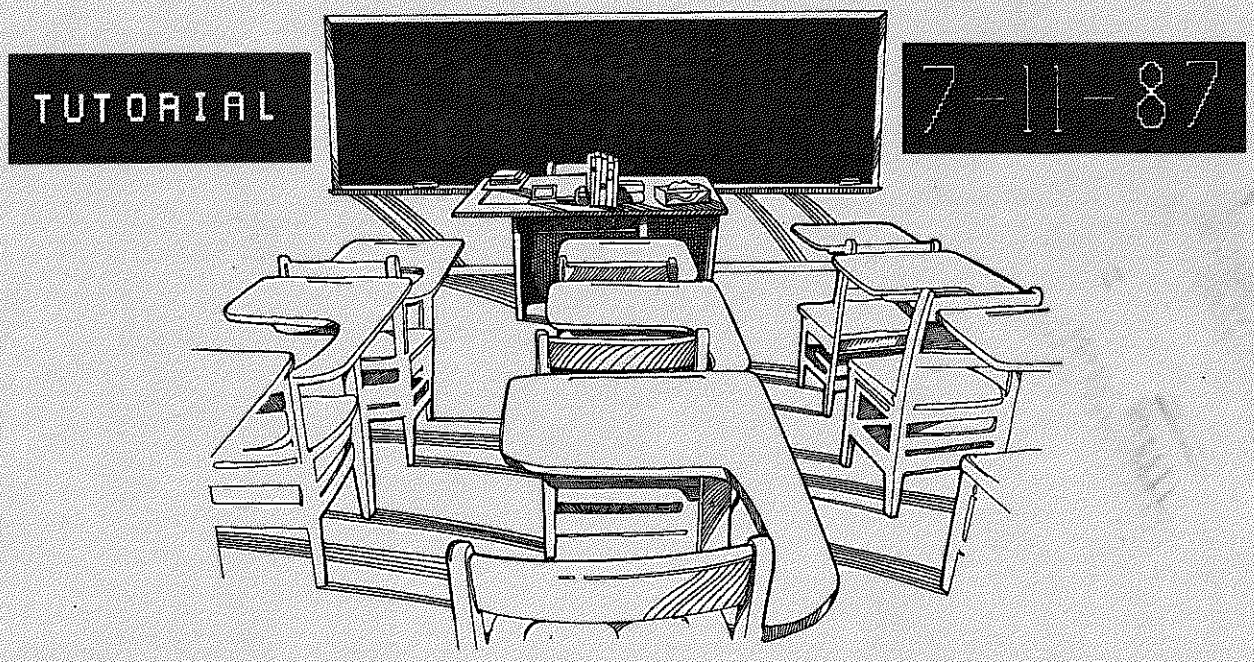
ISHUG NEWS DIGEST

Focusing on the TI-99/4A Home Computer

Volume 6, Number 10 **November 1987**

Registered by Australia Post - Publication No. NBH5933

DO NOT BE LEFT OUT



PUT YOURSELF IN THE PICTURE

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

TISHUG NEWS DIGEST

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

NOVEMBER 1987

Correspondence to:

PO Box 214
REDFERN NSW 2016

Texpac BBS: Tel.: (02)319.1009

COMMITTEE MEMBERS:

Co-Ordinator:
Chris Buttner..Tel.(02)8717753
Secretary:
Terry Phillips.Tel.(02)7976313
Treasurer:
Bert Thomas....Tel.(047)541535
Publications:
Bob Montgomery.Tel.(042)286463
Sysop:
Ross Mudie.....Tel.(02)4562122
Merchandising:
Cyril Bohlsen..Tel.(02)6395847
Technical:
John Paine.....Tel.(02)6256318
Librarian:
Terry Phillips.Tel.(02)7976313

REGIONAL COMMITTEE MEMBERS:

Glebe:
Mike Slattery..Tel.(02)6920559
Penrith:
John Paine.....Tel.(02)6256318
Central Coast:
Russell Welham.Tel.(043)924000
Liverpool:
Arto Heinoe....Tel(046)6038956
Illawarra:
Rolf Schreiber.Tel.(042)842980
Bankstown:
Peter Pederson.Tel.(02)7722396
Carlingford:
Chris Buttner..Tel.(02)8717753
Sutherland:
Peter Young....Tel.(02)5288775
Manly Warringah:
Dennis Norman..Tel.(02)4523920
Coffs Harbour:
Keir Wells.....Tel.(066)551487

MEMBERSHIP AND SUBSCRIPTIONS:

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

GROUP GENERAL MEETING:

This month there is a tutorial at Shirley House, Church Street Burwood. Starts 9:00am

COMMITTEE MEETINGS:

Sometime during the day of the meeting. Starting at ??:?? pm.

TIsHUG NEWS DIGEST ISSN 0819-1984

CONTENTS PAGE

This month's Tutorial day really looks into how to use assembly language to advantage. Many TIsHUG members have machine language programs running out of Extended Basic. George Meldrum has converted many of those that could normally only be used with the E/A module and a disk drive. George will explain and show how it is done. Bring along this magazine as the article on Implanting M/C will help in following the talk.

Ross Mudie will start the day of with some beginner's assembly. See the article on the day's program on page 12.

Again Jim Peterson has many articles in the magazine and these are included because they relate more readily to those who haven't yet moved on to assembly. That also includes the Editor.

Next meeting is the last for the year and also the Christmas party. Plan to be there.

Bob Montgomery

CO-ORDINATOR'S REPORT

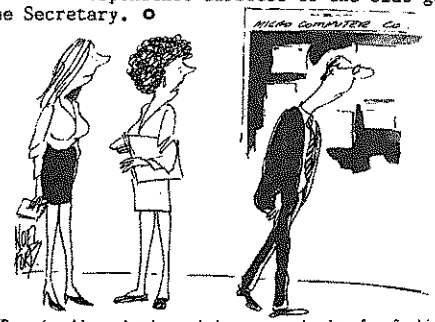
.... Chris Buttner

Firstly, I am happy to be able to tell you we have successfully incorporated as from 22 September. Our full name is TIsHUG (Australia) Limited, a public company limited by guarantee. In the coming weeks the committee members and directors will have the task of formally winding up the activities of TISHUG following which final tax returns will be lodged. The company will take over all the functions formerly provided to you. The final cost to the club is likely to be about $290.00 (very cheap). We could not have achieved this without the support and assistance so generously given Brian Graham. On behalf of all of you I extend thanks to him.

Many of you are aware that Arto Heino has written an excellent graphics/text program which is probably the closest we will come to a Desk Top Publishing program to the TI99/4A. I want to tell you what unfortunately has happened to Arto and his program. To the best of my knowledge, the root of the problem does not lie in our own club but the events bear repeating. Picasso Publisher is NOT a fairware program. Much of the work was done at a time when Arto was unemployed. He sells the program to quite understandably help his financial situation. An interstate TI User acquired a copy of the program from sources unknown. He then sent a copy of the program to a fellow TI User in USA. You might be tempted to say that so far no great harm had been done however the interstate user really took too much upon himself and in sending the program stated that it was freeware (shareware). Arto hopes to market his program in the USA through a software publishing house and I hope for his sake that PICASSO never got onto a bulletin board over there. From what I've heard, such things breed faster that rabbits (due no doubt to the large number of boards, low charges, and nationwide access). The point of this tale is quite simple. If you send programs to others (local, interstate, overseas or wherever) make sure of your facts regarding the program. If it is shareware/fairware say so. If you are unsure of its status, you probably shouldn't send it until you check further and finally, if it has the circled "C" (copyright logo) on it anywhere at all don't send it. So many of the useful programs we now have for the computer were published by third parties. If you follow these simple guidelines, you will not be knowingly harming the interests of the author. After all it is not very smart to bite the hand that feeds.

When your committee started out this year we had in mind a number of projects which we believed would be a major benefit to many of you. Much of the impetus for these projects was sparked by your requests. Happily, most of what was asked for has now been delivered. If there are still unsatisfied demands, please tell us so we can get the ball rolling. You can always write to the Secretary, or speak to one of the committee either at a regular monthy meeting or regional meeting.

A final plea: If you are writing to the club or one of the office bearers, PLEASE use our NEW address; (6 months old) of P.O. Box 214, Redfern, 2016. This is particularly important now that we are incorporated to have all correspondence directed to the club go direct to the Secretary. o



'Poor Arnold — what he needs is an expansion interface for his brain!'

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 *

MEETING 7/11/87 - FULL DAY

The November meeting, a full day tutorial workshop, with lunch available, will be held in the main Woodstock Building.

Reservations have been made for the ground floor Ball Room and Group Room 1 (opposite the tea room) from 9am to 5pm.

Like all full day events in the past, this promises to be a fun day, where many things will be explained and the latest bits and pieces demonstrated, so be there if you can.

For those whom haven't attended a workshop at Woodstock previously, the surrounding grounds are very attractive with spacious grass areas and playground equipment for the children and it is only a short walk from Burwood railway station.

Given a fine day I am sure that this will be the highlight of events this year

MORE RUMORS FROM THE STATES

A programmer on the Eastern Coast is talking about coming out with a flight sumulator program.

Millers Graphics has supposedy brought out a new Extended Basic. My "spy" is attempting to obtain further information.

4A-DOS software now received and by all reports very good.

Still no luck in running the TI-IBM connection due to a non IBM formatted disk being available at the meeting. More on this shortly.

A few extracts received from the ROM newsletter (Orange County UG) suggest this is a very good magazine. I have written to ROM to start an exchange program.

A hearty big TIsHUG welcome to the following 2 new members who joined us during the past month.

J P Brooks - Croydon Park
Jason Scott - Peakhurst

I hope your stay with the group is rewarding and that you can both attend meetings on a regular basis to gain maximum benefit from your TI's.

During the past month 16 local and overseas group magazines have been received and added to the publications library. If you are not yet a member of the 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. Remember it is only $5 to join the publications library.

Now some news and requests from correspondence recently received.

Barrie Stevenson of Eastwood asks if it would be possible to compile a list of tape recorders that members have found to be compatable with the TI. Barrie's problem is that his National RQ309S has packed it in and it would be uneconomical to repair. He doesn't want to spend about $50 on a new machine to find it wont work with his TI. I would like to hear from members on this issue so that the list, as suggested by Barrie can be compiled. For a start I have used a National RQ2309 with very few problems for a number of years, but I believe that this model is no longer made or available.

Eric Whelan of Healesville asks does Console Writer work from a console, TV, XB, tape recorder set up. Eric the answer to this one is yes. Console Writer comes on a module and is a stand alone word processor. XB is not required to use it. Eric also asks if the TIsHUG Shop would import for him Wordwriter, a new cartridge based word processor recently advertised in Micropendium. The answer to this one, Eric is no unless a number of members also requested this software. It would be un-economical to import only one copy. This of course does not stop you importing it yourself, but remember to add about 50% to the price quoted in US dollars to arrive at the likely cost in Australian dollars.

Mark Beck of Creative Filing System fame writes to advise that Version 7.0 is now available and that he is working towards Version 8.0. Anyone with ideas or suggestions on how CFS may be improved is welcome to write to Mark at 8 Forrestridge Circle, Valdosta, GA, 31602 USA. I will be writing to him shortly to obtain a copy of Version 7.0.

Any ham radio addicts may be interested in this bit of news from John Johnson of the San Diego TI User Group. John says that a member in his group, John Moyta operates a ham rig in conjunction with his TI. His call letters are KF6SK and he operates on the 20 meter band around 14.075 to 14.100. So if your into ham radio try and contact John and get the latest info on the TI scene quickly.

A US based "mole" sends the following bits of information.

Fortran, offered by Tenex, appears to be quite acceptable.

Turbo Pascal, recently advertised in Micropendium has not yet been seen, so no comments.

A number of people have bought the Myarc 9640 but so far no MDOS necessary to get it to run. All are looking forward to getting their Myarcs to fly.

Any one buying a copy of Not Polyoptics flight simulator are warned that it operates in real time so at times it is about as interesting as watching paint dry. The dogfighting, balloon bashing and hangar bombing can be real fun though. An excellent manual is provided. (Has any member yet seen this?)

Asgard Software are releasing a new game patterned after Dungeons and Dragons, titled Legend. Not yet seen so no comment.

Well, that's all to report for this month. Remember if you have a piece of news or a query, then pass it on for publication. o

Hardware News

IMPORTANT EVENT AT NEXT GEN. MEETING

The new PEB card, another WORLD FIRST for the land of OZ, will make its debut at this coming TISHUG Meeting. This is the first true multi-function PE box card offering no less than 4 functions. There is the very advanced AT Disk controller with extra call commands, 32K Memory expansion, a PIO printer port, and two RS232 ports, the latter is also greatly enhanced with 19200 Baud, VIATEL, 1200/75 Baud, 50 baud RTTY, and MIDI speed interface, all available from console BASIC. This has been made possible by using the latest technology advances and a 256K EPROM for the DSR.

Also each of the 4 main functions are independent and the board will be offered in partly built-up form if needed. A more detailed report will be offered in the next TND so watch out for it.

TECHO TIME

.... John Paine

As some members may have noticed, I was unable to contribute any articles to the last edition of the NEWS DIGEST last month. This was because of a Catastrophic failure of my TI System. The cause of this failure could in turn lead to a 2000 word explaination on it's own. Suffice to say, I was repairing a faulty RAM DISK and was using an extender board in the PE Box and in a rare fit of madness, I accidently fed a 12 Volt DC signal into the inputs of the address buffer chips. Not content with destroying the buffers on the Ram Disk, this voltage decided to make an excursion down the PE Box bus and systematically destroy every buffer chip and any other chip it could find on the way.

The grand total came to 15 buffer chips, 5 PAL chips and one Disk Controller chip. I was able to source the RS232 and 32k PAL's from TI, in Sydney, but had to ring CorComp and order the PAL chips for the Disk Controller from the US. CorComp, and especially their President, responded magnificently and despatched the necessary chips poste haste. Unfortunately all CorComp's good work and excellent service was negated by lousy postal service between the US and Australia Post. The chips were sent airmail and took some 9 week's to arrive. Verification of the post mark was sufficient to indicate that CorComp did their part with the utmost speed.

I have have noticed in a number of journals that CorComp have been accused of being unresponsive to user problems and being generally unsupportive. I must state that the level of support and help from CorComp on the two occasions that I have had to call them has been nothing short of superb. I wish I could say that about other Hardware suppliers I have had the occassion to contact.

Now lets get down to something a little more on the technical side.

A number of members and others have purchased the MINI PE SYSTEM, which in MOST cases will run directly from the Expansion Port and use the internal Console Power supply. The TI internal supply will definately cope with this extra load if the following upgrade is made. The upgrade consists of replacing the four 1N4001 diodes with diodes of a higher current rating. If your Console is supplying power to an internal 32K Memory Expansion as well as a MINI SYSTEM, then the 1N4001 diodes will become quite distressed. The 1N4001-4 diodes are specified as 1 amp diodes at an ambient temperature of 25 degrees C. I have yet to see any piece of electronic equipment run at this temperature and the TI Console is no exception. Temperatures in and around the power board of the console can be as high as 65-75 degrees. (about the temperature of a good cup of coffee). At this sort of temperature, the diodes have to be derated and can only deliver about 600-700 mA.

The Console itself, requires about 1 Amp to run and the MINI PE SYSTEM when fully configured needs an extra 300-400 mA. What we then have is a recipe for disaster in the long term.

The 1N4001 diodes can be replaced with a number of various types with the 1N5401-4 series being the most commonly available. The existing diodes can be easily cut from the power board and the new diodes soldered directly onto the pads on the PC board. The only major requirement is to ensure that the new diodes are soldered in with the same polarity as the old diodes just removed.

This modification is only necessary if your console has 3 or 4 pins on the power connector at the rear of the console. If your Console only has two pins, then you have a later power board which will have a 5 Amp Bridge rectifier instead of discrete diodes.

When you remove the power board you will see 5 diodes (little black things) adjacent to the power switch. The only diodes that need to be replaced are CR1, CR2, CR3 and CR4. CR5 is only used to generate the -5 Volt power for the console and speech synthesizer if fitted. The 32k memory and/or the MINI SYSTEM do not use this voltage and the current capacity is more than enough.

One note of caution, however. The 1N5401-4 series of diodes are a lot thicker than the existing diodes therefore it may be necessary to stagger the heights of the diodes above the board.

The last article submitted to the magazine indicated that a number of modules would be released as kits for home construction. The first module was to be a Viatel terminal emulator for those members who still were using an unexpanded console with some form of standalone RS232 box. Since then, a number of other modules have been developed for the console only members and as of the time of publication of this magazine, a console based word processor which can save it's documents to both cassette, printer and disk will be available from the SHOP. The necessary parts have been ordered and are available now. The initial modules ordered will be supplied as fully built up units at no extra cost, unless specifically ordered as a kit.

The following list indicates the availability of the various developments and any special requirements that would be needed to run them.

1) Viatel 1. Needs RS232/1
2) Viatel 2. Needs RS232/2
3) CART-Writer . Needs cassette minimum.
4) Diagnostics (Special Purpose Module for testing expansion peripherals)

The boards that will be used are through hole plated and can be used for custom modules or backups of existing modules for club members. Backups will only be supplied to members who produce their original modules to me. I will not copy modules for any other purpose.

The cost of our locally developed modules will be around $20-$25 and backups of your modules will be around the same price. Please contact the shop for the correct pricing structure.

RAM DISK TIP

For those members who have Horizon or our local TI Solid State Disk Drives and are running the John Johnson Menu program V6.0 or later, here is a small tip that will allow you to run extended basic programs that are in Int/Var 254 format. The menu loader only recognises XB programs in program image format. To load I/V format simply write a one line XB loader and load onto ramdisk. Use the Menu load option to load your new loader instead. The following example may be of assistance.

```
FRED .....I/V 254 (will not load)

100 RUN "DSK*.FRED !where * represents RD Drive' #
```

Save this file as LOADFRED on to the ramdisk and then use the inbuilt loader to select and run "LOADFRED".

