAD puts an end to it. This article however, is a promise to myself that all system users be given equal opportunity to paralyze the protection pest.

Below is listed three methods of removing protection. The less equipment you have the more complicated it becomes, but it is all good fun.

Method (1) users with 32K memory expansion.
Method (2) users with cassette and Mini Memory.
Method (3) users with cassette recorder only.

Method (1)

Select Extended BASIC and load the protected program.

```
>OLD CS1
>OLD DSK1.filename
```

Now type the following :-

```
>CALL INIT
>CALL LOAD(-31931,0)
```

**That is it, the protection has now been removed.**

**Method (2)**

Insert the Mini Memory module and select TI BASIC. Execute these program steps making reference to the notes that follow.

```
>CALL LOAD(28672,90,165,255,80,0,4,0,0)
>OLD CS1.123
>CALL PEEKV(1805,A,B,C,D)
>CALL LOAD(28680,256-SGN(B)-A,256-B,C,D)
>OLD MINIMEM
>SAVE CS1
```

That is it, the protection has now been removed and saved to cassette tape. All that is needed now is to insert the Extended BASIC module and reload the program.

**Notes for each step used in Method (2):**

```
>CALL LOAD(28672,90,165,255,80,0,4,0,0)
```

This makes Mini Memory think that it is storing a BASIC program four bytes in length. If several tapes are to be done then this statement need only be executed once at the start of a session.

```
>OLD CS1.123
```

Yes that is "CS1.123". The .123 is necessary to make the device name seven characters long - the same length as "MINIMEM". Follow the cassette prompts and load the protected program. After the tape is loaded an I/O ERROR 50 message will appear.

```
>CALL PEEKV(1805,A,B,C,D)
>CALL LOAD(28680,256-SGN(B)-A,256-B,C,D)
```

This reads the first four bytes of the program file in VDP memory and transfers them to Mini Memory's memory. Also it performs a two's complement on the first word (two bytes).

```
>OLD MINIMEM
```

There is the usual delay when loading a program but when the cursor reappears continue with :-

```
>SAVE CS1
```

Programs written in Extended BASIC do not LIST very well in TI BASIC. However when the newly recorded program is reloaded in Extended BASIC it functions normally and can be LISTed, EDITed, and SAVEd.

Method (3)

All steps in this method are done by using TI BASIC only. Remove any modules then select TI BASIC. To start type in the following program and RUN it.

```
100 REM  UNPROTECTOR AID
110 REM  g.meldrum tishug
120 REM
130 DATA 0,3,63,139,63,136
140 DATA 63,255,0,10,63,141
150 DATA 3,154,32,0
160 REM
170 FOR EYE=1 TO 16
180 READ N
190 A$=A$&CHR$(N)
200 NEXT EYE
210 CALL CLEAR
220 DISPLAY "PREPARE TO RECORD"::"DUMMY PROGRAM":::::
230 OPEN #1:"CS1",OUTPUT,FIXED 64
240 PRINT #1:A$
250 CLOSE #1
260 END
```

It is not necessary to save this 'UNPROTECTOR AID' program. When you run this program it will display prompts for you to record a short file to tape. The short 'dummy program' that the above program produces should be kept on a separate tape as it will be used for each unprotect session.

Next type in the following program. It too shall be used in each unprotect session so record this program on the tape after the dummy program.

```
100 REM  PIRATE FACTORY
110 REM  g.meldrum tishug
120 REM
130 CALL CLEAR
140 DISPLAY "PREPARE TO LOAD MINI-PROGRAM":::::
150 OPEN #1:"CS1",INPUT,FIXED 128
160 INPUT #1:C$,
170 A$=A$&C$
180 IF LEN(A$)=128 THEN 210
190 A$=A$&","
200 IF LEN(A$)=128 THEN 210 ELSE 160
210 CLOSE #1
220 V1=ASC(SEG$(A$,65,1))
230 V2=ASC(SEG$(A$,66,1))
240 T1=255-V1
250 T2=256-V2
260 B$=CHR$(T1)&CHR$(T2)&SEG$(A$,67,62)
270 CALL CLEAR
280 DISPLAY "OUTPUTTING PIRATE HEADER":::::
290 OPEN #1:"CS1",OUTPUT,FIXED 64
300 PRINT #1:B$
310 CLOSE #1
320 END
```

Be sure to include the comma at the end of line 160. Save the above 'PIRATE FACTORY' program with SAVE CS1 but do not run it at this stage.

Now you have all the elements ready to go for a swashbuckling tour through your protected cassette tape programs. Having completed the procedures above you can now issue the following steps for each program that needs unprotecting.

a) Type the following command and load the protected cassette program.

```
>OLD CS1.123456789=123456789=123456789=123456789=12
3456789=123456789=123
```

continued on page 18

# SPRITES

PART 2 - by Jim Peterson

Several sprites can be created by one statement, such as CALL SPRITE(#1,42,16,10,10,#2,65,2,20,20).
The pattern of several sprites can be changed at once by CALL PATTERN(#1,CHAR,#2,CHAR) - this is very useful when changing the pattern of a character which has been created from two or more sprites.

Several sprites can be set in motion simultaneously, or have their motion changed simultaneously, by CALL MOTION(#1,RV,CV,#2,RV,CV,#3,RV,CV) etc. This is also very useful when moving a character formed of two or more sprites.

Several sprites can be recolored simultaneously with CALL COLOR(#1,C,#2,C) etc.
Several sprites can be relocated together by CALL LOCATE(#1,DOTROW,DOTCOL,#2,DOTROW,DO TCOL) etc.
The position of more than one sprite can be found at one time by CALL POSITION(#1,DOTROW1,DOTCOL1,#2,DOTRO W2,DOTCOL2), etc.

A sprite can have only one color, unlike a screen character which can have a foreground and background color. Any dots which are not "turned on" in the character being used for the sprite will be transparent. However, a sprite with a higher number, using a redefined character with all dots turned on and of a different color, can be created at the same dotrow and dotcolumn, giving the illusion of a sprite with foreground and background color. Up to 4 sprites can be stacked in this way to create a multicolored sprite effect. If the sprite is stationary, colored graphics behind all 4 sprites can give the illusion of even more colors.

Sprites always appear to be in front of screen graphics, and lower-numbered sprites always apear in front of higher
numbered sprites. However, by skillful swapping of sprites, remarkable 3-D effects can be created, seeming to show a sprite passing before and then behind another, or before and then behind a graphics object.

Another way to simulate 3D is to place a second higher-numbered sprite behind the first, of the same pattern but of a darker color, and offset by a few dotrows downward and to the side, so that when both are set in motion the one appears to be flying above the surface with the second following as its shadow.

Sprites can also be used to add an apparent third color to screen graphics, which can have only two colors in one character.

It is difficult to create the impression of curved lines with redefined characters because they are composed of dots rather than lines. This becomes even more obvious in sprite magnifications 2 and 4, when each dot is magnified into 4 dots. A circle will appear more round, and of the same size, if it is composed of 4 redefined characters in magnification 3 than of one character in magnification 2.

Larger figures can be created using several sprites placed next to each other, providing that not more than four are in a row horizontally. These can be of several colors, and can be set in motion simultaneously.

