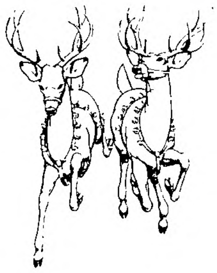
(042) 8812

TISHUG NEWS DIGEST

Focusing on the TI-99/4A Home Computer

Volume 7, Number 11 December 1988

Registered by Australia Post - Publication No. NBH5933



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

TIsHUG (Australia) Ltd.

TIsHUG News Digest

December 1988

All correspondence to:

P.O. Box 214
Redfern, NSW 2016
Australia

Membership and Subscriptions

Joining fee	$5.00
Annual Family Dues	$25.00
Overseas Airmail Dues	AUS$50.00
	or £22.00
	or US$42.00
Publications Library	$5.00
Texpac BBS	$5.00

Cover by George Meldrum

TIsHUG Sydney Meeting

The next meeting will be at 12 noon on 3rd of December at Woodstock Community Centre, Church Street, Burwood.

Printed by
The University of Wollongong
Printery

TIsHUG News Digest ISSN 0819-1984

Index

They're off

by Geoff Trott

We have reached the last magazine for the year, which, as we all know here in Australia, means that summer is approaching. This means that we, the editorial staff (that is Rolf and I), will be able to take a few weeks off to enjoy the festivities and relax without thinking of the deadline approaching next week. I hope you have enjoyed the 11 orange issues and that the next year's will be as good if not better, whatever colour they turn out to be. While we enjoy the heat and sunshine (and rain), some of our readers will be experiencing the opposite extreme of cold and snow. Their days will be getting shorter while ours are getting longer (not just because of daylight saving either) and they will be settling down to many hours at their favourite indoor recreation (not that silly, I mean computing) and we will benefit from that activity next year. It fits in quite well really, with us hard at it when they are on summer vacation and vice-versa. Anyway I really just wanted to say Merry Christmas and a Happy New Year to all our faithful readers where ever you happen to be reading this. Thank you for staying with us for the year. How will you survive to the end of January without your diet of a TND every four weeks. For all the non believers in the Northern Hemisphere, yes we do have Christmas in the middle of summer!

continued on page 31

Secretary's Notebook

by Terry Phillips

First off this month, I hope that all who attended the November meeting had a good time and enjoyed that great demonstration by Ross Mudie and also got the chance to sit in with Craig Sheehan while he gave an interesting session on his XDP Utility. Thanks Ross and Craig for the assistance you provide in imparting your knowledge to the members. Also thanks to Peter Schubert who demonstrated his music card. Contact Peter for further details. Russell was kept busy demonstrating various items, while I hardly a chance to move from the room where I set up a disk copying facility. Thanks also to John Paine who had set up an extra system and took some pressure off me towards the end of the day. And a course a big thanks to Percy, who brought along the food and to the volunteers who took over the BBQ at lunchtime to ensure we were all fed. I must admit I enjoyed the day, although I was a little disappointed at the attendance level.

Further to last month's column, the Inscebot range of software did not arrive in time for the November meeting but the good news is it will most certainly be available at the December meeting. To refresh your memory the items on sale will be:

TI-BASE
TI-ARTIST
ARTIST EXTRAS
DISPLAY MASTER

What I cannot tell you here is the selling prices for each of these items as there is a bit of work to establish just what to charge depending on apportioning the US dollar cost back across the entire shipment plus the added burden of a 20% sales tax slug on the invoiced total of the order. Rest assured however, the costs will be kept very reasonable. This, by the way, is the groups first venture for some time into importing software, so it is up to the membership, and their response, to determine if it is a viable proposition. Judging by the interest shown in TI-BASE in particular, I am sure that sales will be quite good.

We have only one new member to welcome and he (maybe she) is S. Affias of Sydney, Nova Scotia, Canada. Welcome to TIsHUG S. Affias, and I hope to find out what that "S" stands for next time you write.

Jim Banfield has written to advise that he has for sale a surplus Mini PE System (Peter Schubert type) fitted with RS232 and DSDD controller. He is selling for $150, or near offer, and the reason Jim is selling is that it is not suitable for his 1.2MB drives for which he has made his own controller. Well done Jim. Anyone interested in buying can contact Jim at 24 Garibaldi Street, Armidale NSW 2350.

On a more serious note, it will not be too long until the AGM. In this issue you will find an insert giving details of how to nominate members for the position of director. A pro-forma suitable for nominations is also enclosed. I hope you will all give some thought to this over the next month.