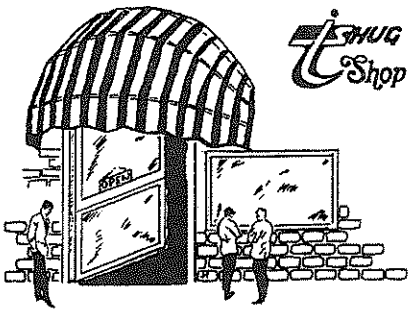
LOADFRED will execute and immediately load and run "FRED".

The same principle can be used to load and run Dis/Fix 80 file eg.

```
100 CALL INIT :: CALL LOAD("DSK*.FILENAME")
110 CALL LINK("STARTNAME")
```

Save this loader as LOADPROG and then use Menu Loader to execute.

o



This month looked like being a very lean time for the shop until two of the keenest members came to my rescue with some good ideas. The first was Arto Heino with an updated version of his Picasso Publisher (version 2.0) plus a couple of disks full of enhancements for his programme.

Bring or send your old original copy of Picasso to the shop for a replacement copy.

The first of these disks from Arto is nearly full of great graphic pictures and math generated contour displays, whereas the second disk contains 536 sector of large character sets. Both these disks will be on sale as club software along with the usual software supplid by Terry Phillips.

The second was our old friend John Paine with a list of modules the club will be making very soon. The first modules will be :-

(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 gear)
(e) EDITOR ASSEMBLER module

The cost of these modules will be about $25.00 each If you are interested in any of these modules please place your order early as they will mainly be made to order.

RAM DISK CARD PARTS :-

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

PRINTER BUFFER 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).....$ 15.00
(b) Spike Protectors.....$ 29.00
(c) T.I. Joystick handles.....$ .50

(d) Peter Schubert's mini-expansion unit
DS/DD Disk controller card.....$190.00
Mini-PE mother board (with one of either-
32K mem :: PIO :: RS232 port).....$ 85.00
Extra options on mother board
32K memory.....$ 50.00
PIO printer port.....$ 50.00
RS232 port.....$ 50.00
Finished painted box for Mini PE...$ 35.00

SECOND HAND ITEMS :-
(a) Grom Ports.....$ 12.00
(b) Ivory Console Cases.....$ 2.00

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

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

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

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

FOR SALE

Expansion Box.....	$200
RS232 T.I.....	$140
32K Memory.....	$50
Disk Controller.....	$100
Interface Card.....	With Box
2 Half Height Disk Drives DSDD.....	$150 Each.
Speech Synthesizer.....	$50.
Key Board Pre 1983. Grey.....	$55.
Cartridge Expander - Naverone.....	$25.

Applications:-	
Multiplan.....	$60.
TI-Writer + Spell Checker.....	$45.
Editor/Assembler.....	$60.
Modules:-	
Extended Basic.....	$70.
T.E. 2.....	$50.
Touch Typing Tutor.....	$15.
Mini Memory.....	$30.
Plato Interpretor + 9 Disks,Elect, Mag.,Optics.etc.	$50.
Statistics.....	$30.
Personal Record Keeping.....	$35.
Video Graphics.....	$20.
Disk Manager 1.....	$5.
Disk Manager 2.....	$30.
Munch Man.....	$10.
Tombstone City.....	$10.
Line by Line Assembler Tape.....	$10.
Electrical Engineering Library Disk	$15.
Graphing Package Disk.....	$15.
Data Base 99 Disk.....	$10.
TI Forth Disk.....	$10.
DM 1000 Disk.....	$10.

ALL OF THE ABOVE COME WITH APPROPIATE MANUALS,CADLES, etc.

Books:-

Intro. to Assembly Lang. for TI. - Molesworth.R.	$10.
Programming Basic for TI. - Pecham. H.	$5.
36 TI Programmes - Turner. L.	$5.
101 TI Programming Tips & Tricks. - Turner. L.	$5.
TI Graphics Programmes. - Turner. L.	$5.

ALL THE ABOVE PRICES ARE NEGOTIABLE
WITH IAN J. SMITH
4,BUCKINGHAM STREET,KINGAROY. Q4610.
Tele. 071 623075.

SOFTWARE COLUMN

NOVEMBER 1987

Elswhere in this issue you will read about the exciting software competition to be conducted with a fabulous first prize of $200 worth of shop stock, plus other minor awards. Now is the time to start on your entries, and remember they must be handed in by no later than the February, 1988 meeting date.

During the past month there has been no new software received, and this coupled with the fact that I will be on holidays commencing late September means that there will only be one disk and one tape released at the October meeting. But the disk and tape are good ones and I would be remiss if I did not thank George Meldrum for the work he has done in coverting some Memory Image files to run out of XB. I won't tell you what the contents are, mainly because I have not yet decided, but I am sure you will be impressed. As a plus on the disk version is a working copy of a program, Inverse Video Simulation, that appeared in the June issue of Micropendium. This is a real good load program, very different and works effectively.

Both disk and tape 1987/10 will required memory expansion.

Hopefully, in the next weeks some software will be received to enable a bigger issue at both the November and December meetings. I already have one in line for December with some excellent Christmas type music.

LATE NEWS: Just received a batch of disk from Jim Peterson with no time to look at before the publication deadline. Watch for more news on these in the next issue.

The above was last months offering which unfortunately arrived too late to meet the Editors deadline for publication.

By now, most, or those who at least got hold of the disk/tape, 1987/10 will know what was on it. For those who didn't these were the programs. As mentioned above all require XB and 32K expansion.

ANTEATER - a fun sort of game of the burrowing/find food type and watch out for the quick witted anteaters.

BLACKHOLE - a two player game where the object is to either blast your opponent or manouver him/her into the black hole, from which of course there is no escape.

CAT & MOUSE - the mouse (you) runs around the screen eating pieces of cheese that pop up here and there. The object of the game is to eat the cheese and avoid a couple of nasty cats that chase you around the screen. Easy at first but after a while look out.

CROSSFIRE - not a game that I am particularly fond of but no doubt many will be. The idea is to shoot various monsters or whatever they are as they run around the lanes. I always found it a bit slow in response to the joystick hence it would never go into my top 10.

GUARDIAN - a two player space shooting game with fast flowing graphics and good background scenarios. Again it starts out fairly easy but as the game progresses it becomes difficult to play.

MOON PATROL - a one or two player game of survival while driving a patrol car on the moon. There are some quite difficult screens in this game and because of this it is one of my current favourites.

TI RUNNER - an excellent strategy jumping game with dozens of different screens to master. Definitely not as easy as it looks as you will find out when you play.

STARFORCE - a fairly boring game of lining up the enemy in your sights and blasting with your laser. I am not very excited about this game at all.

VIDEO VEGAS - probably the best graphics available in a poker machine game for the TI. But after that what can you say? Probably drive you to the club to play the real thing. Boring!

Now this months offerings:

ON DISK:

DISK1987/11 - contents:

Financial assistance programs all designed to help you with your loans and mortgages plus a calories counter and a data base to keep track of coupons, although could be modified for just about anything. Titles are

AMORTIZATION 1, AMORTIZATION 2, ANALYSIS, ANE, BREAKEVEN, BUYING A HOME, CALORIES, CASHFLOW, COMPUND INTEREST, COUPON CATALOG, FINANCE, INVANALYST and LOAN.

DISK1987/11A - contents:

Great music, 12TH STREET RAG, BATTLING BYTES and HARRIGAN (a sing-a-long), some more home use utilities, NUTRITION CALCULATOR, PHONE CALL DATA BASE and STATISTICS plus two educational programs, one from the walk the plank series - similar to hangman - AMERICAN PRESIDENTS and a very well done program on guessing national flags.

DISK1987/11B - contents:

Video labeller program which is designed to enable you to print out adhesive labels to stick on the sides of your videos so you can see at a glance the contents. A very well done program which is set up for various printer control codes. Instructions are included on the disk.

DISK1987/11C - contents:

You probably have heard of Lotus 123, well this disk contains TI 123, and on the disk you will find a wordprocessor, 2 spreadsheets - one for 16K the other for 32K - and a graphics program. There are no instructions on the disk but it shouldn't be too hard to work out how to drive the programs.

Most of the programs on DISK87/11 and 11A came via Jim Peterson. Thanks Jim for the opportunity to pass on the software to our members. You may think it strange I am thanking Jim in this column, however Jim is on our mailing list and receives this magazine. As a matter of fact in his last letter he enquired if anyone was interested in light pen software as he has some available. I told Jim that no members as far as I was aware have a light pen but to send some software anyway as it might stimulate some interest in the building of light pens.

ON TAPE:

Hopefully the majority of programs on DISK1987/11 and 11A will be on tape and available at the November meeting. I say hopefully, because currently I am right out of blank tapes and am awaiting a further supply from our supplier. If they arrive in time for me to do the tapes then they will be on sale if not then they will be available at the December meeting.

THE COMMUNICATORS

Special Interest Group for Users
of the TEXPAC Bulletin Board Service.
by Ross Mudie, SYSOP, 10th October 1987.

1. USER SURVEY.

In September a survey was placed on the BBS for members to assist with the planning of material to be provided on the BBS. Of the 64 active users a total of 25 replied by 9th October 1987.
Here are the results of the survey:

a) 32K memory. 25 out of 25 indicated that they have a 32K memory expansion.

b) Satisfaction with downloadable programs.

22 indicated satisfaction, one said he was fairly satisfied.
Members would like:
(i) New 32K assembly programs
(ii) More educational programs
(iii) Language emulators.
(iv) More assembly utilities.
(v) Multiplan and Finance.

c) Users with disk drive.

21 out of 25 have disk drive. The BBS will continue to support non disk users who are obviously dependant on TEII for their terminal operating system.

d) Satisfaction with text files.

22 indicated satisfaction, 1 was fairly satisfied.
Members would like:
(i) More tutorials, including programming techniques.
(ii) More FORTH.
(iii) More on Geneve and other club news, local & overseas.
(iv) The ability to upload memory image format programs.
(v) A games room.
(vi) MUDIE disks on BBS.
(vii) A NEWS_INFO file simular to the PROG_INFO file.

I wish to thank those members who took the time to reply to the simple survey as with this guide from a small percentage of the BBS membership I will try to provide the type of service that the users want.

BBS members are asking for new software and tutorial material. Clearly I personally am not in a position to do the job of providing all the "new" material single handedly so it is over to the membership. Your individual programming skills are needed to keep your orphaned computer alive. I will be doing my bit in maintaining and improving the BBS, will you be doing your bit by providing some new programs, (which can be entered in the Software Competition), and writing some tutorial material??

The task of sorting through available material each month is very time consuming. The task is not unlike that of an editor with the end goal of providing an interesting mix of material, hopefully something for everyone. There simply are not enough hours to re-type material from overseas publications as well. My sincere thanks to the major contributors of material for the BBS namely George Meldrum, John Paine, Terry Phillips, Cyril Bohlsen, Robert Montgomery, Chris Buttner, Shane Ferrett & Shane Andersen. Regular or guest contributors are welcome.

Anyone willing to type up material from overseas news letters should contact SYSOP. If necessary I can post photocopies of suitable articles to willing typists who can then send the resultant file back on the BBS as electronic mail.

2. SENDING MAIL ON THE BBS.

The current mail facility on the BBS provides for mail sent manually from the user's terminal. The mail system stores the mail on a line by line basis to the mail disk of the BBS. All the mail for an individual user is placed in a common file for that user by a file opened in the APPEND mode. This architecture limits the ability to provide reply prompting or sender deletion facilities. The mail system also limits the file transfer technique to use of the SENDMAIL series of programs.

I am keen to see other "more convenient" methods of file transfer provided in the BBS as time permits. When this occurs the "Discussion Rooms" will also become available for general use.

3. COMMITTEE MEMBERS' FILES IN THE NEWS MENU.

The following TIsHUG committee members can now place their own files directly in the NEWS menu of the BBS without any need for SYSOP intervention. They are EDITOR, CO-ORD, SHOP, TECHOTIME and SECRETARY. These files will carry in the first line details of when the file was uploaded. I recommend that users look at each of these files when on the system as these files may be updated at any time, not just with the monthly file update. If you find that you have seen the file then just press E once to Escape the listing.

This feature is provided to give greater freedom to committee members for "up to the minute communications" to the members, remembering that a third of the TIsHUG membership have BBS access.

4. SENDING ARTICLES FOR THE TND.

One very easy way to submit an article for the TND is to send it to the username EDITOR on the BBS. The deadline for articles for each issue is the Saturday following the TIsHUG meeting. This avoids disappointment of an article not making it due to postal delays. If you reformat the file to 55 columns & provide right justification before transmission it will make the Editor's job a lot easier.

5. HARDWARE FAULTS.

The BBS suffers an occasional failure due to hardware faults which usually manage to render it inoperative until repairs can be made. In September the failure of an IC in the modem caused an evening of unavailibility whilst, during the October long weekend, a faulty cable connector on the rear of the disk controller card took the BBS out of service for 21 hours. (I had to make a special trip back to the BBS on the Monday morning of the long weekend to carry out the repair).
At present if the BBS program fails then usually the modem will still answer the call, but there is no response from the computer. A hardware "Watchdog" has been designed which will prevent the modem from answering if the BBS has failed. I hope to instal the watchdog during November as available time permits.

6. BBS ACCESS.

The BBS is open to members only and there is no "public" access to the BBS. The system is on line seven days per week, 24 hours per day.

Continued on P20

LINKING EXT'D BASIC-ASSEMBLY

13 LINK-IT 13

WITH ROSS MUDIE.

CONTROLLING SIMPLE DEVICES WITH THE TI99/4A JOYSTICK.
By Ross Mudie of TIsHUG, 25th September 1987.

A member of the BBS has asked if it is possible to control a simple robotic arm with the Joystick of the TI99/4A via the Cassette Motor control lines. The program which I developed runs in the Interrupt Service Routine in assembly under extended basic. This means that you can use the routine whilst another extended basic program is running or in the immediate mode.

The program uses both cassette motor controls, which are provided on the black consoles. On the beige consoles the printed circuit tracks exist, but the critical seven components were omitted by TI as a cost cutting measure in the later stages of production. These components are possibly easiest obtained from a junk TI99/4A mother board. Refer to the technical manual for the PCB overlay and sheet 3 of the schematic console diagram for details of physical component placement.

Remove components from the old mother board and place them in the appropiate positions in the mother board of of the computer which is being upgraded to provide two motor controls.

Old mother board	Upgraded Mother board	Type	Component
Q401	Q403	TIS92	Transistor
Q402	Q404	TIS92	Transistor
R415	R416	220 ohm	Resistor
U401	U402	TIL119	Optocoupler
C403	C405	.001uF	Capacitor
C404	C406	.001uF	Capacitor
L400	L401	Ferrite	Choke

Many of the consoles were provided with cassette cords with two cables which provide access to both motor control circuits. If a suitable cable is not available then a 9 pin female D connector may be used with connections to pins 1 - 2 for CS1 control and 6 -7 for CS2 control.

The extended basic program loads the assembly program in object form from disk and then links to place the start address of the joystick scan routine in the User Defined Interrupt Service Routine (ISR). Every 20 milliseconds the processor branches to the program in the ISR, scans the joy stick & outputs to the Cassette motor control lines then goes back to the main Program. In addition a display is given of UP , LEFT or RIGHT on the bottom right corner of the screen. It is possible to delete the screen display lines from the source file if display is not required or to display differently on another part of the screen.

The CS motor controls are: LEFT - CS1, RIGHT - CS2, UP - CS1 and CS2. For any other Joystick position CS1 and CS2 controls are off & the screen display is ——— .

From here on users are on their own as far as the hardware interface is concerned. If you are unable to figure out how to provide a suitable interface then ask around your regional group for someone with the nesessary expertise. To test the outputs from the computer I suggest two circuits as shown on page 19 of the October 1986 TND which shows how to connect a Light Emitting Diode, resistor and battery.

The source file is included for those with Editor/ Assembler whilst a version completely in extended basic CALL LOADs is provided for those who don't have the E/A but want to play around with the program. I suggest that after loading and running that you then load an extended basic music program and be amazed as both run together without affecting each other. Remember that the assembly program requires a 32K memory expansion to be connected to the computer.

For further reading on the methods used for keyboard scan and control of the Cassette Motors refer to LINK-IT #7 on pages 19-21 of the October 1986 TND.

Just a word of caution, if you have the interrupt program running, don't use CALL INIT else the computer will lock up. If you want to execute CALL INIT then you must use CALL LINK("STOP") first. The joystick scan routine doesn't have to operate under an ISR, I just used the ISR because it made the program very versatile. To use the program as a directly linked routine just include the entry point name INTRTN in the DEF list.

Extended basic program.

```
100 ! SAVE DSK1.LOAD
110 CALL CLEAR
120 CALL INIT
130 CALL LOAD("DSK1.O")
140 CALL LINK("JSCS")
150 DISPLAY AT(6,1):"TRY JOYSTICK 1:  UP, LEFT & RIGHT
THEN FCTN CLEAR AND TRY IN  IMMEDIATE MODE"
160 DISPLAY AT(14,1):"If in immediate mode type": :"CAL
L LINK("&CHR$(34)&"STOP"&CHR$(34)&") before": :"rerunni
ng lines 120 to 140"
170 ! here insert RUN "DSK1.NEXTPROG" to go to another
program with joystick control routine still running
180 GOTO 180
```

ASSEMBLY SOURCE FILE.

```
        IDT  'JSCSMTR'      File names: Source=S
                                        Object=O
        DEF  JSCS,STOP                  Entry names

VMBW    EQU  >2024              Video Multiple Byte Write

UP      DATA >80B5,>B080,>8000   UP      Text messages
LEFT    DATA >ACA5,>A6B4,>8000   LEFT    with >60 added
RIGHT   DATA >B2A9,>A7A8,>B400   RIGHT   for assembly to
BLANK   DATA >8D8D,>8D8D,>8000   ----  display under x/b

WS      BSS  32                     Register Work Space

*   CALL LINK("JSCS")           Turn ON Joystick control

JSCS    LI   R1,INTRTN      Address of interrupt program
        MOV  R1,@>83C4       Put entry point in User ISR
        RT

*   CALL LINK("STOP")           Turn OFF Joystick control

STOP    CLR  @>83C4          Turn off Interrupt program
        RT

INTRTN LWPI WS                  Load register work space
        CLR  R12                  Set base value for CRU

        SBZ  18                Set up CRU for Joystick 1
        SBO  19
        SBO  20

        TB   7                              Test for UP
        JEQ  NEXT1                      Jump if not UP
        SBO  22                          Turn CS1 ON
        SBO  23                          Turn CS2 ON
        LI   R1,UP                   Text to display
        JMP  NEXT3                      Go and do it
```

Continued on P27

Implanting Machine Code into BASIC Programs

by George Meldrum

This is to be a topic for the tutorial day. These notes are meant to supplement the talk and hopefully supply an adequate introduction to the topic.