Although it is stated that sprites, once set in motion, will continue to move regardless of what the program is doing, this is not quite true. If the program is doing a lot of calculating, the sprite motion will be jerky and irregular.

By setting a sprite in motion, and using a loop to change it through a series of patterns, remarkable animated graphics can be created, in much the same way that cartoon movies are made.

It is difficult to control motion exactly with CALL MOTION. For more precise control, sprites can be moved from one point to another, dot by dot, by using CALL LOCATE within a loop, such as FOR DC=1 to 100 :: CALL LOCATE(#1,50, DC):: NEXT DC. This movement will be very smooth but slow; adding a STEP 2 or STEP 3 will make it faster but less smooth.

If you have Memory Expansion, CALL LOAD(-31806,96) will freeze all sprite motion and CALL LOAD(-31806,0) will release all sprites to their normal motion. By first freezing the motion and then creating up to 28 sprites with predefined motion, all can be set into motion at once, creating some very remarkable effects. o

---

continued from page 17

After loading an I/O ERROR 50 message must be displayed.

b) Now type >OLD CS1 and load the 'dummy program'.

c) When the cursor reappears type >SAVE CS1 and save to tape. Label this program as 'step c'.

d) Load and run the 'PIRATE FACTORY' program with the commands >OLD CS1 and >RUN.

e) When the prompt to load the 'MINI-PROGRAM' appears then load the short program file SAVEd in step (c) i.e. the 'step c' program.

f) The next step of the 'PIRATE FACTORY' program is to save a 'PIRATE HEADER' to tape. Follow the screen prompts and save to tape which can be labelled 'pirate header'.

g) Type >OLD CS1 and load the original protected program.

h) After the I/O ERROR 50 message appears type >OLD CS1 and load the 'pirate header' program that that was saved in step (f).

i) After the cursor reappears type >SAVE CS1 and save the finally unprotected program.

The program saved in step (i) is the unprotected version of the original program. You could LIST the program while you are still in TI BASIC but as Extended BASIC programs do not list very well that way it would be better to wait. Wait that is, until you insert the Extended BASIC module and reload in an Extended BASIC environment. o

*

*"We need someone who'll go to the users' group meetings to break some pirates' knuckles."*

**Welcome to *Tidbits Six!***
**Presented by *Wade Bowmer.***

This month, I would like to talk about Extended BASIC's prescan commands, keycodes in key units 4 and 5 (Pascal and BASIC scans), and "crunching" Extended BASIC Programs- not necessarily in that order.

**Shortening the Prescan.**

You are aware of the Prescan commands in Extended BASIC, aren't you? You know, **!@P-** and **!@P+**, and remember that the Extended BASIC supplement describes how to make good use of the prescan commands?

I've recently realized the true power of these little statements. Recall that only one reference to a **CALL** or a variable is needed in the prescan, so that means that **DIM** and **OPTION BASE 1** and even **SUB** are only understood during prescan. So, that means that they don't even need to be in an execution path!

Now what does all that boil down to? It means that you can reduce the amount of time taken on the prescan by following the given layout. Place **!@P-** as the very first statement in your program, then put **!@P+** on a line all by itself, after all the top level code, but before any subprograms are defined, and then follow it with a list of all variables used (just separate them with commas), then a list of all subprogram **CALL**s, entered like this: **CALL CHAR :: CALL CLEAR :: CALL HCHAR**, then any **OPTION BASE 1**'s and **DIM**'s, and finally all the **DEF**s.

I think that I need to diagramate, here.

```
2 !@P-
10 !Main Program starts here.
5000 !(or some such suitable number) First
line after all the toplevel code.
5005 !@P+
5010 A,B,C,P$,ZZ
5020 CALL CLEAR :: CALL SCREEN :: CALL CHAR
:: CALL CURSOR
5100 SUB CURSOR(R,C,K)! User-defined
subprograms begin here.
```

The variables are simply those that are used, **A,B,C,P$** and **ZZ** are just names plucked out of the air. The same goes for the subprogram names. If you used **CALL MOTION**, put it in the list. If you created **CALL HELP**, put it in the list, too.

Tony McGovern, in his *Extended Tutorial*, mentioned that "...if you **CALL** a subprogram after you define it, then including its **SUB** in the prescan is sufficient." (My emphases.) *I* haven't had the chance to explore that, although I do not define subprograms earlier in the program than I need them.

One point, by arranging the program the way I have described it, The program will *not* **RUN** under Extended BASIC Version 100. (The prescan commands exist in Version *110* which you all should have, anyway.)

Another statement that should be placed in the prescan is **DEF**. It, too, does not need to be in an execution path, but can be put on the same line as any **DIM**'s.

Maybe, you're wondering just what I'm leading up to, or for that matter, whether I'm leading up to anything at all. Well, I am. *Careful planning of a program with respect to the prescan, can drastically improve the program's performance before it even starts executing.* I have read of Extended BASIC programs that nearly convince the buyer that it has "locked-up" their computer!

**Making and Saving Room.**

I said that I would be presenting another section of Making and Saving Room in a later article.

**3. Crunching the Program.**

Some of you, mainly those of you who have been members of TIsHUG for some time, will remember *Balloon Voyage* in *The Bumper Book #2*. It was one of the programs that I typed in. However, it was slightly too large for Extended BASIC to handle confidently, and so often suffered "automatic editing", or more correctly, random corruption, whenever I set about altering it.

Now that my programming is at a high level of effiency, I can look at *Balloon Voyage* and be shocked at how open the code really is! So I set about "crunching" it. The result is a program that fits much better in the VDP RAM, and suffers "automatic editing" no longer.

What I'm attempting to say, is that programs save space when as many statements as possible are put on one line. Each line takes up *6* bytes of room, plus the actual tokenized program line. The 6 bytes consist of 4 bytes in the line number table, 2 for line number, 2 for where it is in memory, another byte before the address given in the table that states the length of the line, and a **0** byte that heralds the end of the line.

For example, if you enter line 100 as follows: **100 CALL CLEAR**. That takes up *6+1+7= 14 bytes*. (**CALL** is 1 byte, **CLEAR** comprises the token for a literal constant, another byte for the length of it and 5 bytes for *CLEAR*)

If you add another line, **110 CALL SCREEN(5)**, that takes up *6+1+8+1+3+1= 20 bytes*. (*6* overhead, 1 for **CALL**, 8 for **SCREEN**, 1 for the **(**, 3 for the **5** and 1 for the **)**).

In total, that's *14+20= 34 bytes*. Now, if you had typed **100 CALL CLEAR :: CALL SCREEN(5)**, instead, that's 6 bytes overhead, 8 for **CALL CLEAR**, 1 for the **::**, and 14 for **CALL SCREEN(5)**. And that sums to just *29 bytes*.

It might seem a bit minimal, only 5 bytes difference, but that's only with 2 statements, neither taking a lot of memory. And anyway, usually you would want to put more than those two statements on one line anyway. Changing the colour of the entire character set is often done at the same time. But if the same practice is done on *every line* of a *400 line program*, the memory savings are sure to be phenomenal! Surely, the double colon wasn't put in Extended BASIC just to provide a convenience, but to save memory. (You may dispute that, but if TI wrote Extended BASIC, they must have known that pushing the use of the double colon to its limit would save memory in abundance.)