And speaking of meetings here is a rundown on forthcoming events:

DECEMBER 3 - CHRISTMAS PARTY.
Given a fine day this will be one of the major events of the year, with plenty of food and drink available and a great chance to chat with fellow members in a social relaxed environment. There will be plenty of software released for this meeting so you will have a lot to keep you occupied over the Christmas/New Year holiday break. Food and drinks will be provided with a start from about 12 noon onwards for socialising. The shop and library facilities will be available from around 2.30pm onwards.

JANUARY - NO MEETING SCHEDULED.
This is the month when we can all relax and enjoy the summer break.

FEBRUARY 4 (1989) - ANNUAL GENERAL MEETING.
Providing the club auditorium is available, then this meeting will be held at the Burwood RSL Club, Shaftsbury Road, Burwood. This meeting will see the election of a board of Directors to run the group for the following 12 months. o

TIsHUG Software Column by Terry Phillips

Good news for those members waiting on copies of TI-BASE, TI-ARTIST etc. This software has arrived and will go on sale from the December meeting. By now you have probably already heard of and many have seen demonstrations of this software and, you have, no doubt been impressed by its versatility. Now is the chance to get your very own copy. Prices will be very reasonable.

An idea I would like to float is, that as more members get copies of BASE and ARTIST, special interest groups be formed to explore the power of these software items. If anyone is of a similar view then I would like to hear from them.

Very little to report in the way of new disks added to the library this past few weeks. Those received are:

DISK A285 - Extended Display Package Tutorial disk from Craig Sheehan - 181 sectors (SSSD) containing lots of program segments as well as tutorial notes.
DISK A286 - Mudie 88/1 (Trainset) - notes and routines on Ross Mudie's train controller tutorial. 692 sectors (DSSD).
DISK A287 - Various files, many of which have appeared on the BBS. This disk is from Stephen Shaw (UK) and contains some very useful information as well as a couple of extended basic games, Blackbox and Bowls. Has some good information on Turbo Pascal. 718 sectors (DSSD).

Coming up at the next meeting will be the following software releases:

DISK A278 - ARCHIVER V3.02 - the latest release of this much used utility. This release has the ability to pack and compress in the one operation. A disk every serious user should have.
DISK A279 - FUNNELWEB 4.12 - again the latest release with a heavily re-worked version of DM1000 which includes the capability of copying ALL files. Normally released as a double sided disk but "flippies" available for those with single sided systems.
DISK A280 - BEATLES ALBUM - some good old Beatles music to sing-along with over the Christmas holidays. Auto loads and plays through all the songs.
DISK A285 - the Craig Sheehan tutorial disk mentioned above.
DISK A286 - the Ross Mudie tutorial disk mentioned above.
DISK A261 - ASSEMBLY GAMES - Math Catcher, Mission X, St Nick, Startrap and TI-Mazog. Editor Assembler module required.
DISK A266 - SIDEWRITER - a very useful utility for Multiplan users in particular, but has other capabilities as well.

Plus this month a tape with the following contents will be released:

CHRISTMAS TAPES 1988:
1 - Graphic Designer by local author Wade Bowmer. This is a well programmed utility by young Wade so I suggest you get a copy and see for yourself.
2 - A tape of games mainly to keep the kids occupied over the holiday period. Full titles yet to be decided but will contain Beagle Hike, a game where the player has to get Snoopy home, Fruit Cocktail, catch the falling objects, Eat Mince, gobble up pies, Lift Attendant, save falling people, Slugs and Ladders, like snakes and ladders, plus more. c

The Communicators

Special Interest Group for
Users of the TEXPAC BBS.
by Ross Mudie, 10th November 1988.

The communicators column missed out last month due to the massive effort required to prepare the Tutorial Day project for the November meeting. About 40 members saw the project which was a small model train set being controlled from the keyboard and joy stick of a cassette loaded 32K TI99/4A console. I have decided to keep the train set intact for possible showing at some of the regional group meetings early in the new year. The project, including the 56 page book, required about 360 hours of work which occured over a very rushed 2 month period. This project is not just about model trains, it shows a practical way to interface the TI99/4A to many different control functions.

This is my last column before Christmas, so I would like to wish all members and their families, a very happy Christmas and New Year and a safe holiday. The BBS will be on line right through the holiday period. Now to the business side of this month's column.

1. BBS USAGE.

Usage of the BBS has been falling almost all year, regardless of improved system availability. BBS members are again asked to review their BBS usage. It is rather dis-heartening to put in the needed effort each month to prepare programs and review information files to find that less people seem to be using the service. The questions that I ask are (a) Is it just that people are too busy? (b) Has the novelty worn off? (c) Have too many people gone over to the big blue? (d) Do the users want something different in the BBS? Interested users are invited to log on to the BBS and advise the SYSOP by using the electronic mail feature. Here are the statistics for this year so far:

1988	January	February	March	April	May
CALLS	393	329	328	302	250
USERS	49	61	52	59	55
Occupancy Hours/min	156.26	119.39	123.40	120.13	103.37

1988	June	July	August	September	October
CALLS	231	255	229	221	164
USERS	51	52	57	53	47
Occupancy Hours/min	94.24	90.49	84.31	83.38	65.41

2. BBS PROBLEMS.

The BBS failed twice in October due to mains power interruptions. Since the BBS operates unattended, restoration from an evening failure usually occurs on the next working day. Power interruptions are now the major causes of BBS failure.