To explain what we are trying to achieve imagine your BASIC program is a suitcase. Next think of fitting a false bottom to the case inside which you hide your secret code. Open the case (LIST) and all looks normal. Carry the case (OLD and SAVE) and the false bottom goes with it.

BASIC programs which have machine code programs embeded behind them provide a quick and easy form for the computer user. From a users view the advantages are:
(1) Program loads and runs like any other XB program
(2) No special loaders are required
(3) Programs can be easily copied between tape / disk

The hardware required for this form of production technique is : Console + Extended BASIC + 32K memory then either a cassette recorder, or disk drive. The software required is a Debugger type program such as that supplied on the Editor/Assembler disk. If you do not have access to a Debugger program then you can copy a slightly modified version available on the tutorial day. A copy will also go to the software library and probably make its way to the club shop.

So that no complications are introduced select a single, Editor/Assembler option 5 file, for implanting. Keep a note pad handy to record the values displayed while using the Debugger. If you make an incorrect entry in the Debugger then type fctn-x to terminate the command. Be careful to include the "V" where indicated. Now down to it :-

(1) Load the Debugger program :
CALL INIT :: CALL LOAD("DSK1.DEBUGX") -disk users
RUN "CS1" -for cassette users

(2) Load the E/A opt 5 machine code file :
OLD DSK1.filename
OLD CS1
After loading an * I/O ERROR 50 message must be displayed.

(3) Link to the Debugger program :
CALL LINK("DEBUG")

(4) Cassette users ignore this step.
Type :
.M 969V <enter>
0969 = kk <enter>

(5) Cassette users ignore this step.
Type :
.H 966,kk <enter> (use the value kk from step 4)
look for H1+H2=tttt

(6) Disk users use the value tttt from step 5, and calculate the value uuuu as tttt + 5.
Type :
.M ttttV,uuuuV <enter> -disk users
.M 969V,96EV <enter> -cassette users
Output :
0969 = 00 00 ee ee dd dd ******

(7) Find the length of code.
Type :
.H eeee,6 <enter> (eeee from step 6)
look for H1-H2=nnnn

(8) Calculate where to put the code in high-memory.
Type :
.H FF9C,nnnn <enter> (nnnn from step 7)
look for H1-H2=ssss

(9) Cassette users ignore this step.
Work out where the code starts in vram.
Type :
.H tttt,6 <enter> (tttt from step 6)
look for H1+H2=cccc

(A) Transfer code from vram buffer to high-memory.
Use the value cccc from step 9
value ssss from step 8
value nnnn from step 7
Type :
.N ccccV,ssss,nnnn <enter> -disk users
.N 096FV,ssss,nnnn <enter> -cassette users

(B) Calculate new last free address in high-memory.
Type :
.H ssss,1
look for H1-H2=aaaa

(C) Set BASIC's start and end of line number table pointers.
Type :
.M 8330 <enter>
8330 = FFE7 aaaa <space bar>
8332 = FFE7 aaaa <enter>

(D) ** Full system users see note at end of step 10.
Poke in the following code (note that XXXX displayed is an unknown and unimportant value).
Type :
.M FF9C <enter>
FF9C = XXXX 02E0 <space bar>
FF9E = XXXX 3FC0 <space bar>
FFA0 = XXXX 0200 <space bar>
FFA2 = XXXX 03F0 <space bar>
FFA4 = XXXX 0201 <space bar>
FFA6 = XXXX A000 <space bar>
FFA8 = XXXX 0202 <space bar>
FFAA = XXXX 0310 <space bar>
FFAC = XXXX 0420 <space bar>
FFAE = XXXX 202C <space bar>
FFB0 = XXXX 0200 <space bar>
FFB2 = XXXX 08F0 <space bar>
FFB4 = XXXX 0420 <space bar>
FFB6 = XXXX 2024 <space bar>
FFB8 = XXXX 0200 <space bar>
FFBA = XXXX 030E <space bar>
FFBC = XXXX 0420 <space bar>
FFBE = XXXX 2030 <space bar>
FFC0 = XXXX 0200 <space bar>
FFC2 = XXXX 0401 <space bar>
FFC4 = XXXX 0420 <space bar>
FFC6 = XXXX 2030 <space bar>
FFC8 = XXXX 0200 <space bar>
FFCA = XXXX 07F5 <space bar>
FFCC = XXXX 0420 <space bar>
FFCE = XXXX 2030 <space bar>
FFD0 = XXXX 0200 <space bar>
FFD2 = XXXX ssss <space bar>
(take value ssss from step 8)
FFD4 = XXXX 0201 <space bar>
FFD6 = XXXX dddd <space bar>
(take value dddd from step 6)
FFD8 = XXXX 0202 <space bar>
FFDA = XXXX nnnn <space bar>
(take value nnnn from step 7)
FFDC = XXXX CC70 <space bar>
FFDE = XXXX 0642 <space bar>
FFE0 = XXXX 16FD <space bar>
FFE2 = XXXX 0460 <space bar>
FFE4 = XXXX dddd <space bar>
(take value dddd from step 6)
FFE6 = XXXX 0000 <enter>

Continued on P10

(E) Exit the Debugger.
Type :
.Q <enter>

(F) You are now back to BASIC. Type the following:
```
100 REM PROGRAM NAME
110 REM
120 CALL INIT
130 CALL LOAD(8196,63,248,"",16376,71,65,77,69,
    32,32,255,156)
140 CALL LINK("GAME")
```

(10) You MUST remember to save the program now !
Type :
SAVE DSK1.filename
SAVE CS1

You can now try RUNning the program now. Not all programs meant for the Editor/Assembler environment will work with Extended BASIC. You may edit the BASIC program with the only restriction being that you cannot RESEQUENCE (resequencing can effect the embeded machine code).

** Full system users can, if they wish, skip step D and continue with steps E, F, and 10. They can then create and assemble the following code :-

```
*    Emulate E/A environment + code relocation
*
       AORG >FF9C   = >FFE8 - length of this file
VMBW   EQU  >2024
VMBR   EQU  >202C
VWTR   EQU  >2030
START  EQU  >dddd   enter load addr found in step 6
LOAD   EQU  >dddd   enter load addr found in step 6
LEN    EQU  >nnnn   enter length calculated step 7
FROM   EQU  >ssss   enter location found in step 8
*
       LWPI >3FC0       my workspace
       LI   R0,>03F0
       LI   R1,>A000
       LI   R2,>0310
       BLWP @VMBR       read character definitions
       LI   R0,>08F0
       BLWP @VMBW       relocate to E/A position
       LI   R0,>030E
       BLWP @VWTR       E/A default color table
       LI   R0,>0401
       BLWP @VWTR       E/A default pattern table
       LI   R0,>07F5
       BLWP @VWTR       E/A default screen color
*
       LI   R0,FROM
       LI   R1,LOAD
       LI   R2,LEN
LOOP   MOV  *R0+,*R1+   relocate code
       DECT R2
       JNE  LOOP
*
       B    @START      execute program
       DATA 0
       END
```

This same source code can be used each time with the only adjustment required being the four EQUate directives at the start. Now reload the BASIC code saved in step 10 and load the newly assembled code above.

```
OLD DSK1.filename
CALL INIT
CALL LOAD("DSK1.object_filename")
SAVE DSK1.filename
```

Now the code is embeded the program can be run at any time by: RUN"DSK1.filename".

o

Debugs in de Print - by Shane Ferret

Correction to VDPUTIL3 utility
Micropendium, July 1987, page 36.

LINES SHOULD READ AS FOLLOWS

```
32300 CALL LOAD(9726,4,192,2
16,0,131,124,2,224,131,224,4
,96,0,112):: CALL LOAD(8194,
39,4)::SUBEND
```

NB. 9726 INSTEAD OF 3726

```
32320 FOR B=1 TO 16 STEP 2 :
: C=ASC(SEG$(A$,B,1)):: D=AS
C(SEG$(A$,B+1,1)):: C=((C>57
)*7-48+C)*16+(D>57)*7-48+D
```

NB. the plus sign after 16 before bracket

```
32330 CALL LINK("POKEV",767+
8*A+(B+1)/2,C):: NEXT B :: S
UBEND
```

NB. SUBEND ADDED

```
32340 SUB COLOR(A,B,C) :: CA
LL LINK("POKEV",2063+A,(B-1)
*16+C-1):: SUBEND
```

NB. REMOVAL OF SUBEND

ONCE THE PROGRAM IS OPERATIVE MODS CAN BE MADE SUCH AS THE MULITPLE CALL LOAD STATEMENT CALL LOAD(ADDR1,D1,D2,D3,D4,"",ADDR2,D5,D6,D7) TO SHORTEN IT SLIGHTLY

AS A MATTER OF INTEREST FOLLOWING IS THE SOURCE FOR THE CALL LOADS

```
       DEF  POKEV

FAC    EQU  >834A
NUMREF EQU  >200C
XMLLNK EQU  >2018
VMBW   EQU  >2024
PAD    EQU  >8300
GPLST  EQU  >837C
GPLWS  EQU  >83E0
NEXT   EQU  >0070
       AORG 9472          >2500
POKADD BSS  2
INFOBF BSS  16
TEMP1  DATA 100
POKEWS BSS  32
       AORG 9636          >25A4
* START OF PROGRAM
POKEV  LWPI POKEWS      GET OWN WORKSPACE
       LIMI 0           DISABLE INTERUPTS
* GET FIRST PARAMETER FROM BASIC
       CLR  R0
       LI   R1,>1
       BLWP @NUMREF
       BLWP @XMLLNK        CONVERT TO INTEGER
       DATA >12B8
       MOV  @FAC,@POKADD   STORE AT COUNTER
       AB   @PAD+18,@TEMP1+1
* GET REMAINING PARAMETERS FROM BASIC
       LI   R3,>2
MORINP CLR  R0
       MOV  R3,R1
       BLWP @NUMREF
       BLWP @XMLLNK
       DATA >12B8
       MOVB @FAC+1,@POKADD(R3) STORE IN BUFFER
       INC  R3
       C    R3,@TEMP1      LAST PARAMETER?
       JNE  MORINP         NO LOOP FOR ANOTHER
       MOV  @POKADD,R0     GET POKE ADDR IN VDP
       LI   R1,INFOBF      POINT TO INFO TO POKE
       MOV  R3,R2          GET INFO BYTE COUNT
       AI   R2,-2          ADJUST COUNT BY 2-
       BLWP @VMBW          POKE VALUES IN VDP
* EXIT ROUTINE
       CLR  R0             RESET STATUS
       MOVB R0,@GPLST      i.e. no error
       LWPI GPLWS          RESTORE GPL WORKSPACE
       B    @NEXT          GOTO NEXT STATEMENT
                           'RETURN TO BASIC'
```

o

Jenny's YOUNGER SET under 18's page

Hi! Gang,

Just when I thought you had lost interest, along comes some more mail.

This month Peter Mudie has sent in a program. Peter is 11 years old and his degree of documentation shows many other programmers up. I am sure that if Peter keeps this standard up he will soon be able to teach his father a trick or two.

How many of you have read about the great software competition that TIsHUG is running? Big prizes can be won, so how about an entry?

I would like to see some mail come in for next month's TND, preferably on a Christmas theme. Boy, time does fly.

Jenny

DEAR JENNY,
Could you put this program in the Jenny's Younger Set page please. This is a guess the number program.

```
100 ! SAVE DSK1.GUESSTHENO
110 CALL DELSPRITE(ALL) :: CALL MAGNIFY(2) :: CALL CHAR
(96,"FFFFFFFFFFFFFFFF") !       Block char for border.
120 X=0 :: CALL CLEAR :: CALL SCREEN(6):: FOR S=0 TO 12
:: CALL COLOR(S,16,1):: NEXT S !Color for normal chars.
130 CALL COLOR(9,15,15):: CALL HCHAR(1,2,96,30):: CALL
VCHAR(1,31,96,24) :: CALL HCHAR(24,2,96,30) :: CALL
VCHAR(1,2,96,24) !       Drawing the outside border.
140 RANDOMIZE !Randomizing for the number to be guessed
150 DISPLAY AT(2,6):"GUESS THE NUMBER ": :TAB(7);"BY PE
TER MUDIE.": : : :TAB(9);"PRESS A NUMBER 1-9"
160 Z=INT(RND*9)+1 ! Making "Z" equal to the random no.
170 X=X+1 :: CALL SOUND(130,900,0)! Bip says it's ready
180 CALL KEY(3,K,S):: IF S=0 THEN 180  ! Test key press
190 DISPLAY AT(4,1) :: IF K<49 OR K>57 THEN CALL SOUND
(-140,200,0) :: GOTO 180 !          Invalid key press
200 DISPLAY AT(8,1):"YOU GUESSED" :: CALL SPRITE(#1,K,
16,49,115) ! The sprite shows the number guessed
210 K=K-48 !The 48 has to be taken from K for IF K=Z..
220 IF K=Z THEN 260 !Testing for a correct or wrong ans
230 IF K<Z THEN DISPLAY AT(8,16):"TRY HIGHER" ELSE DIS
PLAY AT(8,16):"TRY LOWER " ! Hints higher or lower.
240 DISPLAY AT(15,1):"THAT WAS TRY ";X !Telling try no.
250 CALL KEY(3,KK,SS) :: IF SS<>0 THEN 250 ELSE 170
260 IF X=1 THEN A$="GO" ELSE A$="GOES" !  Spelling set
270 DISPLAY AT(15,1):"CORRECT IN ";X;A$;" !": : : : : :
: :"PLAY AGAIN Y/N?" ! Correct & PLAY AGAIN... message
280 CALL SOUND(100,1400,0)!Beep says computer is ready.
290 CALL KEY(2,K,S):: IF S=0 THEN 290 ELSE IF K=18 THEN
100 ELSE IF K=15 THEN END ELSE 290 !Test PLAY AGAIN Y/N
```

★

ADVENTURE HINTS - Return to Pirate's Isle

Chapter 4

This month's clues are:

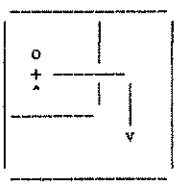
Make glue, glue lens, to mask, empty mask, go sea, wash mask, wear mask, go beach, drop rim, go west, go south, go boat, go sea, hold breath, swim down, swim west, swim opening, go boat, open box, drop amber, drop earring, drop doublo, go pool, hold breath, swim down, swim east, swim up, go boat, go down, go down, start engine, go up, drive boat, go down, stop engine, go up, press button, go up, go sea, remove mask, wash mask, wear mask, hold breath, swim down, go ship, get chest, swim up, go boat, go down, press butt, go down, start engine, go up, drive boat, go up, go dock, go sea, remove mask.....

If Jenny's gang send in mail, by the 13th Nov, then the final clues will be published next month.

o

****
* PICASSO TIPS - Vers 2.0 *
****

1. Moving large chunks of graphics from one part of the large screen to another part.eg..



One way of achieving the result is to use the Move option, by putting the pic into the bottom RH corner then shifting the screen viewing area until it is at the top LH corner.

This method might be OK if you haven't anything drawn there. The other way is to first save the whole file, now move it to the position without worring about what gets in the way, now save the current screen using option 4 on the file utilities. Load your original file, then overlay the graphic file using option 3 on the file utilities and voila you done it.

2. Erasing large graphic chunks is easy by using the Window option and pressing the 'U' to toggle pixels off then press 'A' to fill framed area with blanks.

3. Version 2.0 has few extra features.eg.

```
ICON DRAW      'H'
ICON EDITOR    '.'
ICON SAVE      '3'
ICON LOAD      '4'
TRANSPARENT COPY 'Z'(toggle)
33 TEXTURES
```

4. Also I decided to make the slanted lines from the brushes only 1 pixel wide. The Text cursor is now a square to make it easier to locate the 8*8 Text letters. Area fill was added to Ver 1.4.

5. The Icon option is a very versatile addition, you can now have either a large letter 16*16 or if you load the 'ICON-Z****' it gives you a spray can. When drawing with this option the pixels will to step by 3 , just press the joy slow '-' 2 times to draw continuos pixels. You have 5 ICONS available.

6. I have about a disk full of large character sets that can be loaded in with option 2 & 3 of the file utilites.

Version 2 has about 20 bytes free out of all the CPU & VDP areas, this will mean a restriction as far new or improved versions in the future.

One additional program that will be hopefully finished soon is 'ANIMATOR' it will use the 24 sector files created by Picasso.