If I may make the reference, a Younger Set member wrote in, one month, wanting someone with the Memory Expansion to finish a game program that became too big for his, VDP RAM only, computer. No offense meant, and this is meant to be constructive criticism, but that program must have had open or inefficient coding. (Open coding means that the code looks like it is in TI-BASIC, i.e. one or two statements per line! When many more than that should be per line!)

Programs that are so big that they require **CALL FILES(1)** before they can be loaded off disk can also benefit, for if they are crunched as much as possible, then **CALL FILES(1)** may become obselete.

There is one disadvantage. (There had to be one!) Unfortunately, it makes the program feindishly difficult for us humans to read. Never mind how much space it saves, it can make for hard reading. On the positive side, if it is a listing made in 80 columns by **LIST "PIO"** or **LIST "RS232/1"**, then the readability problem is very much solved. Alternatively, if the program that converts an 80 column listing as a disk file, to a 28 column listing also highlights the line numbers at the start of each line, (Hint! Hint!) than the problem is also solved.

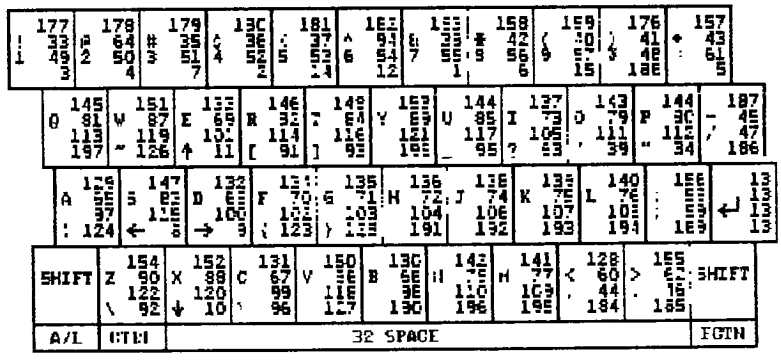
continued on p 20

The readability problem comes only because line numbers at the beginning of program lines can be hard to find when the listing consists of many long lines.

I'm sure that you're all aware of the trick of how to get more than 5 screen lines on a program line, particularly since that technique was discussed at length in last month's **Tidbits**. It needs to be coupled with crunching to be really useful.

**Key Board Full of Numbers.**

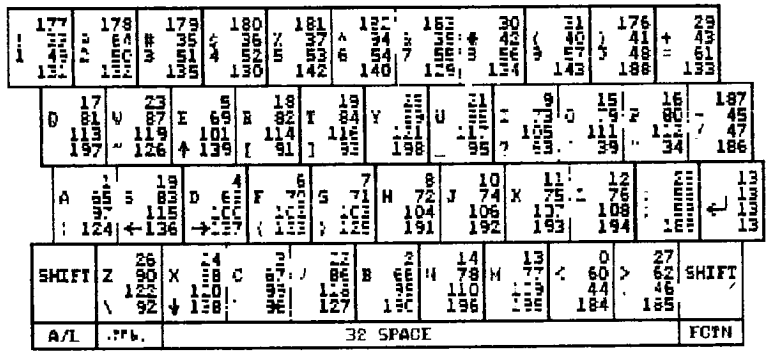
The Editors didn't print this section the first time I included it in one of my articles. It may have been too small, but I hope this one gets printed.




The first scan is the **BASIC Scan**, key-unit 5. The top keycode is the *Control code*, i.e. the key pressed with **CTRL**. The second top is (of course) **SHIFT**. The second bottom is the *regular* or *base keycode*, and the bottommost keycode is the *Function* keystroke.

(The lowest row is squashed because I was a bit short of room.)

The next layout is called the **Pascal Scan**, key-unit 4.




**Your Write.**

Recently, I received from George Meldrum a tape contining an interesting program. It had not been created in **Extended BASIC**, for line 110 contains unkeyable characters. Fortunately, it was quite easy to find the location of this line in memory and **CALL PEEK** it.

However, it caused a curious effect. If it was loaded after a *"Master Screen Visit"*, then it ran correctly. But if *another* program had been loaded in *before* it, but *after* **Extended BASIC** had been selected, then it faltered, producing any sort of error in line 110. Which is what George Meldrum wrote to me about.

But I discovered that my tape player needed head realignment to correctly load the tape. Sure, no trouble, only then readjustment was necessary to load one of my tapes. I finally decided to re**SAVE** this mysterious program with the normal setting on my equipment.

That done, the program suddenly works properly, every time! Oh dear...

So, I'm very sorry to not be able to offer a concrete solution, George, but it seems to me that if you resaved it as per your own equipment, then it would not be erratic in operation. Once again, sorry, but in fairness, I have just finished my Year 11 yearly exams, so I could not find suitable time for experimentation.

All up, though, I am quite happy to receive and at least look at any of your problems. And I still do not have a disk drive, although I hope to remedy that in the coming months.

*Enjoy*.....!

ERRATA! ERRATA! ERRATA! ERRATA! ERRATA!

In Tidbits Five (October '87), I inadvertently reversed two tokens.

Token 16Ø is CLOSE,
and 161 is SUB.

Please mark this in on your copy of October's TND. Very Sorry.....

# Tigercub Tips

## 45

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

Distributed by Tigercub Software to TI-99/4A Users Groups for promotional purposes and in exchange for their newsletters. May be reprinted by non-profit users groups, with credit to Tigercub Software.

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.