3. DOUBLE DENSITY DISK CONTROLLER for BBS.

The directors of TIsHUG have agreed to purchase double density disk controllers for the BBS and the BBS simulation. This will allow a greater number of files and programs to be placed on the BBS at any one time. It will also overcome congestion which was occuring on the NEWS disk due to the size of some of the sub-editor contributions. An immediate effect of the change is that the NEWS menu will tend to over-run the screen but steps will be taken to break the menu up into category areas. There will be more information published on this subject once the necessary software changes have been tried.

4. BBS SESSION TIME LIMITS.

With falling usage of the BBS the Directors of TIsHUG have requested that the system time limit be extended to 60 minutes per call, which has been implemented. Users are asked to still allow 30 minutes between calls to give others a chance to log on.

When on line to the BBS there are two timers which can cause your call to disconnect automatically. These both work on lack of activity, the times being 9 and 12 minutes. These timers are primarily designed to prevent the BBS from being held up if a user forgets to disconnect the modem from line. A problem can occur during program download if there is a delay of greater than 5 minutes from when you select a program to down-load until you actually commence the download as the 12 minute watchdog timer is not given a reset pulse from when the actual program selection is made until you return to terminal mode again and press <enter>. One member has had the BBS force release part way through download. It is therefore strongly advised that long delays in commencing program download after the program is selected should be avoided.

5. BBS MEMBERSHIP.

Access to the BBS is limited to BBS members only. TIsHUG members wishing to join the BBS should contact the Secretary; membership rates are shown on page 1 of the TND magazine.

From the Bulletin Board

MAIL TO : ALL
MAIL FROM : LARRY

Just received copies of Press, Batch-It, Quick-Run and EZ-Keys Plus. These will be demonstrated at the next two meetings. Press is the best word processor that I have ever seen on any computer or PC it is better that the Apple word processor that is advertised a lot on TV. It will do a letter up to 380K in size. The only limit is disk or RAMdisk size. It has a 100,000 word spelling checker and tons of pull down screens to do anything and everything you ever wanted to do with a word processor.

EZ-Keys Plus has checksum, disk catalogue, Hilite, screen dump and macros that you can input one or more times into the Micro. Eg, FOR (input) = (input) TO (input) :: NEXT (input) :: Ideal for database programs.

Bye for now LARRY

MAIL TO : ALL
MAIL FROM : SHOP

Hello to you all, again. The Shop has now received some of the RAMdisk EPROMs. They will sell for $25 each. Any enquiries would you please direct them to me via BBS as at the moment I have had a few small things which I have had to attend to first. I have not as yet received the latest issue of MICROpendium from Stateside. I have heard that there are problems at the docks in Sydney. As soon as I hear otherwise, I will let you know.

That is all for now. See you soon. Steven Carr

MAGNETIC DATA STORAGE has moved from Kensington to Summer Hill. The new address is 5 Grosvenor St, Summer Hill, phone (02) 798 3833. This company sells MPI drives and is very capable of repair of disk drives.

ATTENTION MEMBERS NEEDING TMS9900 or TMS9901 chips.

TIsHUG has purchased a quantity of each of these chips. Contact CYRIL or SHOP for one off prices. For people wanting quantities of these, contact Peter Gleed in Melbourne on (03)877 4790. Prices are approximately $10 each for TMS9900 + packaging and postage. (I do not remember the price of the TMS9901 chips but I think it is <$10 each).

COLOUR MONITORS.

Peter Pedersen is interested in organising a bulk purchase of colour monitors. He needs at least 4 other people to get a bulk discount of $180 each. These are reconditioned units and will require a special interface between the TI99/4A modulator socket on the console and the monitor.

Interested? Contact Peter Pedersen on (02)772 2396

MAIL TO : ALL
MAIL FROM : GAMES

At this moment Games Inc. would like to offer a incredible range of TIsHUG tapes for sale. It includes over 30 tapes. It is a devoted member's collection of tapes way back from the end of 1984 until the beginning of this year. Every tape of 1985 and 1986 are here and all but 2 of 1987, and a few of late 1984. These tapes must be sold together. The cost for the 42 tapes is only $60. These tapes brand new would cost well over $100.