* * *

I HAVE ONE MORE THUNDERER MODEM FOR SALE.
This is a new style stand-alone modem and is a pre-production unit made for ACETRONICS. I will not be making more of these (no time). It has 300 BAUD, 1200/75 and VIATEL, complete with RS232 cable for the TI.
If interested give me a call on (02) 358 5602 or write to P.O.BOX 28 KINGS CROSS.
Peter Schubert.

```
100 DISPLAY ERASE ALL :: OPT
ION BASE 1 :: DIM P$(66,4)::
 CALL TL :: CALL OP(S$,D$,N,
C):: LINPUT #1:N$
110 CALL PB(P$(,),C,N,E):: C
ALL PR(P$(,),D$,C):: IF E=0
THEN 110 ELSE CLOSE #1 :: CL
OSE #2 :: CALL S("GOODBYE"):
: STOP
120 SUB TL :: CALL SCREEN(5)
:: FOR I=0 TO 13 :: CALL COL
OR(I,16,5):: NEXT I :: CALL
CHAR(128,RPT$("F",16)&"00000
OFF")
130 CALL FL(7,6,"PRINTER
LISTING","N?"):: DISPLAY AT(
11,13):"from" :: CALL FL(15,
7,"FUNNELWEB  FARM","N?")
140 CALL S("HELLO+PARTNER TH
IS+IS+YOUR+#TEXAS INSTRUMENT
S# NINETY+NINE+FOUR+A PROGRA
M+PRINTER")
150 CALL S("REMEMBER TURN+ON
+YOUR+PRINTER")
160 CALL S("WHEN+#READY TO S
TART#+PRESS+ANY+KEY"):: DISP
LAY AT(24,6)BEEP:"ANY KEY TO
 PROCEED"
170 CALL KEY(3,I,ST):: IF ST
=0 THEN 170
180 DISPLAY ERASE ALL :: SUB
END
190 SUB DL(A):: FOR A=1 TO A
 :: NEXT A :: SUBEND
200 SUB OP(S$,P$,NS,C):: S$=
"DSK1.LIST" :: P$="RS232.BA=
4800"
210 CALL FL(1,2,"Edit as nee
ded and ENTER","N?"):: CALL
S("PLEASE+GIVE+ME+YOUR+INSTR
UCTIONS")
220 CALL FL(4,4,"Source file
 for listing",S$)
230 IF SEG$(S$,1,3)<>"DSK" T
HEN DISPLAY AT(6,4)BEEP:"Whe
re's your input" :: CALL S("
#WHAT WAS THAT#"):: GOTO 220
240 CALL FL(8,4,"Printer dev
icename",P$)
250 IF P$="" THEN DISPLAY AT
(10,4)BEEP:"Output device ?"
 :: CALL S("UHOH YOU+ARE+#SU
PPOSED TO#+ENTER+A+DEVICE+NA
ME+FOR+THE+PRINTER"):: GOTO
240
260 CALL YN("Space REM, SUB"
,"Y",16,5,X):: NS=NOT(X)
270 CALL YN("Compress print"
,"N",18,5,X):: IF X THEN C=2
 :: GOTO 290
280 DISPLAY AT(18,5)BEEP:"#
of columns (3/4)?  3" :: ACC
EPT AT(18,26)VALIDATE("34")S
IZE(-1)BEEP:A$ :: IF A$="" T
HEN 280 ELSE C=VAL(A$)
290 CALL YN(" Change mind ?"
,"N",24,5,X):: IF NOT(X)THEN
 CALL S("UHOH #TRY AGAIN#"):
: DISPLAY AT(24,1):: GOTO 22
0
300 DISPLAY ERASE ALL :: OPE
N #1:S$,DISPLAY ,INPUT ,VARI
ABLE 80 :: OPEN #2:P$,DISPLA
Y ,OUTPUT,VARIABLE 132 :: SU
BEND
310 SUB PB(PN$(,),CM,NS,E)
320 P=6 :: C=C+1 :: IF C>CM
THEN C=0 :: SUBEXIT ELSE PRI
NT "";"___":"*** Reading col
umn #";C;"___":""
330 IF O$>"" THEN N$=O$ :: O
$="" :: GOTO 350
340 IF E THEN PRINT "":" *":
"*** End of File ***":" *":"
" :: SUBEXIT ELSE CALL BL(N$
,E)
350 CALL WC(P,C,PN$(,),N$,O$
,NS):: IF N$="EOC" THEN 320
ELSE 330
360 SUBEND
370 SUB RS(N$,S):: A$=SEG$(N
$,POS(N$," ",1)+1,4):: IF A$
="!@P+" OR A$="!@P-" THEN S=
0 :: SUBEXIT
380 IF A$="REM " OR SEG$(A$,
1,1)="!" THEN S=NOT(RF):: RF
=-1 :: SUBEXIT ELSE RF=0
390 IF A$="SUB " THEN S=NOT(
S)ELSE S=0
400 SUBEND
410 SUB SL(P,C,P$(,),N$)
420 P=P+1 :: IF LEN(N$)>28 T
HEN P$(P,C)=SEG$(N$,1,28)::
N$=SEG$(N$,29,LEN(N$)-28)::
GOTO 420 ELSE P$(P,C)=N$ ::
N$="" :: SUBEXIT
430 SUBEND
440 SUB WC(P,C,P$(,),N$,O$,N
S):: IF NC THEN P=6 :: NC=0
450 IF NS THEN 470 ELSE CALL
 RS((N$),S):: IF S AND P>57
THEN P=61 :: GOTO 470
460 IF S AND P<>6 THEN PRINT
 "" :: O$=N$ :: N$=" " :: PR
INT "";"** Line space insert
ed #";P :: GOTO 480
470 IF P-INT(-LEN(N$)/28)>=6
0 THEN O$=N$ :: N$="EOC" ::
NC=-1 :: SUBEXIT ELSE O$=""
480 PRINT "";N$:"" :: CALL S
L(P,C,P$(,),N$):: SUBEND
490 SUB PR(P$(,),D$,C):: PRI
NT "";"** Page print startin
g";"":"   to  "&D$:"" :: CAL
L S("START PRINT")
500 IF C>2 THEN PRINT #2:CHR
$(15)ELSE PRINT #2:""
510 FOR I=2 TO 65 :: IF (P$(
I,1)="")AND(P$(I,2)="")AND(P
$(I,3)="")AND(P$(I,4)="")AND
(I<6)AND(I>59)THEN PRINT #2:
"" :: GOTO 560
520 IF C=2 THEN PRINT #2:TAB
(9);P$(I,1);TAB(45);P$(I,2)
530 IF C=3 THEN PRINT #2:TAB
(14);P$(I,1);TAB(48);"|";TAB
(54);P$(I,2);TAB(88);"|";TAB
(94);P$(I,3)
540 IF C=4 THEN PRINT #2:TAB
(8);P$(I,1);TAB(37);"|";TAB(
39);P$(I,2);TAB(68);"|";TAB(
70);P$(I,3);TAB(99);"|";TAB(
101);P$(I,4)
550 P$(I,1),P$(I,2),P$(I,3),
P$(I,4)=""
560 NEXT I :: IF C>2 THEN PR
INT #2:CHR$(18)&CHR$(12)ELSE
 PRINT #2:CHR$(12)
570 SUBEND
580 SUB BL(N$,E):: N$="" ::
IF NX$="" THEN LINPUT #1:NX$
590 N$=N$&NX$ :: IF LEN(NX$)
<80 OR EOF(1)THEN NX$="" ::
E=EOF(1):: SUBEXIT ELSE LINP
UT #1:NX$
600 PX=POS(NX$," ",1):: IF P
X<2 OR PX>6 THEN 590
610 P=POS(N$," ",1):: IF PX<
P THEN 590
620 NR=-1 :: FOR I=1 TO PX-1
 :: C=ASC(SEG$(NX$,I,1)):: N
R=NR AND C>47 AND C<58 :: NE
XT I :: IF NOT(NR)THEN 590
630 IF SEG$(N$,LEN(N$),1)="
" THEN 590
640 IF VAL(SEG$(NX$,1,PX-1))
<=VAL(SEG$(N$,1,P-1))THEN 59
0
650 M$=SEG$(N$,4,LEN(N$)-4):
: IF M$="THEN" OR M$="ELSE"
OR M$="GOTO" OR M$="OSUB" OR
 M$="RROR" THEN 590
660 NQ,I=0
670 I=POS(N$,CHR$(34),I+1)::
 IF I THEN NQ=NQ+1 :: GOTO 6
70 ELSE IF NQ<>2*INT(NQ/2)TH
EN 590
680 SUBEND
690 SUB YN(A$,B$,R,C,X):: DI
SPLAY AT(R,C)BEEP:A$&" (Y/N)
 "&B$
700 ACCEPT AT(R,C+LEN(A$)+7)
VALIDATE("YN")SIZE(-1)BEEP:A
$ :: X=A$=B$ :: R=R+2 :: SUB
END
710 SUB FL(R,C,M$,D$):: DISP
LAY AT(R+1,C):RPT$(CHR$(129)
,LEN(M$)):: DISPLAY AT(R,C):
M$ :: IF D$<>"N?" THEN DISPL
AY AT(R+2,C):D$ ELSE SUBEXIT
720 ACCEPT AT(R+2,C)SIZE(-14
)BEEP:D$ :: IF D$="N?" THEN
DISPLAY AT(R+2,C):: GOTO 720
730 SUBEND
740 SUB S(A$):: CALL PEEK(-2
8672,S):: IF S=96 THEN CALL
SAY(A$)ELSE CALL DL(5*LEN(A$
))
750 SUBEND
```

Tutorial Day, 9am 7/11/87.
at
WOODSTOCK COMMUNITY CENTRE
Church St Burwood.

Program:

Morning: John Paine on Multiple Ramdisk Access using version 6.3 software.

Ross Mudie on Beginners Linked Assembly.

Lunch provided at a small fee.

Afternoon: George Meldrum on how to put Assembly in an x/b program.

Basic Electronics for beginners.

Peter Schubert will demonstrate the new PE Box card which incorporates RS232, PIO, 32K memory and advanced disk controller on ONE card.

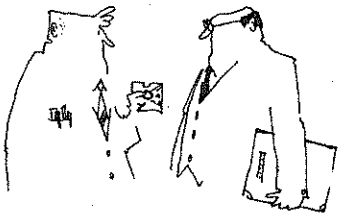
The SHOP will be open on the day.

Topics of future meetings...

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

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

```
100 CALL CLEAR
110 RANDOMIZE
120 A=0
130 B=12
140 C=0
150 CALL CHAR(96,"1818181818
181818")
160 CALL CHAR(104,"6699997E5
595A5A5")
170 CALL CHAR(112,"3C7EFFFFF
F7E3C3C")
180 CALL CHAR(113,"3C3C7EFFF
F7E3C3C")
190 CALL CHAR(97,"000000FFFF
")
200 FOR E=1 TO 9
210 CALL COLOR(E,16,1)
220 NEXT E
230 CALL SCREEN(2)
240 CALL COLOR(10,14,1)
250 CALL COLOR(11,3,1)
260 FOR E=1 TO 10
270 D(E)=3
280 NEXT E
290 A$="SPIDER BOP"
300 F=4
310 G=11
320 GOSUB 890
330 F=22
340 G=5
350 A$="USE <ARROW KEYS> TO
MOVE"
360 GOSUB 890
370 F=8
380 G=3
390 A$="WHAT ARE YOUR INITIA
LS? ___"
400 GOSUB 890
410 NAME$=""
420 FOR E=27 TO 29
430 CALL KEY(0,KEY,ST)
440 IF ST=0 THEN 430
450 CALL HCHAR(8,E,KEY)
460 CALL SOUND(10,500,0)
470 NAME$=NAME$&CHR$(KEY)
480 NEXT E
490 F=10
500 A$="SET UP NEW HIGH SCOR
E FILE?N"
510 GOSUB 890
520 CALL KEY(0,KEY,ST)
530 IF ST=0 THEN 520
540 CALL SOUND(10,500,0)
550 IF KEY=13 THEN 570
560 CALL HCHAR(10,30,KEY)
570 CALL GCHAR(10,30,TE)
580 IF TE<>89 THEN 600
590 GOSUB 2050
600 F=24
610 G=3
620 A$="PRESS <ENTER> TO BOP
 SPIDERS"
630 GOSUB 890
640 A$="PUNCH POWER:"
650 F=12
660 G=10
670 GOSUB 890
680 A$="(H)I (M)ED (L)OW"
690 F=14
700 G=8
710 GOSUB 890
720 CALL KEY(0,H,I)
730 IF I=0 THEN 720
740 IF (H<>72)*(H<>77)*(H<>7
6)THEN 720
750 IF H<>72 THEN 780
760 J=1
770 GOTO 830
780 IF H<>77 THEN 810
790 J=2
800 GOTO 830
810 IF H<>76 THEN 830
820 J=3
830 GOSUB 930
840 F=1
850 G=17
860 GOSUB 1040
870 GOSUB 1470
880 GOTO 860
890 FOR E=1 TO LEN(A$)
900 CALL HCHAR(F,G+E-1,ASC(S
EG$(A$,E,1)))
910 NEXT E
920 RETURN
930 F=1
940 CALL CLEAR
950 G=10
960 A$="SCORE: 0"
970 GOSUB 890
980 FOR E=1 TO 10
990 CALL HCHAR(3,E*3,96)
1000 CALL HCHAR(4,E*3,104)
1010 NEXT E
1020 CALL HCHAR(24,B,112)
1030 RETURN
1040 A=INT(RND*10)+1
1050 IF D(A)+J>23 THEN 1110
1060 D(A)=D(A)+J
1070 CALL SOUND(-50,110+RND*
50,0)
1080 CALL VCHAR(D(A)-J,A*3,9
6,J)
1090 CALL HCHAR(D(A),A*3,104
)
1100 RETURN
1110 CALL HCHAR(D(A),A*3,32)
1120 B=B+1
1130 FOR E=A*3 TO B STEP SGN
(B-A*3)
1140 CALL HCHAR(24,E,104)
1150 CALL HCHAR(24,E,32)
1160 NEXT E
1170 CALL HCHAR(24,E,104)
1180 FOR E=1 TO 400
1190 NEXT E
1200 CALL CLEAR
1210 F=1
1220 G=12
1230 A$="GAME OVER"
1240 GOSUB 890
1250 GOSUB 1810
1260 OPEN #1:"DSK1.SF",INTER
NAL,VARIABLE
1270 FOR I=5 TO 1 STEP -1
1280 INPUT #1:A1,A1$
1290 F=5+I*2
1300 G=13
1310 A$=A1$
1320 GOSUB 890
1330 G=17
1340 A$=STR$(A1)
1350 GOSUB 890
1360 NEXT I
1370 CLOSE #1
1380 F=20
1390 G=3
1400 A$="PRESS <ENTER> TO PL
AY AGAIN."
1410 GOSUB 890
1420 CALL KEY(0,H,I)
1430 IF I=0 THEN 1420
1440 IF H=13 THEN 100
1450 CALL CLEAR
1460 END
1470 CALL KEY(0,H,I)
1480 IF I=0 THEN 1640
1490 CALL SOUND(-10,-3,5)
1500 IF H<>68 THEN 1560
1510 CALL HCHAR(24,B,32)
1520 IF B<30 THEN 1540
1530 B=0
1540 B=B+3
1550 CALL HCHAR(24,B,112)
1560 IF H<>83 THEN 1620
1570 CALL HCHAR(24,B,32)
1580 IF B>3 THEN 1600
1590 B=33
1600 B=B-3
1610 CALL HCHAR(24,B,112)
1620 IF H<>13 THEN 1640
1630 GOSUB 1650
1640 RETURN
1650 FOR K=24 TO 5+J*2 STEP
-1
1660 IF K=D(B/3)THEN 1700
1670 CALL HCHAR(K,B,112)
1680 NEXT K
1690 GOTO 1770
1700 CALL SOUND(100,-5,0)
1710 CALL VCHAR(K-4+J,B,32,4
)
1720 CALL HCHAR(K-5+J,B,104)
1730 D(B/3)=K-5+J
1740 C=C+1
1750 A$=STR$(C)
1760 GOSUB 890
1770 FOR E=K TO 23
1780 CALL HCHAR(E,B,32)
1790 NEXT E
1800 RETURN
1810 OPEN #1:"DSK1.SF",VARIA
BLE,INTERNAL
1820 FOR I=1 TO 5
1830 INPUT #1:HS(I),NAM$(I)
1840 NEXT I
1850 IF HS(1)>C THEN 2030
1860 HS(1)=C
1870 NAM$(1)=NAME$
1880 FOR I=1 TO 4
1890 FOR J=I+1 TO 5
1900 IF HS(I)<=HS(J)THEN 197
0
1910 CHANGE=HS(I)
1920 NN$=NAM$(I)
1930 HS(I)=HS(J)
1940 NAM$(I)=NAM$(J)
1950 HS(J)=CHANGE
1960 NAM$(J)=NN$
1970 NEXT J
1980 NEXT I
1990 RESTORE #1
2000 FOR I=1 TO 5
2010 PRINT #1:HS(I),NAM$(I)
2020 NEXT I
2030 CLOSE #1
2040 RETURN
2050 OPEN #1:"DSK1.SF",VARIA
BLE,INTERNAL
2060 FOR I=1 TO 5
2070 PRINT #1:0,"___"
2080 NEXT I
2090 CLOSE #1
2100 RETURN
```



'No! No! No! I said an antique dEsk!'

```
100 CALL CLEAR
110 REM  BASIC NUMBER FACTS
      BY COL CHRISTENSEN,
    17 CENTAUR ST,REDCLIFFE.

120 CALL CHAR(37,"0101010101
010101")
130 CALL CHAR(39,"8080808080
808080")
140 CALL CHAR(96,"071F3F3F7F
7B717B")
150 CALL CHAR(97,"E0F8FCFCFE
DE8EDE")
```