Here is a versatile printer utility which will accept all printer control codes, print in 1 to 5 columns with choice of column separation and margin width, allow alternate margins and pause at end of page to turn paper over, and will load and print a diskfull of files one after another. It is set up for the Gemini 10X and may require modification for other printers.

```
100 DIM M$(400),F$(50)
110 GOTO 150
120 K,ST,SET,S,P$,P,CL,DW$,S
S$,I$,D$,E$,NC,CW,TC,TA,TX,A
V,CS,S$,LT,A$,LSP,LP,RM,OK$,
QQ$,X,F$(),SL,F,IP,M$(),T$,F
LAG,J,PP,LT$
130 CALL CLEAR :: CALL KEY :
: CALL COLOR :: CALL SCREEN
:: CALL SOUND
140 !@P-
150 CALL CLEAR :: CALL KEY(3
,K,ST):: ON WARNING NEXT
160 FOR SET=0 TO 14 :: CALL
COLOR(SET,2,8):: NEXT SET ::
 CALL SCREEN(5)
170 DISPLAY AT(3,6):"TIGERCU
B PRINTALL": :TAB(7);"Copyri
ght 1987":TAB(6);"Tigercub S
oftware" !programmed by Jim
 Peterson
180 DISPLAY AT(12,1):"May be
 distributed without":"restr
iction providing that":"no p
rice or copying fee is":"cha
rged."
190 DISPLAY AT(18,7):"TURN P
RINTER ON!"
200 DISPLAY AT(20,8):"PRESS
ANY KEY" :: DISPLAY AT(20,8)
:"press any key" :: CALL KEY
(0,K,S):: IF S=0 THEN 200 EL
SE CALL CLEAR
210 DISPLAY AT(12,1):"PRINTE
R DESIGNATION?" :: ACCEPT AT
(14,1)BEEP:P$ :: IF POS(P$,"
.LF",1)=0 THEN P$=P$&".LF"
220 ON ERROR 230 :: OPEN #1:
P$,VARIABLE 255 :: ON ERROR
STOP :: PRINT #1:CHR$(27);"@
" :: CALL CLEAR :: GOTO 240
230 DISPLAY AT(20,1):"CANNOT
 OPEN PRINTER!" :: RETURN 21
0
240 DISPLAY AT(12,1):"PRINT
SIZE?": :" (1) PICA":" (2)
 ELITE":" (3) CONDENSED"
250 ACCEPT AT(12,13)VALIDATE
("123")SIZE(1):P :: PRINT #1
:CHR$(27);"B";CHR$(P);
260 !The values 80, 96 and 1
36 in the next line are the
maximum number of pica, elit
e and condensed characters p
er line on Gemini 10X
270 !Change as necessary for
 your printer!
280 CL=(P=1)*80+(P=2)*96+(P=
3)*136 :: CL=ABS(CL)
290 DISPLAY AT(12,1)ERASE AL
L:"DOUBLE-WIDTH? (Y/N) N" ::
 ACCEPT AT(12,21)SIZE(-1)VAL
IDATE("YN")BEEP:DW$ :: IF DW
$="Y" THEN PRINT #1:CHR$(27)
;"W";CHR$(1);:: CL=CL/2
300 DISPLAY AT(12,1)ERASE AL
L:"SUPERSCRIPT? (Y/N) N" ::
ACCEPT AT(12,20)SIZE(-1)VALI
DATE("YN")BEEP:SS$ :: IF SS$
="Y" THEN PRINT #1:CHR$(27);
"S";CHR$(0);
310 DISPLAY AT(12,1)ERASE AL
L:"ITALICS? (Y/N) N" :: ACCE
PT AT(12,16)VALIDATE("YN")SI
ZE(-1)BEEP:I$ :: IF I$="Y" T
HEN PRINT #1:CHR$(27);"4";
320 DISPLAY AT(12,1)ERASE AL
L:"DOUBLE-STRIKE? (Y/N) Y" :
: ACCEPT AT(12,22)VALIDATE("
YN")SIZE(-1)BEEP:D$ :: IF D$
="Y" THEN PRINT #1:CHR$(27);
"G";
330 IF P<>3 AND P<>4 THEN DI
SPLAY AT(12,1):"EMPHASIZED?
(Y/N) Y" :: ACCEPT AT(12,19)
VALIDATE("YN")SIZE(-1)BEEP:E
$ :: IF E$="Y" THEN PRINT #1
:CHR$(27);"E";
340 DISPLAY AT(12,1)ERASE AL
L:"NUMBER OF COLUMNS? (1-5)"
 :: ACCEPT AT(12,26)VALIDATE
("12345")SIZE(1)BEEP:NC
350 DISPLAY AT(12,1):"COLUMN
 WIDTH (NUMBER OF": :"CHARAC
TERS?" :: ACCEPT AT(14,13)VA
LIDATE(DIGIT)BEEP:CW
360 TC=NC*CW :: TA=CL-TC ::
TX=TC+NC*2-2
370 IF TX<=CL THEN 390 :: DI
SPLAY AT(18,1):STR$(NC)&" co
lumns of "&STR$(CW)&" charac
ters":"plus 2-column spacing
 equals"
380 DISPLAY AT(20,1):STR$(TC
)&" characters; maximum":"av
ailable in print size":"sele
cted is "&STR$(CL)&".":"****
Please reselect****" :: GOTO
 240
390 AV=INT(TA/(NC-1)):: DISP
LAY AT(12,1)ERASE ALL:"COLUM
N SEPARATION?":"MINIMUM 2":"
MAXIMUM "&STR$(AV)&" AVAILAB
LE ":"2"
400 ACCEPT AT(15,1)VALIDATE(
DIGIT)SIZE(-2)BEEP:CS :: IF
CS<2 OR CS>AV THEN 32767 ELS
E S$=RPT$(" ",CS)
410 TA=TA-CS*(NC-1):: IF TA<
2 THEN 450
420 DISPLAY AT(12,1)ERASE AL
L:"LEFT MARGIN WIDTH?": :"MA
```

continued on page 22

```
XIMUM "&STR$(TA)&" AVAILABLE
" :: ACCEPT AT(12,20)VALIDAT
E(DIGIT)BEEP:LT :: IF LT>TA
THEN 420
430 DISPLAY AT(12,1):"ALTERN
ATING LEFT/RIGHT": :"MARGIN?
 (for pages to be":"later re
produced on both":"sides) (Y
/N) N"
440 ACCEPT AT(16,14)VALIDATE
("YN")SIZE(-1):A$
450 LSP=12 :: DISPLAY AT(10,
1):" ":" ":"LINES PER PAGE?
60":" ":" ":" ":" " :: ACCEP
T AT(12,17)VALIDATE(DIGIT)SI
ZE(-3):LP :: IF LP<70 THEN 4
90
460 DISPLAY AT(12,1):"LINE S
PACING - 72 INCH" :: DISPLAY
 AT(11,16):"__" :: ACCEPT AT
(10,16)VALIDATE(DIGIT)BEEP:L
SP
470 IF LP/(INT(72/LSP))>11.5
 THEN DISPLAY AT(20,1):"WON'
T FIT!" :: GOTO 450
480 PRINT #1:CHR$(27);"A";CH
R$(LSP);
490 RM=TA-LT
500 DISPLAY AT(12,1)ERASE AL
L:STR$(NC)&" columns of":STR
$(CW)&"-character width":"le
ft margin of "&STR$(LT)&" sp
aces"
510 DISPLAY AT(15,1):STR$(LP
)&" lines per page":"with "&
STR$(LSP)&"/72 line spacing"
520 DISPLAY AT(17,1):STR$(CS
)&" spaces between columns":
"right margin of "&STR$(RM)&
" spaces": :"OK? (Y/N) Y"
530 ACCEPT AT(20,11)VALIDATE
("YN")SIZE(-1)BEEP:OK$ :: IF
 OK$="N" THEN 240
540 DISPLAY AT(12,1)ERASE AL
L:"PAUSE AT END OF PAGE? N"
:: ACCEPT AT(12,23)VALIDATE(
"YN")SIZE(-1):QQ$
550 DISPLAY AT(1,1)ERASE ALL
:"INPUT FILENAMES TO BE":"PR
INTED.":"PRESS ENTER WHEN DO
NE"
560 X=X+1 :: DISPLAY AT(X+3,
1):"FILENAME? DSK" :: ACCEPT
 AT(X+3,14)SIZE(-12)BEEP:F$(
X)
570 IF F$(X)="" THEN X=X-1 :
: GOTO 600 ELSE F$(X)="DSK"&
F$(X)
580 ON ERROR 590 :: OPEN #2:
F$(X):: CLOSE #2 :: GOTO 560
590 ON ERROR STOP :: CALL SO
UND(1000,110,0,-4,0):: DISPL
AY AT(20,1):"CANNOT OPEN "&F
$(X):: X=X-1 :: RETURN 560
600 SL=1
610 F=F+1 :: IF F>X THEN 700
 :: ON ERROR 620 :: OPEN #2:
F$(F),INPUT :: DISPLAY AT(22
,1):"READING ";F$(F):: ON ER
ROR STOP :: GOTO 630
620 CALL SOUND(1000,110,0,-4
,0):: DISPLAY AT(20,1):"COUL
D NOT OPEN "&F$(F):: STOP
630 FOR IP=SL TO LP*NC :: LI
NPUT #2:M$(IP):: IF LEN(M$(I
P))=0 THEN 670 :: IF NC>1 AN
D POS(M$(IP),CHR$(13),1)<>0
THEN M$(IP)=SEG$(M$(IP),1,LE
N(M$(IP))-1)
640 IF ASC(M$(IP))>126 OR AS
C(M$(IP))<32 THEN IP=IP-1 ::
 GOTO 680
650 IF LEN(M$(IP))<=CW THEN
670 :: M$(IP)=SEG$(M$(IP),1,
CW):: CALL SOUND(1000,110,0,
-4,0):: DISPLAY AT(12,1):M$(
IP);" OVER";CW;"CHARACTERS":
"TRUNCATED TO ";T$:"OK?"
660 CALL KEY(3,K,S):: IF S=0
 THEN 660 ELSE IF K<>89 THEN
 STOP
670 M$(IP)=M$(IP)&RPT$(" ",C
W-LEN(M$(IP)))
680 IF EOF(2)=1 THEN CLOSE #
2 :: SL=IP+1 :: GOTO 610
690 NEXT IP :: IF EOF(2)=1 T
HEN CLOSE #2 :: GOTO 720 ELS
E GOTO 720
700 ON ERROR 710 :: FLAG=1 :
: FOR J=IP+1 TO NC*LP :: M$(
J)="" :: NEXT J :: GOTO 720
710 STOP
720 PP=PP+1 :: IF PP/2=INT(P
P/2)AND A$="Y" THEN LT$=RPT$
(" ",RM)ELSE LT$=RPT$(" ",LT
)
730 FOR J=1 TO LP :: ON NC G
OSUB 750,760,770,780,790 ::
NEXT J :: PRINT #1:CHR$(12):
: SL=1 :: IF F>X THEN STOP E
LSE IF QQ$="N" THEN 630
740 DISPLAY AT(24,1)BEEP:"PR
ESS ANY KEY TO CONTINUE" ::
CALL KEY(0,K,S):: IF S=0 THE
N 740 ELSE DISPLAY AT(24,1):
"" :: GOTO 630
750 PRINT #1:LT$&M$(J)&CHR$(
10):: RETURN
760 PRINT #1:LT$&M$(J)&S$&M$
(J+LP)&CHR$(10):: RETURN
770 PRINT #1:LT$&M$(J)&S$&M$
(J+LP)&S$&M$(J+LP*2)&CHR$(10
):: RETURN
780 PRINT #1:LT$&M$(J)&S$&M$
(J+LP)&S$&M$(J+LP*2)&S$&M$(J
+LP*3)&CHR$(10):: RETURN
790 PRINT #1:LT$&M$(J)&S$&M$
(J+LP)&S$&M$(J+LP*2)&S$&M$(J
+LP*3)&S$&M$(J+LP*4)&CHR$(10
):: RETURN
```

This is an improved version of the math program in Tips #36.

```
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
```

Here's an idea for an unusual title screen -

```
100 CALL CLEAR :: FOR SET=1
TO 8 :: CALL COLOR(SET,1,1):
: NEXT SET :: CALL CHAR(100,
"0",101,"0")
110 X$(0)="4043241818244202"
 :: X$(1)="4021261818648402"
 :: X$(2)="2020131C38C80404"
 :: X$(3)="1010101FF8080808"
 :: X$(4)="081010907E111020"
120 X$(5)="080808F81F101010"
 :: X$(6)="0404C8381C132020"
 :: X$(7)="0284641818262140"
130 A$=RPT$(CHR$(100)&CHR$(1
01),13):: FOR R=1 TO 24 :: C
=C+1+(C=2)*2 :: DISPLAY AT(R
,C):A$ :: NEXT R
140 CALL VCHAR(1,29,1,168)
150 CALL SCREEN(2):: CALL CO
LOR(9,5,16):: FOR S=1 TO 8 :
: CALL COLOR(S,16,2):: NEXT
S
160 DISPLAY AT(5,5):" TIGERC
UB SOFTWARE ";:: DISPLAY AT(
8,6):" SQUIRMY SCREEN ";
170 FOR J=0 TO 7 :: CALL CHA
R(100,X$(J)):: CALL CHAR(101
,X$(7-J)):: NEXT J
180 CALL KEY(0,K,S):: IF S=0
 THEN 170
```

MEMORY FULL

Jim Peterson

## PUTTING IT ALL TOGETHER #3

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 together to do what you want them to do! Key in this mini-program and run it to see what it does. Then read the explanation of each line and see how it does what it does.

```
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)::
Z=Z+1 :: DISPLAY AT(24,1):"P
LANES LOST";Z :: CALL DELSPR
ITE(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
```

This is not a finished program but a mini-program to demonstrate the use of sprites.

Line 110 first clears the screen. The CALL CHAR lists a hex code string of 32 characters. The first 16 of these will reidentify ASCII 100 into an airplane pointing down, the remaining 16 will reidentify the next ASCII, 101, to an airplane pointing upward. The screen is colored dark blue, the sprite magnification is set at 2 (single character in double size) and RANDOMIZE insures a different flight pattern each time.

In line 120, V and W and P are all set or reset to 0 (note that all can be included in one statement) because the program execution returns here from line 160 to restart after each crash. The J loop runs 7 times to put 7 sprites on screen, numbered 1 to 7, using ASCII 100 (the down-pointing plane), colored red (color code 7), at dotrow 1 (top of screen) and at a randomly selected dotrow between 1 and 250 (thus each game will be different), moving at a speed of 10 downward and 4 to the right.

The D loop creates a delay so that each sprite will drift downward before the next is created, so that no two will overlap and be later detected as a coincidence in line 150; also, so that more than 4 will not appear in a row and be blanked out. After these have been placed, sprite #11 with ASCII 101 (the upward pointing plane), colored white (16), is placed at dotrow 160, dotcolumn 128, without motion.

In the CALL KEY in line 130, the use of mode 3 insures that the ASCII of an upper case S or D will be returned even if the alpha lock happens to be up. W is a counter for the number of times that the keyboard is scanned (program execution passes through this line). When the count reaches 150, a jump to line 170 places a second plane on the screen, and at 300 a jump to line 180 adds the third plane.