Continued on P14

```
160 CALL CHAR(98,"7E7E3F3B3C
1F0F03")
170 CALL CHAR(99,"7E7EFCDC3C
F8F0C0")
180 CALL CHAR(100,"7E7E3F3C3
B1F0F03")
190 CALL CHAR(101,"7E7EFC3CD
CF8F0C0")
200 CALL COLOR(9,16,10)
210 CALL CLEAR
220 RANDOMIZE
230 PRINT "    BASIC NUMBER
FACTS": : : :
240 PRINT "1. YEAR 1 — X TO
 9": :
250 PRINT "2  YEAR 1 — + TO
 10": :
260 PRINT "3. YEAR 2 — X TO
 3X9": :
270 PRINT "4. YEAR 2 — + TO
 9+9": :
280 PRINT "5. YEAR 2 — + 1S
T EXTENSION": :
290 PRINT "6. YEAR 3 — X TO
 7X9": :
300 PRINT "7. YEAR 3 — + 2N
D & 3RD                E
XTENSION": :
310 PRINT "8. YEAR 4 — X TO
 9X9": :
320 PRINT "9. EXTENDED X TO
12X12": :
330 INPUT "     WHICH LEVEL?
":LEVEL
340 IF (LEVEL>0)*(LEVEL<10)T
HEN 380
350 PRINT "    PROGRAM NOT A
VAILABLE.          TRY AGAI
N"
360 GOSUB 2070
370 GOTO 330
380 QUES=0
390 SCORE=10
400 CALL CLEAR
410 REM SELECT PARAMETERS FO
R LEVEL
420 ON LEVEL GOTO 430,500,57
0,700,760,920,1060,1220,1290
430 REM LEVEL 1
440 CALL SCREEN(11)
450 O$="X"
460 A=INT(RND*3)+2
470 B=INT(RND*3)+2
480 IF A*B>9 THEN 460
490 GOTO 1360
500 REM LEVEL 2
510 CALL SCREEN(10)
520 O$="+"
530 A=INT(RND*8)+2
540 B=INT(RND*8)+2
550 IF A+B>10 THEN 530
560 GOTO 1360
570 REM LEVEL 3
580 CALL SCREEN(8)
590 O$="X"
600 V=INT(RND*4)
610 W=INT(RND*8)+2
620 INV=INT(RND*2)+1
630 ON INV GOTO 640,670
640 A=V
650 B=W
660 GOTO 1360
670 A=W
680 B=V
690 GOTO 1360
700 REM LEVEL 4
710 CALL SCREEN(4)
720 O$="+"
730 A=INT(RND*8)+2
740 B=INT(RND*8)+2
750 GOTO 1360
760 REM LEVEL 5
770 CALL SCREEN(12)
780 O$="+"
790 V=INT(RND*8)+2
800 W=INT(RND*8)+2
810 IF V+W>9 THEN 790
820 X=INT(RND*9)+1
830 Y=X*10+V
840 INV=INT(RND*2)+1
850 ON INV GOTO 860,890
860 A=Y
870 B=W
880 GOTO 1360
890 A=W
900 B=Y
910 GOTO 1360
920 REM LEVEL 6
930 CALL SCREEN(15)
940 O$="X"
950 V=INT(RND*6)+2
960 W=INT(RND*8)+2
970 IF V*W<10 THEN 950
980 INV=INT(RND*2)+1
990 ON INV GOTO 1000,1030
1000 A=V
1010 B=W
1020 GOTO 1360
1030 A=W
1040 B=V
1050 GOTO 1360
1060 REM LEVEL 7
1070 CALL SCREEN(15)
1080 O$="+"
1090 V=INT(RND*7)+3
1100 W=INT(RND*7)+3
1110 IF V+W<10 THEN 1090
1120 X=INT(RND*9)+1
1130 Y=X*10+V
1140 INV=INT(RND*2)+1
1150 ON INV GOTO 1160,1190
1160 A=Y
1170 B=W
1180 GOTO 1360
1190 A=W
1200 B=Y
1210 GOTO 1360
1220 REM LEVEL 8
1230 CALL SCREEN(8)
1240 O$="X"
1250 A=INT(RND*8)+2
1260 B=INT(RND*8)+2
1270 IF A*B<20 THEN 1250
1280 GOTO 1360
1290 REM LEVEL 9
1300 CALL SCREEN(11)
1310 O$="X"
1320 A=INT(RND*9)+4
1330 B=INT(RND*9)+4
1340 IF A*B<40 THEN 1320
1350 GOTO 1360
1360 CALL VCHAR(1,2,32,352)
1370 CALL HCHAR(21,1,32,128)
1380 ERROR=0
1390 QUES=QUES+1
1400 REM RANDOMIZE FORMAT
1410 P=INT(RND*4)+1
1420 ON P GOTO 1440,1500,159
0,1440
1430 REM FORMAT #1
1440 PRINT A;O$;B;"= ?": :
1450 INPUT "ANSWER = ":ANS
1460 PRINT : :
1470 ON LEVEL GOTO 1490,1480
,1490,1480,1480,1490,1480,14
90,1490
1480 IF A+B<>ANS THEN 1780 E
LSE 1680
1490 IF A*B<>ANS THEN 1780 E
LSE 1680
1500 REM FORMAT #2
1510 IF A*B=0 THEN 600
1520 ON LEVEL GOTO 1550,1530
,1550,1530,1530,1550,1530,15
50,1550
1530 PRINT A;"+ ? =";A+B: :
1540 GOTO 1560
1550 PRINT A;"X ? =";A*B: :
1560 INPUT "ANSWER ? ":ANS
1570 PRINT : :
1580 IF ANS<>B THEN 1780 ELS
E 1680
1590 REM FORMAT #3
1600 IF A*B=0 THEN 600
1610 ON LEVEL GOTO 1640,1620
,1640,1620,1620,1640,1620,16
40,1640
1620 PRINT "? +";B;"=";A+B:
:
1630 GOTO 1650
1640 PRINT "? X";B;"=";A*B:
:
1650 INPUT "ANSWER ? ":ANS
1660 PRINT : :
1670 IF ANS<>A THEN 1780 ELS
E 1680
1680 CALL HCHAR(4,1,32,160)
1690 CALL HCHAR(10,20,96)
1700 CALL HCHAR(10,21,97)
1710 CALL HCHAR(11,20,98)
1720 CALL HCHAR(11,21,99)
1730 CALL SOUND(200,740,1)
1740 CALL SOUND(-200,784,1)
1750 IF QUES=10 THEN 2100
1760 ON LEVEL GOTO 460,530,6
00,730,790,950,1090,1250,132
0
1770 REM ERROR ROUTINE
1780 ERROR=ERROR+1
1790 GOSUB 2220
1800 CALL SOUND(200,784,1)
1810 CALL SOUND(-200,523,1)
1820 IF ERROR=2 THEN 1880
1830 PRINT " TRY AGAIN ": :
1840 GOSUB 2010
1850 SCORE=SCORE-1
1860 ON P GOTO 1450,1560,165
0,1450
1870 REM 2ND-ERROR ROUTINE
1880 GOSUB 2220
1890 CALL SOUND(200,392,1)
1900 CALL SOUND(-200,262,1)
1910 ON LEVEL GOTO 1940,1920
,1940,1920,1920,1940,1920,19
40,1940
1920 PRINT "  SAY";A;"+";B;"
=";A+B
1930 GOTO 1950
1940 PRINT "  SAY";A;"X";B;"
=";A*B
1950 CALL HCHAR(21,3,95,19)
1960 CALL VCHAR(22,2,37,3)
1970 CALL VCHAR(22,22,39,3)
1980 CALL HCHAR(24,3,95,19)
1990 GOSUB 2040
2000 GOTO 1750
2010 FOR M=1 TO 100
2020 NEXT M
2030 RETURN
2040 FOR M=1 TO 3000
2050 NEXT M
2060 RETURN
2070 FOR I=1 TO 500
2080 NEXT I
2090 RETURN
2100 PRINT
2110 PRINT "YOU HAD";SCORE;"
RIGHT OUT OF 10": : : :
2120 CALL HCHAR(21,2,45,29)
2130 FOR I=1 TO 10
2140 CALL SOUND(50,1000,1)
2150 CALL SOUND(50,20000,30)
2160 NEXT I
2170 PRINT "   TRY SOME MORE
?  Y OR N"
2180 CALL KEY(0,K,S)
2190 IF S=0 THEN 2180
2200 IF K=89 THEN 380
2210 IF K=78 THEN 210
2220 CALL HCHAR(4,1,32,160)
2230 CALL HCHAR(10,20,96)
2240 CALL HCHAR(10,21,97)
2250 CALL HCHAR(11,20,100)
2260 CALL HCHAR(11,21,101)
2270 RETURN
```

```
100 REM *************
110 REM ***FROGGER***
120 REM *FROM  ATICC*
130 REM *XB REQUIRED*
140 REM *************
150 REM
160 CALL CLEAR
170 CALL MAGNIFY(3)
180 CALL CHAR(60,"0000000018
33FFFFFFFC30180F040201192234
A89020F0FFFE081020E0201000")
190 CALL CHAR(136,"FFFFFFFFF
FFFFFFF")
200 CALL CHAR(36,"0000004021
1B1F071F77050B1B3244844042C4
CE8061C2FC0F05867203018042")
210 CALL CHAR(96,"010317151F
1F07CFDFDFFFFFFFFFE74380C0E8
A8F8F8E0F3FBFBFFFFFFFFE7C2")
220 CALL CHAR(100,"0101003C0
D3F6FFFFF6F3F0D3C000101FEFF3
EFCFCFEFFFFFFFFFEFCFC3EFFFE"
)
230 CALL CHAR(104,"7FFF7C3F3
F7FFFFFFFFF7F3F3F7CFF7F80800
03CB0FCF6FFFFF6FCB03C008080"
)
240 CALL CHAR(108,"43E7FFFFF
FFFDFDFCF071F1F15170301C2E7F
FFFFFFFFBFBF3E0F8F8A8E8C080"
)
250 A$=RPT$("182442814224180
0",4)
260 CALL CHAR(112,A$)
270 CALL CHAR(33,"FFFFFFFFFF
FFFFFF")
280 CALL CHAR(40,"00000000FF
FFFFFFFFFFFFFF00000000000000
00FFFFFFFFFFFFFFFF00000000")
290 CALL CHAR(116,"0000007C7
D117FFFFF7F117D7C000000001F1
F04FFFFFFFFFFFFFFFF041F1F00"
)
300 CALL CHAR(120,"0000031B3
B7BFFFFFFFF7B3B1B0300000000F
FFFFFFFFFFFFFFFFFFFFFF0000"
)
310 CALL CHAR(124,"00F8F820F
FFFFFFFFFFFFFF20F8F00000000
03EBE88FEFFFFFE88BE3E000000"
)
320 CALL CHAR(128,"070911374
99593FF9393954937110907E0100
8D4225191FF91915122D40810E0"
)
330 CALL SCREEN(15)
340 CALL COLOR(2,16,1)
350 CALL COLOR(1,4,1)
360 CALL COLOR(14,6,1)
370 CALL HCHAR(1,1,33,128)
380 FOR I=7 TO 25 STEP 9
390 CALL HCHAR(3,I,32,3)
400 CALL HCHAR(4,I,32,3):: N
EXT I
410 CALL HCHAR(17,1,45,32)::
 CALL HCHAR(20,1,45,32):: CA
LL HCHAR(23,1,45,32)
420 LIFE=4 :: HM=26 :: SP=10
 :: SD=0 :: SCORE=0 :: P=0
430 CALL HCHAR(11,1,33,96)
440 CALL HCHAR(5,1,136,192)
450 CALL SPRITE(#2,116,11,16
0,1,0,-10,#3,116,14,160,85,0
,-10,#4,116,7,160,170,0,-10)
460 CALL SPRITE(#5,124,14,13
6,70,0,SP,#6,124,5,136,155,0
,SP,#7,124,3,136,240,0,SP)
470 CALL SPRITE(#8,120,2,112
,1,0,-10,#9,120,9,112,85,0,-
10,#1,120,16,112,170,0,-10)
480 CALL SPRITE(#11,60,16,64
,238,0,5,#12,60,15,64,256,0,
5,#13,60,11,64,70,0,5)
490 CALL SPRITE(#14,60,11,40
,240,0,5,#15,60,16,40,210,0,
5,#16,60,16,40,70,0,5)
500 CALL SPRITE(#17,128,16,3
5,100,0,5,#18,128,12,64,110,
0,5)
510 IF START<>0 THEN 570
520 DISPLAY AT(8,12)SIZE(7):
"FROGGER" :: DISPLAY AT(11,5
)SIZE(22):"PRESS ANY KEY TO
START"
530 CALL KEY(0,K,S)
540 IF S=0 THEN 530
550 START=1
560 GOTO 420
570 IF SCORE>=2000 AND P=0 T
HEN LIFE=LIFE+1 :: P=1 :: CA
LL DELSPRITE(#6)
580 DISPLAY AT(1,2):USING "S
CORE ####":SCORE :: DISPLAY
AT(1,15):USING "FROGS LEFT #
":LIFE
590 IF LIFE=0 THEN 1030
600 ROW=184 :: COL=128
610 CALL SPRITE(#10,96,13,18
4,128):: CALL SOUND(100,1047
,1)
620 CALL JOYST(1,J1,J2)
630 CALL COINC(ALL,C)
640 IF C=-1 THEN 1130
650 IF ROW<32 THEN 900
660 IF ROW<80 THEN 950
670 IF (COL>256 OR COL<1)AND
 ROW<80 THEN 970
680 REM MOVE ROUTINE
690 X=J1/4+2
700 Y=J2/4+2
710 ON X GOTO 770,720,730
720 ON Y GOTO 840,620,810
730 FROG=104
740 COL=COL+16
750 IF COL>256 THEN COL=256
760 GOTO 880
770 FROG=100
780 COL=COL-16
790 IF COL<1 THEN COL=1
800 GOTO 880
810 FROG=96
820 ROW=ROW-16
830 GOTO 880
840 FROG=108
850 IF ROW>70 THEN CALL MOTI
ON(#10,0,0)
860 ROW=ROW+16
870 IF ROW>184 THEN ROW=184
880 CALL SPRITE(#10,FROG,13,
ROW,COL)
890 GOTO 620
900 REM CHECK POSITION ROUTI
NE
910 CALL POSITION(#10,ROW,CO
L):: CALL SPRITE(#10,96,13,R
OW,COL)
920 IF COL>47 AND COL<58 THE
N 1320
930 IF COL>119 AND COL<130 T
HEN 1320
940 IF COL>191 AND COL<202 T
HEN 1320
950 REM DROWN ROUTINE
960 CALL PATTERN(#10,112)::
CALL MOTION(#10,0,0)
970 CALL SOUND(1000,262,1,33
0,1,392,1)
980 GOSUB 1260
990 LIFE=LIFE-1
1000 IF ROW<88 THEN SCORE=SC
ORE+100
1010 IF SCORE<0 THEN SCORE=0
1020 GOTO 570
1030 REM END OF GAME
1040 IF SCORE>HIGH THEN HIGH
=SCORE
1050 CALL DELSPRITE(ALL)
1060 CALL HCHAR(5,1,32,192)
1070 DISPLAY AT(6,10)SIZE(9)
:"GAME OVER"
1080 DISPLAY AT(8,5):"PRESS"
;TAB(11);"0 TO STOP";TAB(11)
;"1 TO GO AGAIN"
1090 CALL KEY(0,K,S)
1100 IF S=0 THEN 1090
1110 IF K=49 THEN DISPLAY AT
(2,2)SIZE(15):USING "HIGH SC
ORE ####":HIGH :: GOTO 420
1120 IF K=48 THEN CALL CLEAR
 :: STOP ELSE 1090
1130 REM COLLISION BETWEEN S
PRITES
1140 IF ROW>90 THEN 1210
1150 IF ROW<35 THEN 950
1160 IF ROW<56 AND SD<>0 THE
N S=SD ELSE S=5
1170 CALL MOTION(#10,0,S)
1180 CALL JOYST(1,J1,J2)
1190 CALL POSITION(#10,ROW,C
OL)
1200 GOTO 670
1210 REM SQUASHED ROUTINE
1220 CALL PATTERN(#10,36)::
CALL MOTION(#10,0,0):: CALL
POSITION(#10,M,N):: CALL SPR
ITE(#10,36,7,M,N)
1230 CALL SOUND(1000,-7,0)::
 CALL SOUND(1000,40000,30)
1240 GOSUB 1260
1250 GOTO 990
1260 CALL SOUND(250,131,1)::
 CALL SOUND(50,40000,30):: C
ALL SOUND(250,131,1):: CALL
SOUND(50,40000,30)
1270 CALL SOUND(125,131,1)::
 CALL SOUND(50,40000,30):: C
ALL SOUND(250,131,1):: CALL
SOUND(50,40000,30)
1280 CALL SOUND(250,156,1)::
 CALL SOUND(50,40000,30):: C
ALL SOUND(125,147,1):: CALL
SOUND(50,40000,30):: CALL SO
UND(125,147,1)
1290 CALL SOUND(50,40000,30)
:: CALL SOUND(125,131,1):: C
ALL SOUND(50,40000,30):: CAL
L SOUND(250,131,1)
1300 CALL SOUND(50,40000,30)
:: CALL SOUND(125,123,1):: C
ALL SOUND(50,40000,30):: CAL
L SOUND(250,131,1)
1310 RETURN
1320 REM FROG HOME ROUTINE
1330 SCORE=SCORE+300
1340 CALL DELSPRITE(#10):: C
L=124
1350 IF COL>191 THEN CL=196
1360 IF COL<73 THEN CL=52
1370 CALL SPRITE(#HM,108,2,1
6,CL)
1380 GOSUB 1440
1390 HM=HM+1 :: IF HM<29 THE
N 570
1400 SP=SP+5 :: HM=26 :: SD=
SD+3
1410 CALL DELSPRITE(#26,#27,
#28):: CALL MOTION(#5,0,SP,#
6,0,SP,#7,0,SP,#14,0,SD,#15,
0,SD,#16,0,SD,#17,0,SD)
1420 IF SCORE>=3000 THEN CAL
L DELSPRITE(#5)
1430 GOTO 570
1440 REM FROG HOME SOUND
1450 CALL SOUND(250,262,1,33
0,1,393,1):: CALL SOUND(250,
262,1,330,1,393,1)
1460 CALL SOUND(250,262,1,34
9,1,440,1):: CALL SOUND(250,
262,1,349,1,440,1)
1470 CALL SOUND(250,262,1,39
2,1,494,1):: CALL SOUND(250,
262,1,349,1,440,1):: CALL SO
UND(500,262,1,330,1,393,1)
1480 RETURN
```

IMAGINATIVE PROGRAMMING

by Jim Peterson

Most of the fun (and frustration) of programming is in figuring out a way to do something you haven't done before. Self-taught programmers often devise some highly unconventional methods of solving problems, and this example contains some doozies!

This is a greatly improved version of a program which appeared in Tips from the Tigercub.

It is used to make a list of all program lines to which another line jumps, in order to avoid deleting those lines when modifying a program.

Since it reads a program saved into a D/V 163 file by SAVE DSK (filename),MERGE it requires a disk drive.

Line 130 lists most of the Extended Basic statements which can direct a jump. In order to keep the array at less than 10 subscripts, for reasons to be explained later, I omitted the rarely used GO, BREAK and UNBREAK.

Line 140 dimensions an array to be used to store the records, and reads the DATA into another array. Line 150 asks for the filename of the MERGE format program to be examined.