Otherwise, if K=68 (D, the right arrow, is pressed), the speed of the CALL motion in line 140 is increased by 2. If this velocity is increased beyond 127 the program will crash, but if the relational expression (V>125) is true it will have a relational value of -1, and V=127+2+(-1)*2 will still be 127. If K=83 (S, the left arrow), the speed is decreased by 2, and the same formula insures that the velocity will not go below -127.

In line 140, P is the counter for the number of planes in the formation, initially set at 0 for one plane in line 120, increased to 1 and 2 in lines 170 and 180. This determines whether the CALL MOTION will change the speed (if any key was pressed) of 1, 2 or 3 sprites. The CALL MOTION has a 0 row velocity to keep the plane at the bottom of the screen, and a column velocity determined by the last keypress. Since all sprites are controlled in a single CALL MOTION, the speed change is simultaneous and they stay in formation.

In line 150, the CALL COINC checks whether any sprite is overlapping any other, even slightly. Since the red planes were all placed on the screen separate from each other and all traveling in the same direction and speed, they will never touch so any coincidence must be between a white plane and a red one. If there is no coincidence, A will equal 0 and program execution goes back to the CALL KEY.

If A=-1 there is a coincidence and line 160 creates a sound (I wish the TI was capable of a good bang!), the counter of crashes is incremented by one and displayed, all sprites are deleted, and execution goes back to line 120 to reset variables to 0 and place a new random formation on the screen.

When program execution jumps to line 170 from 130 after 150 key scans, the P counter for number of planes is changed to 1. CALL POSITION finds the dotrow R (not needed but must be included) and dotcolumn C currently occupied by the white plane. CALL SPRITE uses that value to place a second plane, sprite #12, 40 dotcolumns to the left.

If the original plane is less than 40 dotcolumns from the left of the screen, this would cause a crash because dotcolumn cannot be a negative number. The use of relational values again solves this problem. Suppose that sprite #11 is at dotcolumn 10. 10-40-(-1)*256 will place sprite #12 at dotcolumn 226 which will be 40 dotcolumns to the left of #11 when wraped around.

Similarly, after 300 keyscans, line 180 puts a third sprite-plane 40 dot columns to the right of the first one.

This program could be modified in many ways. The number of red planes can be changed in the J loop in line 120 - if more are added, the D loop might also need adjustment to insure that none will wrap around before all are placed.

The rate of speed-up can be adjusted by changing the 2 in line 130 to some other value, even a decimal value such as 2.5

If you would rather use a joystick than the keyboard, change line 130 to -

```
130 CALL JOYST(1,X,Y):: K=X*
10+Y :: W=W+1 :: IF W=150 TH
EN 170 ELSE IF W=300 THEN 18
0 ELSE IF K=40 THEN V=V+2+(V
>125)*2 ELSE IF K=-40 THEN V
=V-2-(V<-125)*2
```

By using variable names rather than values for the J loop in line 120, or for the velocity in line 130, you could offer options of difficulty at the start of the program. When doing any program ming with sprites in motion, it is always necessary to do a good deal of on-screen experimentation and program modification to get the desired results.

STRINGING AND UNSTRINGING

by Jim Peterson

The following program will give the plural form for most words. I will leave it to you to improve on it, and to teach the computer the correct plural form of PANTS, TOOTH, MOUSE, FUNGUS, DATA, and the other inconsistencies of the English language.

```
100 INPUT W$
110 L=LEN(W$)
120 Z$=SEG$(W$,L,1)
130 Y$=SEG$(W$,L-1,2)
140 ON POS("EFHNSXYZ",Z$,1)+
1 GOTO 320,150,180,210,240,2
70,270,290,270
150 IF SEG$(W$,L-2,2)<>"IF"
THEN 320
160 PL$=SEG$(W$,1,L-2)&"VES"
170 GOTO 330
180 IF (Y$<>"AF")*(Y$<>"LF")
*(Y$<>"RF")*(W$<>"HOOF")THEN
 320
190 PL$=SEG$(W$,1,L-1)&"VES"
200 GOTO 330
210 IF (Y$<>"CH")*(Y$<>"SH")
THEN 320
220 PL$=W$&"ES"
230 GOTO 330
240 IF SEG$(W$,L-2,3)<>"MAN"
 THEN 320
250 PL$=SEG$(W$,1,L-3)&"MEN"
260 GOTO 330
270 PL$=W$&"ES"
280 GOTO 330
290 IF (Y$="AY")+(Y$="EY")+(
Y$="OY")+(Y$="UY")THEN 320
300 PL$=SEG$(W$,1,L-1)&"IES"
310 GOTO 330
320 PL$=W$&"S"
330 PRINT PL$
340 GOTO 100
```

This program is also a good example of four of the Basic statements which are used to manipulate strings.

Remember that a string is a character, a group of characters, a word, a sentence, even a punctuation mark or a blank space or one or more numeric digits, which does not represent a numeric quantity. And a string variable name must end in a dollar sign.

Line 100 asks you to input a string. Line 110 contains the first of our string manipulators, LEN. All it does is to count the number of characters in the string, including blank spaces. Try it - type PRINT LEN("THIS IS A TEST") and Enter. So, L equals the number of characters in the string you input.

Line 120 introduces the next string manipulator, SEG$. That stands for SEGMENT, and what it does is to pick a segment out of the string, starting at the position you specify and continuing for as many characters as you specify. So, SEG$(W$,L,1) selects the portion of W$ beginning at L and continuing for 1 character. Since we have just defined L as being the number of characters in the string, we are starting with the position of the last character and it would do no good to specify more than one character. So, Z$ is the last character of W$. Try it - type PRINT SEG$("TEST",LEN("TEST"),1).

Similarly, line 130 defines Y$ as being the segment of W$ starting with the character in the position of the length of the string, minus 1, and continuing for 2 characters - in other words, the last 2 characters of the string. Try that too - PRINT SEG$("TEST",LEN("TEST")-1,2).

Line 140 introduces the manipulator POS, which means POSITION. It tells the computer to find the first occurrence, in the first string, of the character or characters in the second string, starting the search at the position specified. Try it - type PRINT POS("TEST","T",1). The answer is 1 because you told the computer to start searching for T, starting at the first character of "TEST", and "T" is the first character of "TEST". Try PRINT POS("TEST","T",2). Now the answer is 4, because you asked for the search to start at the second character, so the first "T" it found was the 4th character of "TEST". Finally, try PRINT POS("TEST","X",1). The answer is 0 because no "X" was found in "TEST".

Line 140 is a bit difficult to understand, but it shows one of the best uses of POS. It asks for the first occurence, starting with the first character of the string "EFHNSXYZ", of the string Z$ - which is the last character of the input string, remember? So, if W$ is a word ending in "E", Z$="E", and the POS of "E" in "EFHNSXYZ" is 1. If W$ ends in "Z" and Z$="Z", the POS of Z$ in "EFHNSXYZ" is 8 - get that? And if W$ ends in "W", Z$="W", and the position found by POS is 0 because it didn't find a W - remember trying that in the last paragraph.

Now, line 140 uses the value found by POS to GOTO the appropriate line number to continue the program. The trouble is, an ON _ GOTO must be able to read a consecutive series of values starting with 1, and if POS gives it a 0, the program will CRASH! That's why the +1 in line 140. If W$ does not end in any of the letters "EFHNSXYZ" then POS=0, +1=1, ON 1 GOTO 320.

So let's go to 320 and look at our last string manipulator, the ampersand, "&", "and sign", or in computerese, "concatenation". All it does is to join two strings into one. PL$ is our variable name for the plural form of W$, and in this case it consists of W$ with an "S" tacked onto the end.

If W$ ends in "E", POS=1, +1=2, ON 2 GOTO 150. In line 150, the <> when dealing with strings means "other than" or "is not", and the line means "if the segment of W$ consisting of 2 characters starting with the 2nd character before the last character, is not "IF" then go to 320." In other words, if W$ is not "WIFE" or "KNIFE" or some such, just put an "S" after the word, in 320.

Otherwise, in line 160, PL$ (the plural) consists of the segment of W$ starting with the first character and containing the number of characters equal to the length of W$ minus 2, with "VES" tacked on. So, "WIFE" becomes "WIVES", etc. - yes, I know the plural of "FIFE" is not "FIVES", but....!

If W$ ends in "F", POS=2, +1=3, on 3 GOTO 180. Remember that Y$ is the last two characters of W$. The coding here had best be the subject of another article, but just read those asterisks as "and" - if Y$ is not "AF" (as in LOAF) and Y$ is not "LF" (as in CALF) and Y$ is not "RF" (as in SCARF) and W$ is not "HOOF" (but it could be ROOF!) then go to 320 to tack an "S" on the end, otherwise drop through to 190 to pick out the segment from the 1st character to the next-to-last and add "VES" to it.

Similarly, in lines 210-220, if words ending in "H" to do not end in "CH" or "SH" they take the "S" ending, otherwise "ES". In lines 240-250, if words ending in "N" do not end in "MAN" they take "S", otherwise take the segment from the 1st character to the 4th from the end and add "MEN". Words ending in "S", "X" or "Z" are referred to line 270 to add the "ES" ending, and lines 290-300 figure out that words ending in "Y" preceded by a vowel take the "S" ending, otherwise knock off the "Y" and add "IES".

I hope that I haven't overlooked some other rule of plural endings here - but anyway, I'm trying to teach you how to program, not how to spell! o


*"These educational packages sure have learned me a lot, Dad!"*

# WORD PROCESSING

SOME FORMATTER TRICKS
OR
"HOW TO AVOID READING THE MANUAL"

by Jean Wilcox, SunCoast 99ers

Some time back, Mr. Molander very kindly gave me the information I needed to bring a Basic or X-Basic program into the word processor, for which many thanks. The way to handle this is to list your program in a file format that can be read by TI Writer, i.e. LIST "DSKn.filename". Then, using the Text Editor, the program can be loaded into your text buffer, either first or following a given line number of existing text. (A few rude souls nearby suggested it might be to my advantage to read the manual, and I fully intend to. Not today, of course, but sometime very soon.). I did encounter a couple of small difficulties and made some gigantic messes before I figured out how to handle the situation. For one thing, if you are planning to print through the Formatter, as I generally do, the programs you draw up out of file will do some really strange things if you forget to use the .TL command before printing.

Looking at the screen, I assumed that what I saw was what I would get. Not so. It's easy to forget that the exponent sign for maths is also the Required Space sign for text; the "@" key is handy to use as a variable name, but causes the printer to double-strike; the ampersand, the symbol for concatenation of strings, is used to underscore in word processing. A large proportion of programs will use either the exponent or the ampersand, or both, so it's a good idea to .TL these before you attempt to print them out, unless you want to duplicate my goofs.

Here's something else that TI never told us about TI Writer. Every time I think I have found all the things that will give trouble printing through the Formatter, I find another one. The newest character to add to the list is the asterisk. Assuming it is to be followed by a space or letter, (or group of spaces or letters), you can print asterisks all day long. You won't have any problems with A*B, 5*C, or HELLO****STRANGER. But you will be confounded if you attempt to print one followed directly by a number. As an example, if you need "A*123", what you will get will be "A*3".

Part of my wasted time was spent trying to find out why the miserable thing was doing what it was doing.. I finally located the one place in the manual where an asterisk is mentioned as having a function, rather than as just another character to be printed. It's on pages 111-113, listed under Alternate Input, Mail Merge Option. (How many of you send out form letters?) If you feel like getting really technical about the thing, read the part about Define Prompt, too. It's on the same pages. Even after finding this I still didn't immediately associate the Mail Merge with the problem I was having, since the command necessary to carry out the job in a form letter is described as "*n*", an asterisk sandwich with a number in the middle. So I just quit worrying about the Why of the situation and started working on the What To Do About it.

I must not be too swift because it took another hour of typing all sorts of strange stuff containing asterisks in odd configurations to realize that I could always get what I wanted if a space immediately follows the asterisk. In a formula it will look lop-sided, (since the space is printed, too), so, as a dedicated neat freak, I'm typing a space before and after it. It's just as easy to remember that as the other, and the results look as though it were what you had intended all along.

A day or two later, Irene called to say she had information on this from the head guru, Guy-Stefan Romano. He explained that it was indeed the Mail Merge Option that was the culprit. It seems that when TI Writer encounters as *, it looks for one or two numbers for the Value File needed for the form letter, then proceeds to strip out the asterisk and the numbers following it. His solution was to print two asterisks, followed by two dummy numbers, one space, and then the figure you want printed. This works well, but you're going all the way around your elbow to get your thumb. Why not just put a space fore and aft and carry on?

Another thing that caused me trouble also springs from my neatness fetish. When the various program lines are formatted they get drawn all up in a knot.. if there's room on a line of type to print something, it will get printed, whether you want it there or not. The obvious answer to this, of course, is to add a carriage return at the end of each program line. "Enter" is supposed to do this, but for me it undependable. If the line is short it might work, then again it might not. I found that if I enter the <cr> symbol by using CTRL 8 I was in business. This causes a blank line to be inserted after each existing line, but they are easily deleted. There are probably dozens of ways, all shorter, simpler, and more efficient, to accomplish these things, but they work for me. So until I get around to reading the book, this is the way I'm going to do it.