Note that the ACCEPT places the cursor after DSK, to make it plain that drive number and filename are to be entered. Line 160 opens the file and jumps over the error routine in 170.

If the filename is not on the disk, etc., line 170 prints the error message and goes back to give you another chance. ON ERROR STOP prevents an erroneous I/O ERROR message if some later error should occur.

Line 180 obtains your choice of output, with a default of P, and rejects anything other than a single character P or S.

If printer output is selected, line 190 asks for printer designation, opens printer.

Line 200 reads in a program line, using LINPUT rather than INPUT so as to accept any and all characters. ASCII 201 is the token code that always precedes a line number reference, so line 210 searches for ASCII 201, starting at the 3rd character so as not to look for it in the program line number itself. If POS=0 there are no jumps from this line, so to 260.

Else, line 220 converts the tokenized 2-byte line number, of the line being examined, to its decimal form in LN, and sets P to start search for ASCII 201 at the 3rd byte. In line 230, X will be the position of the first ASCII 201, and the 2 bytes following it will be the line number jumped to, which is converted to its decimal equivelant LREF. The search for the statement directing the jump will begin at S, one byte to left of ASCII 201.

In line 240, Z is the ASCII code found at position S. The POS statement lists the 3-digit ASCII codes of the words listed in line 130, in the same sequence. If Z matches one of them, (G+2)/3 gives the subscript number of GO$ for that word. Otherwise, if Z<100 the ASCII is not a token, or if G=0 then (G+2)/3 is less than 1, so the next search starts one byte to the left. If S becomes less than 3, the token being searched for must be that for GO, BREAK, or UNBREAK, which are not in the POS statement, so we will use the subscript 1 for GOTO instead.

Otherwise, go back and continue searching.

When the token is found, line 250 converts the "to" line number into a string followed by a period, followed by the string representation of the program line number followed by the string representation of the subscript number of GO$.

For instance, if line 100 was 100 GOTO 1000, C$ would be 1000.1001, which can be converted into a valid decimal 1000.1001 .

Then, the search will continue starting at the 3rd byte after the last ASCII 201 we found, the add-on variable C$ is cancelled before being reused, and we go back to see if the program line contains another jump. If not, IF X=0 THEN 260, where we check EOF to see if the end of the file is reached, otherwise go back to read in the next program line, until finished.

Now, the data in C() is in program line number sequence, but we want to list it in "to" line number sequence. That's the reason for the weird method of decimal storage. Note that tacking the subscript number, from 1 to 9, onto the end of the decimal prevented the trailing zeros from being dropped. If C$="1000.100" then VAL(C$) would equal 1000.1, but VAL("1000.1001") equals 1000.1001.

So now, a simple sorting routine arranges the data into "to" line number sequence and, within that, into program line number sequence and even within that into subscript number sequence!

Finally, lines 290-330 convert the C() array back into a string so it can be taken apart into its three components and printed. Note that if printer output was not selected in line 180-190, D was not given a value and still equals 0 (its value in subroutine LONGSHELLN is not passed back to the main program) and PRINT #0 causes output to the screen. o

```
100 DISPLAY AT(3,10)ERASE AL
L:"GO-SEARCH": :TAB(7);"by J
im Peterson"
110 DISPLAY AT(7,1):" For us
e before modifying a":"progr
am.":" Searches a program sa
ved in":"MERGE format, finds
 all":"lines containing a ju
mp,"
120 DISPLAY AT(12,1):"sorts
into 'to' line number":"sequ
ence, outputs to screen":"or
 printer."
130 DATA GOTO,GOSUB,THEN,ELS
E,RUN,RETURN,RESTORE,USING,E
RROR
140 DIM C(200):: FOR J=1 TO
9 :: READ GO$(J):: NEXT J ::
 A=1
150 DISPLAY AT(16,1):"FILENA
ME? DSK" :: ACCEPT AT(16,14)
:F$
160 ON ERROR 170 :: OPEN #1:
"DSK"&F$,INPUT ,VARIABLE 163
 :: GOTO 180
170 DISPLAY AT(18,1):"I/O ER
ROR" :: ON ERROR STOP :: RET
URN 150
180 DISPLAY AT(18,1):"OUTPUT
 TO P":" (S)creen":" (P)ri
nter" :: ACCEPT AT(18,11)SIZ
E(-1)VALIDATE("PS"):Q$
190 IF Q$="P" THEN DISPLAY A
T(22,1):"PRINTER? PIO" :: AC
CEPT AT(22,10)SIZE(-18):P$ :
: OPEN #2:P$ :: D=2
200 LINPUT #1:A$
210 IF POS(A$,CHR$(201),3)=0
 THEN 260
220 LN=ASC(SEG$(A$,1,1))*256
+ASC(SEG$(A$,2,1)):: P=3
230 X=POS(A$,CHR$(201),P)::
IF X=0 THEN 260 :: LREF=ASC(
SEG$(A$,X+1,1))*256+ASC(SEG$
(A$,X+2,1)):: S=X-1
240 Z=ASC(SEG$(A$,S,1)):: G=
(POS("13413517612916913614821
37165",STR$(Z),1)+2)/3 :: IF
 Z<100 OR G<1 THEN S=S-1 ::
IF S<3 THEN G=1 ELSE 240
250 C$=STR$(LREF)&"."&STR$(L
N)&STR$(G):: C(A)=VAL(C$)::
A=A+1 :: P=X+3 :: C$="" :: G
OTO 230
260 P=3 :: C$="" :: IF EOF(1
)THEN CLOSE #1 ELSE 200
270 DISPLAY AT(24,1):"SORTIN
G" :: A=A-1 :: CALL LONGSHEL
LN(A,C())
280 IF D=0 THEN DISPLAY AT(1
2,1)ERASE ALL:"ANY KEY TO PA
USE"
290 FOR J=1 TO A :: A$=STR$(
C(J)):: X=POS(A$,".",1):: Y=
VAL(SEG$(A$,LEN(A$),1)):: A$
=SEG$(A$,1,LEN(A$)-1)
300 PRINT #D:SEG$(A$,1,X-1);
TAB(7);GO$(Y);" FROM ";TAB(2
1);SEG$(A$,X+1,LEN(A$))
310 CALL KEY(0,K,ST):: IF ST
=0 THEN 330
320 CALL KEY(0,K2,ST2):: IF
ST<1 THEN 320
330 NEXT J :: END
340 SUB LONGSHELLN(N,NN())
350 D=N
360 D=INT(D/3)+1 :: FOR I=1
TO N-D :: IF NN(I)<=NN(I+D)T
HEN 390 :: T=NN(I+D):: J=I
370 NN(J+D)=NN(J):: J=J-D ::
 IF J<1 THEN 380 :: IF T<NN(
J)THEN 370
380 NN(J+D)=T
390 NEXT I
400 IF D>1 THEN 360
410 SUBEND
```

Loading Memory Image Files

by G.Meldrum

The term "memory image file" simply refers to a file, such as that stored on a disk or cassette tape, that is transferred in one big chunk to memory exactly as it was stored. On a disk catalogue these files are listed as PROGRAM type files.

A block of memory saved to cassette tape using the "S" command in Mini-Memory's Easy Bug is a PROGRAM type file. Also are the files that load in option 5 "RUN PROGRAM FILE" in the Editor/Assembler module. And naturally most of our BASIC programs are saved in this form.

When these programs are reloaded into the computer an area is set up in the console's vram memory to accept the incoming program file. The file is loaded provided that there is enough space in the vram buffer area to accept it. The interesting thing is that any PROGRAM file will successfully load into the buffer area regardless of whether it is correct for the module being used or not.

Let us take, for example, a short BASIC program and load it into the vram buffer using the "L" command of Easy Bug. Whilst in Easy Bug you are able to inspect the hex code of the BASIC program. You will find it starts at vram address >1100 and can be displayed by using the V1100 command. For those interested the PAB is at V1000.

If you try loading a 'non-BASIC' PROGRAM file using BASIC's "OLD" command then the complete file is loaded into a vram buffer before an error message is displayed. With a Mini-Memory or Editor/Assembler module it is possible to view the start of such a file using the "CHARPAT" command.

For example:

```
>OLD CS1
    :
 * DATA OK
    :
 * I/O ERROR 50

>CALL CHARPAT(128,A$,129,B$,130,C$,131,D$)
>DISPLAY A$,B$,C$,D$
 05000709000030CF
 6000001BBAA00004
 60B3102020434F50
 5952494748542031
```

The first line displayed plus the first two characters in the second line is the PAB; there follows the start of the input file proper. With disk files the start of the file will be a bit further along because of the extra length of the filename.

The first few bytes of the common memory image files have their own standard header information. Most of this is to do with relocating itself from the vram buffer to the appropriate memory location. By the 'common' files I refer to :

(1) BASIC files
(2) Editor/Assembler option 5 files
(3) Mini Memory Easy Bug files

>>>>>>> Header Information

(1) For TI BASIC and Extended BASIC
Example:

```
OC99  validity check
2660  end of line number table address
24F7  start of line number table address
3FFF  end of memory address
```

The first two bytes check for a valid BASIC file. It is calculated by taking an Exclusive OR value of the two values of the start and end of line number table addresses (refer XB manual p.44 for XOR description). If the program was saved using Extended BASIC's "PROTECTED" feature then the validity check is two's complemented (reverse bits then add 1). The next six bytes represent memory addresses of the program location as if it were stored in vram at the time it was saved. The 'end of memory address' is always >3FFF for cassette only users, and usually >37D7 for disk users, i.e. provided a CALL FILES command had not been executed. The length of a BASIC file can be calculated by:
(end of mem addr) - (start of line nr table addr) + 1

(2) For Editor/Assembler option 5 files
Example:

```
0000  more files flag
2000  length of file
A000  load address
```

If there were more files to follow this one then the 'more files flag' would have been >FFFF. Next there follows the length of file value which includes the six byte header. In this case the program is : >2000 - 6 = >1FFA long. And next is the load address to where the program code will be transferred. The first load address in a multi-file program is also the starting location of the program. Note that multi-file program files need not be loaded in a continuous block of memory. You can use the Editor/Assembler "SAVE" program to save a small block at say >A000. It is then possible to use the "SAVE" program again to save at >E000. With a disk fixer type program you can alter the first file's 'more files flag' to >FFFF. Naturally the filenames need to be identical with the last character of the second filename incremented.

(3) For files saved using the Mini Memory's Easy Bug's "S" command
Example:

```
7000  load address
1000  length of code
```

The length of code in this case excludes the four byte header.

The DEBUGGER and Extended BASIC

by G.Meldrum

The Debugger program is supplied on Disk A with the Editor/Assembler cartridge. Although this programs primary use was to aid in the debuging of machine code programs it contains features enduring to the budding hacker.

Although being able to view reams of ROM, GROM, and VRAM memory may not be everyones cup of tea, the ability to easily move great chunks of memory around opens up lots of possibilities. I frequently use this feature to implant machine code programs behind a BASIC program. You also have the ability to carry out hex arithmetic and alter values in (ram) memory.

The assembled form of the Debugger program is called DEBUG on the Editor/Assembler disk. Unfortunately it is in a compressed format which cannot be loaded from Extended BASIC. Fortunately however the source code for this program is also supplied on the same disk. This means you can easily assemble this program in an uncompressed format just by following these few steps :-

Continued on P18

1) Copy from the Editor/Assembler Disk A the files DEBUGA, DEBUGB, DEBUGC, DEBUGD, DEBUGE, DEBUGS to a new disk.

2) Insert the Editor/Assembler cartridge and select EDITOR/ASSEMBLER. Now select option 2 ASSEMBLE from the menu. Insert Disk A into drive 1 and answer "Y" to the 'LOAD ASSEMBLER (Y/N)?' prompt.

3) After the assembler is loaded insert your new disk with the DEBUGA/B/C/D/E/S files on it into drive 1 and type your answers to the prompts as listed below :-

```
* ASSEMBLER *
SOURCE FILE NAME?
DSK1.DEBUGS
OBJECT FILE NAME?
DSK1.DEBUGX
LIST FILE NAME?
<press enter>
OPTIONS?
R
```

You should end up with zero errors at the end of the assembly.

At this stage you can remove the Editor/Assembler cartridge and replace it with the Extended BASIC module. Select EXTENDED BASIC and load and run the Debugger program by entering the following commands:-

```
>CALL INIT
>CALL LOAD("DSK1.DEBUGX")
>CALL LINK("DEBUG")
```

To use the Debugger when entered from a BASIC environment it is necessary to press "U" to display a visible "." prompt. After the dot appears you can use any of the commands as listed in Section 23 of the Editor/Assembler manual. To return to BASIC simply type Q.

If you are familiar with the Editor/Assembler Editor then you may wish to modify the Debugger program so as to remove the need to press "U" when being called from BASIC. Three new lines of code need to be included in the file 'DEBUGB'.

```
       JGT  CLRMR        this is line 9
*
* insert the three new lines of code here
*
       LI   POINT,>6000
       MOV  POINT,@SCROS
       MOV  POINT,@TSCROS
*
       LWPI VREGS        etc.
```

Note that the syntax above is correct. Rather than use only R1, R2, R3, etc. labels for register values, TI chose more meaningful names. In the file 'DEBUGA' you can find the EQUates for these labels.

```
e.g.  WORDFG EQU 7
      POINT  EQU 8
```

Include the three extra lines of code before you assemble the program and you will find the message "*** 99/4 DEBUGGER ***" with a dot prompt appearing on the next line when using the program from BASIC. Oh well it saves pressing "U" anyway.

One last comment about the Debugger manual regarding the "M" command. It quotes that accessing GROM can alter the GROM program counter, preventing correct return to your program. So viewing GROM memory is ok provided you do not wish to return back to BASIC. It is possible to safely return to BASIC after viewing GROM memory however, if you take the following steps:-

just before you wish to exit the Debugger type :

(for Extended BASIC) M C41DG <enter> <enter> Q <enter>
(for E/A TI BASIC) M 6892G <enter> <enter> Q <enter>
(for M-M TI BASIC) M 6916G <enter> <enter> Q <enter>

FORUM

Q. I have an assembly language program which has to be loaded using editor assembler option 3. I can not remember the program entry point name, is there any way I can find the entry point name.

A. Yes this is easy. Load the assembly object file using the editor and examine the second last line. The entry points are found here.

Q. In some extended basic programs which contain assembly language the following statements appear:
CALL INIT ; CALL LOAD("DSK1.PROGRAM") ;
CALL LINK("START") ; what do these statements do?

A. CALL INIT loads routines from the extended basic Command Cartridge into the low RAM commencing from >2000. INIT allows linking between extended basic and assembly.

CALL LOAD("DSK1.PROGRAM") will load an assembly object file named PROGRAM from DSK1 into memory.

CALL LINK("START") will allow the computer to branch into the assembly program at the address of the label START. In the assembly source file START will appear in both a DEF statement and the label field.

Here is an example:

```
      DEF  START
START BLWP @0
      END
```

This little routine will return to the computer's master screen.

LETTERS TO THE EDITOR

The Editor,
TISHUG News Digest
12 October 1987

I would like to take this opportunity to say how pleased I was with the way the annual club auction was conducted on Saturday the 3rd of October at Burwood. Although I was able to sell some unwanted modules, and for that I am grateful, this is not the reason that I enjoyed it so much.

The previous auction that I attended was such a frustrating experience that I promised myself that it would also be my last. However, I let myself be talked into going to this year's auction, and what a pleasant surprise it was. Even if I had not sold a single item I would still have enjoyed it because of the manner in which it was conducted. Full marks to John Paine and Bert Thomas for conducting the auction in a very fair and friendly atmosphere. A large number of items were sold quickly and efficiently, and John did not let the proceedings drag out (as his predecessor did on the previous occasion).

I believe that most attendees benefited from the sale - either directly or indirectly.

I would, however, like to make two suggestions to improve the next auction:

Firstly, I believe that major items - ie those with a reserve of $50 or more - should be auctioned first, followed by all other items.

Secondly, avoid holiday periods so as to improve on the already very good attedance rate.

Thanks again to John and Bert. I am looking forward to next year's sale!

Lou Amadio
Illawarra Regional Group

DEBUGGING

by Jim Peterson

When you have finished writing a program, the next thing you should do is to run it. And, very probably, it will crash!

Don't be discouraged. It happens to the very best of programmers, very often.

So, the next thing to do is to debug it. And you are lucky that you are using a computer that helps you to debug better than some that cost ten times as much.

There are really three types of bugs. The first type will prevent the program from running at all - it will crash with an error message. The second type will allow the program to run, but will give the wrong results.

And the third type, which is not really a bug but might be mistaken for one, results from trying to run a perfectly good program with the wrong hardware, or with faulty hardware. As for instance, trying to run a Basic program, which uses character sets 15 and 16, in Extended Basic.

First, let's consider the first type. The smart little TI computer makes three separate checks to be sure your program is correct. First, when you key in a program line and hit the Enter key, it looks to see if there is anything it can't understand - such as a misspelled command or an unmatched quotation mark. If so, it will tell you so, most likely by SYNTAX ERROR, and refuse to accept the line.

Next, when you tell it to RUN the program, it first takes a quick look through the entire program, to find any combination of commands that it will not be able to perform. This is when it may crash with an error message telling you, for instance, that you have a NEXT without a matching FOR, or vice versa.

And finally, while it is actually running and comes to something that it just can't do, it will crash and give you an error message - probably because a variable has been given a value that cannot be used, such as a CALL HCHAR(R,C,32) when R happens to equal 0.

The TI has a wide variety of error messages to tell you when you did something wrong, what you did wrong, and where you did it wrong. But, it can be fooled! For instance, try to enter this program line (note the missing quotation mark). 100 PRINT "Program must be s aved in:"merge format."

And, sometimes you may be told that you have a STRING-NUMBER MISMATCH when there is no string involved, because the computer has tried to read a garbled statement as a string.

Also, the line number given in the error message is the line where the computer found it impossible to run the program; that line may actually be correct but the variables at that point may contain bad values due to an error in some previous line.