(NOTE: The <cr> does not appear when you have previously deleted it on a line, inserted a line, or deleted lines after the cursor. Another way to get the <cr> on the line is CTRL M, but you will still get the extra line.)

o

---

TI Writer FILE PRINTER

by George Steffen, LA 99'ERs

Reprinted from LA Topics Vol 4 No.6

In the last month's newsletter I first noticed the wish for a simple program to print TI Writer files. Here is one that I wrote last year for Chick DeMarti before he acquired a TI Writer. Unfortunately, because the INPUT statement assumes that a comma signals the end of a variable, we have to use Extended Basic and the LINPUT statement. Lines 150 and 160 get the names and open the input and output files. The entire printing portion of the program is in lines 170 and 180. The second statement in line 170 will eliminate the Tab pointers which are kept along with the actual text in TI Writer files.

```
100 REM FILEPRINT
110 REM TI EXTENDED BASIC
120 REM GEORGE F. STEFFEN, LA 99ER CG
130 REM VERSION 1.0, 5/10/84
140 READ P$,T$ :: DISPLAY AT (3,1):ERASE
    ALL:"THIS PROGRAM WILL PRINT TIW FOR
    MATTED FILES OR PROGRAMS LISTED TO DI
    SK ON"
150 DISPLAY AT(6,1):"YOUR PRINTER.": :"YO
    UR PRINTER NAME?":P$ :: ACCEPT AT(9,1
    )SIZE(-28)BEEP:P$ :: OPEN #2:P$,OUTPU
    T
160 DISPLAY AT(11,1):"FILE TO BE PRINTED?
    ":T$ :: ACCEPT AT(12,1)SIZE(-15)BEEP:
    T$ :: OPEN #1:T$,DISPLAY ,INPUT
170 IF EOF(1) THEN GOTO 190 ELSE LINPUT #
    1:L$ :: IF LEN(L$)THEN IF ASC(L$)<128
    THEN PRINT #2:L$
180 GOTO 170
190 CLOSE #1 :: DISPLAY AT(14,1):"DO ANOT
    HER (Y/N)? Y" :: ACCEPT AT(14,28)SIZE
    (-1)VALIDATE("Y/N":L$ :: IF L$="Y" TH
    EN 160
200 CLOSE #2 :: STOP
210 DATA PIO,DSK1.TEXT
```

o

PRETTY PLEASE, PINCH MY DEAR

AUNT SALLY RUDELY!

by Jim Peterson

My apologies to dear old Sal. That mnemonic device is usually given as just "My Dear Aunt Sally", but I expanded it a bit. It is intended to remind you of the sequence in which your computer solves an equation, which is -

(P)arentheses
(P)owers (exponentiation)
(P)refixes (plus and minus)
(M)ultiplication
(D)ivision
(A)ddition
(S)ubtraction
(R)elational operations

So what? Well, if one of your program lines isn't giving you the expected results, it may well be that you forgot to pinch Saly properly!

The computer goes through the line from left to right 5 times (I don't know if it really does, but that is the easiest way to explain it!) The first time through, it looks for a left hand parenthesis. If it finds one, it stops at the first right hand parenthesis. If it finds one but not the other, it CRASHES! When it finds a right parenthesis, it backs up leftward until it comes to the closest left hand parenthesis. It solves everything between those two parentheses, step by step in accordance with the following priorities, and then erases those two. Then it goes through the same routine again until it finds no more parentheses.

Need a "for instance"? OK -

```
X=((10*2)-6)+(8/4)
X=((20)-6)+(8/4)
X=(20-6)+(8/4)
X=(14)+(8/4)
X=14+(8/4)
X=14+(2)
X=14+2
X=16
```

Next it goes through the equation looking for the caret sign. That is the little ^ that tells it to multiply the preceding number by itself as many times as the following number. Example -

4^2 means 4 times 4
6^3 means 6 times 6 times 6

Then, the prefixes. That just means that, for instance, if removing the parentheses from -(-6) has left you with --6, it becomes a +6, of course. I suppose that ABS and SGN are also worked here.

Now, multiplication and division. These are both done in one pass through because it doesn't make any difference which is done first. 10*2/4 is the same as 2/4.

Next, addition and subtraction, also in one pass because 10+4-2 is the same as 4-2+10.

Finally, the relational operations, which had best be the subject of a separate article. And finally finally the string concatenations, but let's keep old Sal out of those.

Note that everything between a pair of parentheses is worked as a separate equation, step by step in the above sequence, before the parentheses are erased.

So, why should you need to worry about all this? Well -

```
10*4-2=38
10*(4-2)=20
10*4^3=640
(10*4)^3=64000
((10*4)^3=....SYNTAX ERROR!
```

Makes a difference, doesn't it?

The important things to remember are -

If you want to add two numbers together before you multiply or divide their sum, put them in parentheses (2+3)*4.

If you want to subtract one number from another before you multiply or divide the result, put them in parentheses (10-4)/2.

If you want to add, subtract, multiply or divide numbers before you increase them by any power, put them in parentheses (10*4+8)^3.

If you keep Sally in mind, you will have fewer bugs in your programs! o

---

USING CONTROL 2 WITH THE TERMINAL
EMULATOR COMMAND MODULE

To dump a screen to an output device, such as, disk,printer or other.

First, make sure host is in a pause state. Second, press CTRL 2. At this time the TEII emulator asks that you type in the output device of you choice.

Valid inputs are.

```
DSK1.filename
DSK2.filename
DSK3.filename
CS1
CS2
TP
PIO
RS232/1
RS232/2
```

After typing the device name of your choice, press enter. At this time the TEII emulator will dump the contents of the screen to the device you specified.

For those of you dumping to disk.

After you are thru dumping all the screens to the same file, you must close the file, or you will loose it all.

There's two ways to close a file.

1)PRESS CTRL 2 than 2 for NO
2)CTRL 0 (WHEN HOST HAS DICONNECTED)

Keep in mind, that once you close a file, you can not reopen it.

To dump more info you must open a new file...

Retrieving data from Disk

All data dumped to disk, are saved in a DIS/VAR80 file.

There's three methods of recalling data from disk.

1) use of TI WRITER
2) use of E/A MODULE
3) use of the program on page 20 in the TEII manual.

NOTE

Line 250 should read.

250 GOTO 220

o

# REGIONAL GROUP NEWS

---

GLEBE Regional Group.
10th December 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.

Next meeting:

SATURDAY 12th December 1987 at 2pm.

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.

Christmas meeting will be SUNDAY, 6th December, 1987 at the home of Ebel Cummings, 48 Manoa Rd, Budgewoi. This will be a poolside B-B-Q, bring your own meat and swimmers.

Contact Russell Welham (043 92 4000)

---

CARLINGFORD Regional Group.

The next Carlingford Regional Group Commencing Time 7.30 pm

For further information contact Chris Buttner.

Regular meetings are third wednesday of each month.

---

ILLAWARRA Regional Group.

Next meeting 14/12/87 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.

Regular meetings are third or fourth Thursday of the month.

---

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.

---

All TIsHUG Regional groups are invited to submit items for this department. Send details to EDITOR.

---

GLEBE Regional Group.
7th January 1988, 8pm, 43 Boyce St, Glebe. Contact Mike Slattery, 692 0559.

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

The activities at these regional meetings are rather informal include looking at new hardware, hardware repairs, looking at new software and having a general chat; often not finishing till 3am!

---


ADVENTURE HINTS

Return to Pirate's Isle

Chapter 5

This is the final chapter of this great adventure. You will be able to get out but not get a perfect score. The challenge now is how to score 100.

Wash mask, wear mask, hold breath, swim down, swim east, swim up, go dock, get doll, remove mask, wash mask, wear mask, look dock, go down, hold breath, swim down, look pilings, swim west, swim up, open oyster, use nail, go dock, go boat, go sea, hold breath, swim down, swim west, swim opening, remove mask, wash mask, clean mask, breath, wear mask, go boat, drop pin, drop doll, drop chest, drop book, go pool, remove mask, wash mask, wear mask, go boat, drop pearl, score ??????.

The pirate of this column wishes you all a merry Christmas and a happy New Year.