If the error occurs in a program line which consists of several statements, and you cannot spot the error, you may have to break the line into individual single-statement lines. This is the easiest way to do that - Be sure the line numbers are sequenced far enough apart. Bring the problem line to the screen, put a ! just before the first ::, and enter it. Bring it back to the screen with FCTN 8, retype the line number I higher, use FCTN 1 to delete the first statement and the ! and ::, put a ! before the first ::, and continue. Then, when you have solved the bug, just delete the ! from the original line and delete all the temporary lines.

Pages 212-215 of your Extended Basic manual list almost all the error codes, and almost all the causes of each one - it will pay you to consult these pages rather than guessing what is wrong.

You may create some really bad bugs when you try to modify a program that was written by someone else - especially if you add any new variable names or CALLs to the program. Your new variable might be one that is already being used in the program for something else, perhaps in a subscripted array. I have noticed that programmers rarely use @ in a variable name, so I always tack it onto the end of any variable that I add to a program.

Also, the program that you are modifying may have ON ERROR routines, or a prescan, already built in. The ON ERROR routine was intended to take care of a different problem than the one you create, so it could lead you far astray - you had better delete that ON ERROR statement until you are through modifying.

The prescan had better be the subject of another lesson, but if the program has an odd-looking command !@P- up near the front somewhere, it has a prescan built in. And if so, if you add a new variable name or use a CALL that isn't in the program, you will get a SYNTAX ERROR even though there is no error. One way to solve this is to insert a line with !@P+ just before the problem line, and another with !@P- right after it.

When a program runs, even though it crashes or is stopped by FCTN 4 or a BREAK, the values assigned by the program to variables up to that point will remain in memory until you RUN again, or make a change to the program, or clear the memory with NEW. This can be very useful. For instance, if the program crashes with BAD VALUE IN 680, and you bring line 680 to the screen and find it reads CALL HCHAR(R,C,CH) just type PRINT R;C;CH and you will get the values of R, C and CH at the time of the crash. You will find that R is less than 1 or more than 24, or C is less than 1 or more than 32, or CH is out of range.

In Extended Basic, you can even enter and run a multi-statement line in immediate mode (that is, without a line number), if no reference is made to a line number. So, you can dump the current contents of an array to the screen by FOR J=1 TO 100::PRINT A(J);: : NEXT J - or you can even open a disk file or a printer to dump it to.

You can also test a program by assigning a value to a variable from the immediate mode. If you BREAK a program, enter A=100 and then enter CON, the program will continue from where it stopped but A will have a value of 100.

You can temporarily stop a program at any time with FCTN 4, of course (the manual says SHIFT C, but it was written for the old 99/4), and restart it from that point with CON. Or you can insert a temporary line at any point, such as 971 BREAK if you want a break after line 970. Or, you can put a line at the beginning of the program listing the line numbers before which you want breaks to occur, such as 1 BREAK 960,970,980 Note that in this case the program breaks just BEFORE those listed line numbers. You can also use BREAK followed by one or more line numbers as a command in the immediate mode.

The problem with using BREAK and CON is that BREAK upsets your screen display format, resets redefined characters and colors to the default, and deletes sprites. So, it is sometimes better to trace the assignment of values to your variables by adding a temporary line to DISPLAY AT their values on some unused part of the screen. If you want to trace them through several statements, it will be better to GOSUB to a DISPLAY AT. And if you need to slow up the resulting display, just add a CALL KEY routine to the subroutine.

Continued on P20

Sometimes, your program will appear to be not flowing through the sequence of lines you intended (perhaps because it dropped out of an IF statement to the next line!) and you will want to trace the line number flow. This can be done with TRACE, either as a command from the immediate mode or as a program statement, which will cause each line number to print to the screen as it is executed. If used as a command, it will trace everything from the beginning of the program, so it is usually better to insert a temporary line with TRACE at the point where you really want to start. Once you have implemented TRACE, the only way to get rid of it is with UNTRACE.

TRACE has its limitations because it can't tell you what is going on within a multistatement line, and it will certainly mess up any screen display. Sometimes it is better to insert temporary program lines to display line numbers. I use CALL TRACE( ) with the line number between the parentheses, and a subprogram after everything else 30000 SUB TRACE(X)::DISPLAY AT(24,1):X :: SUBEND

Some programmers use ON ERROR combined with CALL ERR as a debugging tool, but I can't tell you much about that because I have never used it. ON ERROR can give more trouble than help if not used very carefully, and I cannot see that CALL ERR gives any information not available by other means.

Sometimes you can debug a line by simply retyping it. It is only very rarely that the computer is actually interpreting a line differently than it appears on the screen, but retyping may result in correcting a typo error that you just could not see. In fact, most bugs turn out to be very simple errors.

When you are debugging a string-handling routine, don't take it for granted that a string is really as it appears on the screen - it may have invisible characters at one or both ends. Try PRINT LEN(M$) to see if it contains more characters than are showing; or PRINT "*"&M$&"*" to see if any blanks appear between the asterisks and the string.

There is no standard way to debug a program. Each problem presents a challenge to figure out what is going wrong, to devise a test to find out what is really happening.

Don't debug by experimenting, by changing variable values just to see what will happen, etc. Even if you succeed, you will not have learned what was wrong so you will not have learned anything - and if your program contains lines that you didn't understand when you wrote them, you will have real problems if you ever try to modify the program. (Believe me, I speak from experience!)

o

Continued from P7

Members of TIsHUG pay an additional $5 membership fee to gain access to the BBS. All requests for BBS membership should be sent to the SECRETARY TIsHUG, PO Box 214, Redfern. N.S.W. 2016. You should nominate your preferred user name (a nickname is usually suitable). The minimum system for BBS access is a TI99/4A, TV, TE2 module, RS232 card or stand alone and a Modem, with ready access to your telephone line. Other computers can be used for access to the file areas but memory image basic and extended basic programs can only be downloaded to a TI99/4A.

To assist new members, and sometimes older members, files called BBS_HELP and SENDMAIL are provided which give information on how to best use the facilities of the BBS. The BBS provides menus which assist users in all aspects of operation.

o

PUTTING IT ALL TOGETHER #2

by Jim Peterson

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

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

Your computer won't take that 6th line in line 110? Just type 5 full lines and enter, bring it back by typing 110 and FCTN X, use FCTN D to scoot the cursor to the end of the line, and type some more.

```
1 !      2-LINE GAME
   by Jim Peterson
- use S&D keys to paint the
white line on the highway
100 CALL CLEAR :: A$=RPT$(CH
R$(143),6):: CALL COLOR(14,2
,2,2,16,16):: CALL SCREEN(4)
:: T=11 :: C=14 :: CALL HCHA
R(22,C+2,42):: RANDOMIZE
110 T=T+INT(3*RND-1)+(T=21)-
(T=1):: PRINT TAB(T);A$ :: C
ALL KEY(3,K,S):: C=C+(K=83)-
(K=68):: CALL HCHAR(22,C+2,4
2):: IF C<T OR C>T+5 THEN ST
OP ELSE 110
```

This is not a finished program, but an example of the ways that efficient programming can accomplish a great deal in very little memory. The screen is cleared and A$, which will be the highway, is defined as ASCII 143 repeated 6 times. A single CALL COLOR is used to color set 14 (the highway) black on black and set 2 (the painter) white on white. T sets the first line of the highway to begin at TAB 11 and C places the painter 3 spaces to the right, in the middle of the highway. CALL HCHAR places the painter on row 22 and column C+2 because an HCHAR column is 2 spaces to the left of a TAB or PRINT column. RANDOMIZE makes a different highway for each game.

INT(3*RND) gives a random value of 0, 1 or 2; subtracting 1 from this gives -1, 0 or 1, so the tab position for the next line of the high- way shifts one space left or right or, if 0, remains the same. +(T=21)-(T=1) uses relational values. If tab is already at 21, adding one would cause the 6-line road to print one line lower and at the left of the screen. So, if T=21 then (T=21) has a true relational value of -1; +(-1) = -1, so 1 is subtracted to keep the tab from going beyond 21. If the tab is already at 1, (T=1) has a true value of -1; -(-1) = +1, so 1 is added to keep tab from reaching 0. If T is not 21 and T is not 1, both have a false value of 0 and no change is made.

A line of the highway is printed, and CALL KEY looks for a keyboard input; the mode 3 accepts any input as upper case even if the alpha lock is up.

C=C+(K=83)-(K=68) is another example of the use of relational values for compact programming. K is the ASCII value of the key that was pressed; 68=D and 83=S. If S was pressed then C=C+(-1)-(0) and C=-1-0 and the painter moves one space left. If D was pressed, C=C+(0)-(-1) and C=C+0+1 and the painter moves right; if no key was pressed (K will equal -1) or any other key was pressed, C=C+(0)-(0).

So, the new position of the painter is printed by HCHAR; if it is less than the current tab position or more than 5 spaces to the right of tab, he is off the road and crashes; otherwise execution goes back to calculate the next ran- dom tab position and start over.

And all that in two lines of programming!

Now, what two values could you change to make the game more challenging? Try changing the 6 to a 5 in A$=RPT$(CHR$(143),6) and the 5 to a 4 in C>T+5. How could you offer the option of an easy or difficult game? How could you restart after a crash? Improve the graphics?

o

SPRITES PART I

by Jim Peterson

The sprites of TI Extended Basic are mostly used in fast-action arcade-type games, but they have other uses as well.

Up to 28 sprites can be placed on the screen at one time, but there is one very serious limitation - if more than 4 of them are in a line horizontally, only the 4 lowest-numbered ones will be visible. That is why, if you have numerous sprites moving about the screen, one of them will occasionally disappear and reappear, or a horizontal slice of a magnified sprite will become transparent.

A sprite is placed on the screen by the statement CALL SPRITE(#N,ASC,COL,DOTROW,DOTCOL)

N is the sprite number, between 1 and 28, and it must be preceded by the # sign. ASC is the ASCII code of the character that you wish the sprite to have. It must be between 32 and 143 - the ASCII characters 33 through 126 are the keyboard characters, the others will be blank unless you redefine them. COL is the color you wish the sprite to have, using the same color codes, 1 to 16, as are used for CALL SCREEN or CALL COLOR.

DOTROW and DOTCOLUMN are the dot row and dot column at which you wish the sprite to appear. You know that the monitor screen consists of 24 rows and 32 columns. Using HCHAR or VCHAR, you can place a character on any one of those 768 spaces (PRINT and DISPLAY start at column 3 of the graphics screen). Each of those spaces consists of a grid of 8 x 8 dots, totaling 64. By turning various of those dots off (blank) or on (colored), a character is displayed on the screen. Therefore the screen is 8 x 32 or 256 dotcolumns wide and the visible screen is 8 X 24 or 192 dotrows deep. Actually dotrow can be anything up to 256; dotrows 193 through 256 are hidden below the bottom of the screen, and sprites can be hidden there.

The upper left hand corner of your sprite will be at whatever dotrow and dotcolumn you specify.

To convert an graphics screen (HCHAR) position into dotrow and dotcolumn, use DOTROW=ROW*8-7 and DOTCOL=COL*8-7; to convert a PRINT/DISPLAY position, you must use DOTCOL=(COL+2)*8-7.

So, CALL SPRITE(#1,42,16,89,121) will place sprite #1, in the form of the asterisk (ASCII 42), colored white (16) in the middle of the screen. If you want, you can give it motion when you create it, by giving it a row-velocity and a column-velocity. These velocities can be from -128 to 127. A positive row velocity moves the sprite down, negative moves it up; a positive column velocity moves it right, negative moves it left.

Velocity 0 is a standstill,and speed increases from 1 upwards and from -1 downwards.

So, CALL SPRITE(#1,42,16,89,121,5,5) will place that white asterisk in the middle of the screen and start it moving slowly at a 45 degree angle downward to right (since the values 5 and 5 are positive and equal). It will continue moving at that direction and speed until you tell it to do otherwise, all by itself and without program control. When it reaches the right edge of the screen, it will "wrap around" and appear at the left. When it reaches the bottom, it will disappear briefly while it passes through those hidden dotrows, and "wrap around" to appear at the top.

If you want to change the pattern of the sprite, there are three ways to do so. You can CALL SPRITE again with the same sprite number but a different ASCII character - but if the existing sprite is not in the position of the dotrow and dotcolumn you specify, it will disappear and reappear in the new position. Or you can reidentify a character by CALL CHAR, and any sprite having that character will change accordingly, without affecting its color, position or movement. Or you can use CALL_PATTERN(#N,ASC) to change the pattern of sprite #N to the pattern of the specified ASCII character, without affecting color, position or motion.

There are also two ways to change the color of a sprite. CALL SPRITE with the same sprite number and ASCII but a different color code will recreate the sprite with the new color, but in whatever position is specified. CALL COLOR(#N,COLOR) will recolor sprite #N to the specified color code without affecting its pattern, position or motion.

If you want to change the position of a sprite, CALL LOCATE(#N,DOTROW,DOTCOL) will make it disappear at its old location and appear at the new location. The pattern and color will be unchanged, and if it was in motion the same motion will continue from the new position.

To change the motion of a moving sprite, or to start a stationary sprite into motion or vice versa, use CALL_MOTION(#N,RV,CV) - RV and CV being the same row velocity and column velocity optionally used in CALL SPRITE. CALL MAGNIFY will change the size of your sprite. You do not specify a sprite number with this CALL, because it affects all sprites that are on the screen or are subsequently placed on the screen. CALL MAGNIFY(2) enlarges the sprite 4 times so that it fills 4 of the graphic screen spaces, 256 dot spaces. CALL MAGNIFY(3) causes the sprite to consist of 4 characters, occupying 4 graphic screen positions. The upper left of these characters will be the ASCII specified in the CALL SPRITE or CALL PATTERN, provided that the ASCII is evenly divisible by 4 - otherwise, it will be the next smaller ASCII evenly divisible by 4. The next higher ASCII will be in lower left, the next in upper right, the next in lower right. In other words, if you use CALL MAGNIFY(3) and CALL SPRITE(#1,64,2,10,10) you will get a sprite looking like this - @B
AC

- and if you CALL SPRITE(#1,65,2,10,10) you will get exactly the same thing, because the computer will substitute the next lower number, 64, which is evenly divisible by 4.

Naturally, you will not have much use for sprites consisting of four characters, unless you redefine them into a single pattern, and in that case you must remember that they will appear in that upper left/lower left/upper right/lower right sequence. Fortunately, there are sprite editor programs to take care of this for you.

CALL MAGNIFY(4) will enlarge that 4-character sprite so that it fills 16 graphic screen positions. Note that magnification options 2 and 4 actually enlarge each dot to fill 4 dot positions, so that the sprites have a more angular, blocky appearance.

And finally, CALL MAGNIFY(1) will return magnified sprites to their normal single-space size.

Programming with sprite motion is unlike any other programming, because you do not control the program execution step-by-step. When you set a sprite in motion, it continues in motion while the program goes on to do whatever it is supposed to do next. When you want to control the sprite again, you must catch up with it and find out where it is. There are three ways to do this.

Continued on P24

EDITOR/ASSEMBLER TIPS- AUTO START.
From Mack McCormick

Have you ever wanted to have an object code assembly language program execute automatically saving the user a few keystrokes? Most folks already know if you add the label of your program entry point to the END statement that the program will execute automatically using the LOAD AND RUN options of Editor/Assembler or Mini-Memory. Right? We usually refer to this as an end start. Suppose you want it to execute automatically from E/A BASIC or Extended Basic. How would you do that without a CALL LINK statement? As you know the Extended Basic loader does not recognize an end start tag but will read it in without an unrecognized tag error. After the CALL LOAD statement executes it goes back to the GPL workspace register R11 to find out where to go next. Whatever address you place in this register is where the program will return. The GPLWS is located at >83E0 in scratchpad RAM inside the 9900 CPU chip. Just AORG the program counter to >83E0+22 and follow it with a DATA statement containing your program entry point. Follow with an END START and the program will execute automtically from any environment.
Example:

```
ENTRY     LWPI MYREG

<YOUR PROGRAM HERE>

          AORG >83E0+22  THIS IS THE ADDRESS
                         FOR GPLWS R11

          DATA ENTRY

          END  ENTRY
```

★ ★

HOW TO REDEFINE THE SHAPE OF THE CURSOR IN EXTD BASIC.

by Ross Mudie of TIsHUG.

The cursor in extended basic can be redefined by using CALL COLOR(0,F,B), since it is in colour set zero. The CHAR subprogram however will not allow access to the cursor which is ascii character 30.

This small assembly routine will allow you to define a new pattern for the cursor in the line with the label CURPAT. The DATA uses the same pattern information as is explained in the User's Reference Guide in section II, pages 76 to 79.

The assembly information provided is the source file. Type the source file using the editor/asembler editor, then save the file with the file name DSK1.CSR . Load the Assembler. Use DSK1.CSR for the source file name and DSK1.CURSOR for the object file name. Press enter for the list file name and use R for the options.

Go to extended basic and type: CALL INIT :: CALL LOAD("DSK1.CURSOR") :: CALL LINK("CURSOR") <enter>.

Your extended basic cursor will now be redefined until you leave extended basic or turn the computer off.

```
        IDT  'CURSORmu'              source CSR  16.11.85
        DEF  CURSOR                        object CURSOR
VMBW    EQU  >2024

CURPAT  DATA >0018,>2442,>8142,>2418      Diamond cursor
*       You can design your own cursor pattern, refer to
*                    User's Reference Guide page II-76.
CURSOR  LI   R0,1008     VDP address of char table entry
        LI   R1,CURPAT      Where to get the new pattern
        LI   R2,8    Number of bytes to write to VDP RAM
        BLWP @VMBW                              Do it !
        RT                        Return to extended basic

        END
```

Another short article on MODULES.

This article is on TERMINAL EMULATOR II and how to get a bit of extra text without getting hung up between Control S and Control Q.

The Terminal Emulator II actually has a continuous buffer that holds about 3 full screens.

To recycle the stuff, you have got to get the other system to stop sending. If it does not obey a Control S, get about 3 screens and then do an ESC. It should give a selection menu but if ESC will not do it, sometimes Control C works. Control 1 and Control 7 will stop some BBSs. If necessary, carefully pull the MODEM out. You can still save to tape. The Function Up Arrow will now give access to the "lost" material, so you can print it screen by screen, using the Function Up Arrow to scroll it around to the postition that you want to print, (or save to tape if you pulled the MODEM out of the computer, taking care to pull it STRAIGHT out avoiding a RESET) now with practice you can save or print the text JUST THE WAY YOU WANT IT without having to get the system to re-send fore you have to an ESC or a Control C and get the rest of the PAMS or whatever text you are trying to dig out of the BBS you have logged into!

D.N.Harris

★ ★

Continued from P23

CALL COINC(ALL,C) will give a value of -1 to C if any two sprites on the screen are overlapping, even slightly, or 0 if they are not. CALL COINC(#1,#2,TOL,C) will give C a value of -1 if the upper left hand corners of sprites #1 and #2 are within TOL dotrows and dotcolumns of each other. TOL may be any number you want, depending on whether you want to catch them only when they are right on top of each other, or just getting close. If not within tolerance, C will equal 0.

CALL COINC(#1,DOTROW,DOTCOL,TOL,C) will give C a value of -1 if the upper left corner of sprite #1 is within TOL dotrows and dotcolumns of the specified DOTROW and DOTCOL.

CALL COINC is not foolproof. If you give the sprites a fast motion, a coincidence may not be caught. And when you alternate your CALL COINC with other statements such as CALL JOYST, a coincidence will be missed if the program is executing some other statement at the time.

CALL POSITION(#N,DOTROW,DOTCOL) will give the dotrow and dotcolumn that the upper left corner of the sprite is occupying at the instant it is called. This one again is not foolproof because the sprite will have moved from that position before another statement can be executed to do anything with the information.

CALL DISTANCE(#1,#2,D) or CALL DISTANCE(#1,DOTROW,DOTCOL,D) will give to D a value depending on the distance between the two sprites, or between the sprite and the location. The value, as I understand it, is the square root of the total of the squares of the difference between the dotrows added to the squares of the differences between the dot columns. I'm not sure how useful all that is, and I have rarely seen this CALL used by programmers.

Finally CALL DELSPRITE(#N) will delete sprite #1 from the screen and CALL DELSPRITE(ALL) will delete them all.

Those are just the basics of sprite programming. What can be done depends solely on your ingenuity. o

Programs That Write Programs

Part 3

Jim Peterson

Let's start learning how to actually write a program that writes a program.

A MERGEd program is a D/V 163 file, so - OPEN #1:"DSK1.(filename),VARIABLE 163,OUTPUT

Every program line begins with a line number, of course. In MERGE format the line number, whether 1 or 32767, is squished into two characters. We don't need to get into how this is done, but you can accomplish it with CHR$(INT(LN/256))&CHR$(LN-25 6*INT(LN/256)), where LN has ben predefined as the line number.

To print a statement or command, anything that is represented by a token in the token list, just print the CHR$ of its token ASCII. For instance, the token for DATA is 147, so you would print CHR$(147).

To print a variable name, either numeric or string, just enclose it in quotes, "A" or A$".

To print a value, or a string which is not in quotation marks (such as in a DATA statement), or the word which follows a CALL, you must print CHR$(200) followed by a token giving the number of characters to follow, such as CHR$(5) for a 5-letter word such as CLEAR, then the value in quotes. For instance, the token for CALL is 157, so CALL CLEAR is CHR$(157)&CHR$(200)&CHR$(5)&"CLEA R" . Similarly, tokens for paren- theses are 183 and 182, so the variable name A(1) is "A"&CHR$ (183)&CHR$(200)&CHR$(1)&"1"&CHR$(182).

A quoted string is handled in the same way except that it is preceded by token 199, so PRINT "HELLO" is CHR$(156)&CHR$(199) &CHR$(5)&"HELLO". Don't worry about the quotation marks, the computer will handle that.

If you need to refer to a line number, as in GOTO 500, use token 201 followed by the line number formula, thus CHR$(134)&CHR$(201) &CHR$(INT(500/256))&CHR$(500-256* INT(500/256)).

Don't print more than 163 characters in a record. You can print multiple-statement XBasic lines, but be sure to use the double-colon token 130 as the separator, not two of the 181 colon tokens.

Each program line must end with CHR$(0) as the end-of-line indicator, and the last record you print must be CHR$(255)&CHR$ (255) as the end-of-file indicator.

If you get an I/O ERROR 25 when you try to merge your program, it means that you left off the final double-255. If the program merges, but crashes when you run it, you will probably be able to spot an obvious error in the line when you LIST it. If the line looks OK but gives you a DATA ERROR or SYNTAX ERROR, you left off a CHR$(0) or gave the wrong count of characters after token 199 or 200. The program published in Part 2 will help you to track down these bugs.

Now let's write a program. What is the longest possible one-liner program?

Well, RANDOMIZE is the longest statement that can stand alone. It is represented by the single token 149, and to repeat it must be followed by the double-colon token 130. Since any line number will take two bytes, let's use a 5-digit line number. And don't forget that final CHR$(0). That still leaves us 160 of the 163 bytes, so we can repeat tokens 149 and 130 for 79 times, followed by a final 149.

```
100 OPEN #1:"DSK1.LONG",VARI
ABLE 163,OUTPUT
110 FOR J=1 TO 79 :: M$=M$&C
HR$(149)&CHR$(130):: NEXT J
:: M$=CHR$(254)&CHR$(254)&M$
&CHR$(149)&CHR$(0):: PRINT #
1:M$ :: PRINT #1:CHR$(255)&C
HR$(255)
120 CLOSE #1
```

RUN, NEW, MERGE DSK1.LONG and LIST - over 34 lines long! But that one-liner doesn't do anything, so try this one -

```
100 OPEN #1:"DSK1.LONG",VARI
ABLE 163,OUTPUT
110 FOR J=1 TO 52 :: M$=M$&C
HR$(162)&"X"&CHR$(130):: NEX
T J :: M$=CHR$(254)&CHR$(254
)&M$&CHR$(162)&"X"&CHR$(0)::
 PRINT #1:M$
120 PRINT #1:CHR$(255)&CHR$(
255):: CLOSE #1
```

Again RUN, enter NEW, then MERGE DSK1.LONG, then RUN. You'll get a message BREAKPOINT IN 32510 (don't ask me why!) but just enter RUN again.

Well, if you have tried your hand at any MERGE format program writing, you have already discovered that it is slow work, and you need to cram more onto a line than will fit. When a little CALL HCHAR(24,12,32,5) turned into CHR$(157)&CHR$(200)&CHR$(5)&"HCHAR"&CHR$ (183)&CHR$(2 00)&CHR$(2)&"24"&CHR$(179)&CHR$(200)&CHR $(2)&"12"&CHR$(179)&CHR$(200)&CHR$(2)&"32"&CHR$(179)&C HR$(200)&CHR$(1)&"5"&CHR$(182) you gave up? There is an easier way! Using DEF can make the job so simple that you might decide to do all your programming in MERGE format - well no, it's not quite that easy.

The DEF does slow up program execution time considerably, especially when DEFs call each other, but we can tolerate that here.

For instance, that complicated mess of parentheses to squish a line number can be written just once as DEF LINES$(X)=CHR$(INT(X /256))&CHR$(X-256*INT(X-256)) and then, whenever you need a line number, just write LINE$(100) or whatever.

The flag token and counting of characters and all for an unquoted string can be DEF'd as U$(X$)=CHR$(200)&CHR$(LEN(X$))&X$ Then, to write "HELLO" just write U$("HELLO") and let the computer do the work. For a numeric value in the unquoted string, use UN$(X)=CHR$(200)&CHR$ (LEN(STR$(X)))&STR$(X), and then 999 becomes UN$(999).

CALL HCHAR can be DEF HCHAR$=C HR$(157) for CALL and, since one DEF can call another, U$("HCHAR") and, since it is always followed by an opening parentheses, CHR$(183) - but wait, let's define that open parentheses as OP$=CHR$(183). Now DEF HCHAR$=CHR$(157)&U$("HCH AR")&OP$, and you can use HCHAR$ for CALL HCHAR(.

Let's also DEF the comma with DEF C$=CHR$(179) and the closing parentheses with DEF CP$=CHR$(182). Now that long HCHAR that had you discouraged can be abbreviated to CHAR$&UN$(24)&C$&UN$(12)&C$&UN$(32)&C$&UN$ (5)&CP$.

I have written a program of 162 of these DEFs, and another program to print out a handy look-up chart of them. It would take 4 pages to print them, so if you want them just ask me for a copy. o

THOSE CONFUSING VARIABLE NAMES

by Jim Peterson

A variable name is just a name, a symbol, a "handle", which represents a number and is used in place of the number in a program. So, why not just use the number itself? Here's why -

```
100 FOR J=1 TO 20
110 PRINT J
120 NEXT J
```

Line 100 tells the computer to make the variable name J represent 1 the first time it reads the line, and to add 1 to that value each subsequent time until it passes 20. Line 110 tells the computer to print the current number represented by J. Line 120 says to go back and do it over again, until the 20 loops have been completed. So, in 3 program lines we have told the computer to print the numbers from 1 to 20. That is easier than -

```
100 PRINT "1"
110 PRINT "2"
and so on for 20 times!
```

To continue, a string variable name is just a name, a symbol, a "handle", used to represent a sentence, a word, a group of letters, a single letter, or even a group of numerals not being used as a number.

Some beginning programmers get the idea that certain variable names must be used for specific purposes, such as IR for "interest rate" and PR for "principal", etc. Not so! You could just as well use X, XY, WWWW or @7.

A variable name can be up to 15 characters long, if you want to use up that much memory. Unlike other home computers, the 99/4A will read and use the entire name (not just the first four characters). However, the variable name must begin with a letter, the "at" symbol (@), the left or right bracket ([]),backslash (\) or the underline (_). For the remainder you can use letters, numbers, (@) or (_). If you try to use lower case letters, the computer will change them to upper case. The variable name cannot contain a blank space, and it cannot be a word which is already part of the Basic language, such as LIST or RUN, etc. And, a string variable name must end with the dollar sign ($). Those are the rules, and if you break them the computer will call you a BAD NAME! (only in Basic, if the name is too long, otherwise a different ERROR is given.)

It will also give you an ERROR message if you copy a program from another BASIC dialect which allows the same name to be used for a simple variable and a subscripted variable. We won't get into that, but if the program has something like C=1 and C(1)=2, just change that plain C to C@. Finally, the computer will also insult you if you try to run a Basic program in Extended Basic and the programmer has used a variable name which happens to be a part of the Extended Basic language, such as MERGE.

Some programmers like to use complete words as variable names, i.e. INTEREST=RATE*PRINCIPAL. Maybe that does make the program easier to understand, but it uses up a lot of memory.

Other programmers like to use abbreviated words. That saves memory, but it can result in bugs crawling all over your program if you abbreviate PRINCIPAL as PRN the first time and as PRIN the second time.

Still others prefer single letter or two-letter variables, assigned arbitrarily. This works just fine, if you keep a piece of scratch paper handy to record what you used for what. Some programmers start with A for the first name they need and work through the alphabet. Others start with X, Y and Z, and then take off in all directions.

It really doesn't make any difference. BUT - with 24 other letters and hundreds of combinations available, why do programmers insist on using I and O? They are so easily mistaken for 1 and 0, especially in the tiny print of magazine listings or the fuzzy print of mimeographed newsletters! Come to think of it, I also have trouble with B and 8.

It is probably a good idea to get in the habit of using certain letters for certain purposes, such as R and C for row and column, K for key, J for loops, D for delay loops, etc., and then don't use those letters for anything else.

I don't normally use the (@) symbol in programs, but I try to include it in variable names when I write a utility subroutine or subprogram which I will save to merge into other programs. I also use the (@) in temporary debugging routines, in modifying other people's programs, etc. That way, I can be fairly sure that I'm not duplicating a name that is already used in the program.

* *

This file was provided by Steven, user name SUS in response to the following question on compound interest.

I NEED HELP WITH A FORMULA FOR MULTIPLAN TO WORK OUT COMPOUND INTEREST. COMPOUNDING PERIODS A YEAR OF 365 TIMES 12 TIMES AND 4 TIMES, FOR 1 TO 10 YEARS IS THERE ANY ONE WHO CAN ASSIST?

The formula is:

A=P(1+R)^N

A=the final amount ie principal + interest
p=principal, what you put in
R=rate, as a percentage
N=number of interest recalculations, NOT number of years
eg: $10000 invested at 11%pa for 5 years compounded annually:
A=10000x(1+0.11)^5 = 10000x(1.11)^5 = $16850.58

NOTES: 1) ^5 means raise to the fifth power ie 7^2 is 7 squared, 49, and 3^5 is 3 to the fifth power which equals 243.
2) ONLY the (1+R) is raised to the Nth power, NOT the Principal.
3) the Rate must be expressed as a decimal ie 0.11 and NOT 11
4) if you want interest calculated quarterly or daily of then follow these steps:

a) divide the per-annum interest rate by the number of recalculations ie. 4 times for quarterly, 365 times for daily etc
b) multiply the number of years invested by the number of recalculations
c) use a) for Rate and b) for Number of terms

eg Calculate $10000 invested at 11%pa for 5 years with interested calculated:
a) yearly b) twice a year c) quarterly d) monthly
e) daily

a) A=10000(1.11)^5 =$16850.58
b) a=10000(1+[0.11/2])^(5*2)
=10000(1.055)^10
=$17081.44
c) A=10000(1+[0.11/4])^(5*4)
=10000(1.0275)^20
=17204.28
d) A=10000(1+[0.11/12])^(5*12)
=10000(1.009166667)^60
=$17289.16
e) A=10000(1+[0.11/365])^(5*365)
=10000(1.000030096)^1825
=$17331.09

REGIONAL GROUP NEWS

CARLINGFORD Regional Group.

The next Carlingford Regional Group meeting will be 18th November 1987. Commencing Time 7.30 pm

For further information contact Chris Buttner.

Regular meetings are third Wednesday of each month.

ILLAWARRA Regional Group.

Next meeting 16/11/87 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.

Meeting is expected to be 19/11/87 8pm.

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.

LIVERPOOL REGIONAL GROUP

A good turnout at Larrys place. We were eager to see the Super XB cart, alas no cartridge. Larry had sent it back to get replaced because it had blown a chip. No worries he had let the TI mouse out of the bag, this device ran flawlessly, a winner. One problem we had was we didn't know how to use the disk manager that came with it properly.

Peter Schubert showed his new MULTI- FUNCTION CARD, not quite finished but WOW what a CARD. Superbly laid out with PIO/RS232/DISK CONTROLLER/32K all aboard, imagine you can now free up 2 extra slots in your PE BOX.

Arto gave a quick demo of Picasso 2.0 and loaded some files from a program he received from the USA. Picasso loaded the files without mods and the pictures were 3-D maths functions created with an XB utility program.

NEXT MEETING AT : STEVEN CARR
146 SOUTH LIVERPOOL RD.
BUSBY
PHONE : (02) 608 1968
TIME : 1.30 pm
DATE : SATURDAY 14-11-87

----BARBECUE----
Bring your own meat & drinks.

CENTRAL COAST Regional Group.

Meetings are normally held on Second Saturday of each month at 6.30 pm at Toukley Tennis Club hall, Header St, Toukley.
Next meeting 14th November 1987.

Contact Russell Welham (043 92 4000)

THE CENTRAL COAST REGIONAL GROUP WILL BE HAVING THEIR CHRISTMAS PARTY ON THE FIRST SUNDAY OF DECEMBER AT EBEL CUMMINS HOUSE BRING SWIMMERS AND MEAT

Continued from P8

```
NEXT1  TB   4              Test for LEFT
       JEQ  NEXT2          Jump if not LEFT
       SBO  22             Turn CS1 ON
       SBZ  23             Turn CS2 OFF
       LI   R1,LEFT        Text to display
       JMP  NEXT3          Go and do it

NEXT2  TB   5              Test for RIGHT
       JEQ  NEXT4          Jump if not RIGHT
       SBZ  22             Turn CS1 OFF
       SBO  23             Turn CS2 ON
       LI   R1,RIGHT       Text to display
       JMP  NEXT3          Go and do it

NEXT4  SBZ  22             Turn CS1 OFF
       SBZ  23             Turn CS2 OFF
       LI   R1,BLANK       Text to display(----)

NEXT3  LI   R0,760         Where to put text on screen
       LI   R2,5           Number of bytes in text
       BLWP @VMBW          Write text to screen

       LWPI >83E0          Reload GPL Work Space
       RT                  Return to extended basic

       END
```

EXTENDED BASIC PROGRAM with Assembly in CALL LOADs.

```
100 ! SAVE DSK1.LOAD1
110 CALL CLEAR
120 CALL INIT
130 CALL LOAD(16368,83,84,79,80,32,32,37,54)
140 CALL LOAD(16376,74,83,67,83,32,32,37,44)
150 CALL LOAD(8194,37,140,63,240)
160 CALL LOAD(9460,128,181,176,128,128,0,172,165,166,1
80,128,0,178,169,167,168,180,0,141,141,141,141)
170 CALL LOAD(9482,128,0)
180 CALL LOAD(9516,2,1,37,60,200,1,131,196,4,91)
190 CALL LOAD(9526,4,224,131,196,4,91,2,224,37,12,4,20
4,30,18,29,19,29,20,31,7,19,5)
200 CALL LOAD(9548,29,22,29,23,2,1,36,244,16,18,31,4,1
9,5,29,22,30,23,2,1,36,250)
210 CALL LOAD(9570,16,11,31,5,19,5,30,22,29,23,2,1,37,
0,16,4,30,22,30,23,2,1)
220 CALL LOAD(9592,37,6,2,0,2,248,2,2,0,5,4,32,32,36,2
,224,131,224,4,91,78,213)
230 CALL LINK("JSCS")
240 DISPLAY AT(6,1):"TRY JOYSTICK 1:
     UP, LEFT & RIGHT;THEN FCTN CLEAR AND TRY IN IMME
DIATE MODE"
250 DISPLAY AT(14,1):"If in immediate mode type": :"CA
LL LINK("&CHR$(34)&"STOP"&CHR$(34)&") before": :"rerun
ning line 120"
260 ! here insert RUN "DSK1.NEXTPROG" to go to another
program with Joy Stick control routine still running
270 GOTO 270
```