Also for sale is a Tunnels Of Doom cartridge plus 2 tapes full of different scenarios so this is not a cartridge that one can get bored from. The price for the cartridge and 2 cassettes is only $45. Please forward your questions about these goods to GAMES. Thankyou.

PS Games would like to point out that these are virtually collector's items that will be desired among many.

MAIL TO : ALL
MAIL FROM : PETESAKE

I now have brand-new 3 1/2 inch disk drives which I have mounted into the same size diecast box as the Mini-PE System. These are complete with power supply and interface cable and with the AT disk controller will store 180K (720 sectors) in SSDD format on standard 3 1/2 diskettes and can use the fastest Head step setting of 3ms. Power consumption is only 300mA at 12 volt so these drives can be used on a portable system. Also included is a switch to select as drive number 1 or 3. Price of the complete disk drive box is $130 while they last.

FOR SALE..... one TI RS232 card for PE box for $140. Phone Peter on 358 5602.

MAIL TO : ALL
MAIL FROM : PAINEY

For Sale:

One only HRD+ 1 meg Ramdisk, fully chipped and working. $600.00 Chip cost alone around $800.00
One only PE box (surplus) $200.00
With 2 slim line drives add $200.00
One TI RS232 card $120.00
One AT multifunction card with 32K, DSDD disk controller, dual RS232 and PIO port. $250.00

All the above are surplus to existing requirements and urgent sale is required.

I also have a number of other bits of hardware available. Please leave mail on this BBS or call John Paine (02)625 6318 after 7.00pm any night.

Regards John Paine

MAIL TO : ALL
MAIL FROM : PETESAKE

Mini-PE Components

I now have further stocks of the Mini-PE Motherboard/Expansion. This is the one that allows you to add a PIO printer port, and also has room for two RS232 Ports and 32K Memory if needed. Other Mini-PE boards that can Piggy-back onto it include the D-Dens. disk controller, 128-512K RAMdisk board, and shortly the 16 Channel Music Board will be available. For more information contact Peter Schubert, Box 28, Kings Cross 2011, or Phone (02)358 5602.

FOOTNOTE: I am presently out of stock of Disk controller boards for the Mini-PE but if you are contemplating getting one in the near future please let me know as I will need to order more boards soon.

MAIL TO : ALL
MAIL FROM : MUSICMAN

<u>FOR SALE; FOR SALE; FOR SALE.</u>

Complete TI System.

Consisting of: SSSD Disc Drive; TI Control card; 32K memory; RS232 card.

Including: Epson LX86 Printer; Paper and ribbons; Disc, tapes and books.

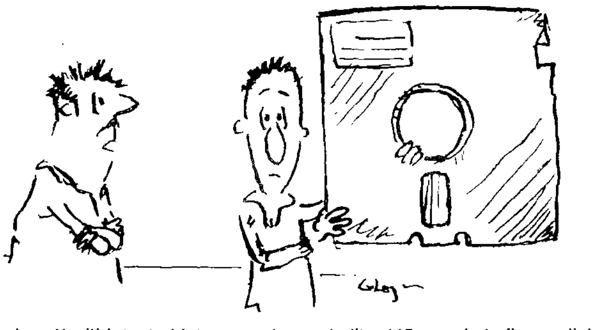
Cost: $1500-00.

Contact: Scott Bone. (043)92 5058.

---

continued from page 19

```
810 REM
820 CALL CLEAR
830 PRINT "RESULTS":"-------": : :
840 PRINT "RANGE IN ";D$
850 PRINT "TIME IN SECONDS"
860 PRINT
870 PRINT L$
880 PRINT
890 DISPLAY "RANGE COVERED:";TAB(15);R
900 DISPLAY "HIGHEST POINT:";TAB(15);A
910 DISPLAY "  FLIGHT TIME:";TAB(15);D
920 PRINT
930 PRINT L$
940 PRINT
950 INPUT "<A>GAIN OR <Q>UIT? ":A$
960 IF (A$<>"A")*(A$<>"Q")THEN 950
970 IF A$<>"Q" THEN 600
980 REM
990 CALL CLEAR
1000 END
1010 REM
1020 REM *TITLE DATA*
1030 REM
1040 DATA "**********************"
1050 DATA "*                    *"
1060 DATA "* TRAJECTORIES DEMO *"
1070 DATA "*                    *"
1080 DATA "**********************"
1090 DATA ,,,"      04/85"
1100 DATA "      SUBFILE99"
1110 DATA ,,,,,,,
1120 REM
1130 REM ******************
1140 REM *TRAJECTORY CALCS*
1150 REM ******************
1160 REM
1170 REM
1180 REM *CMS CALC*
1190 REM
1200 R=CMSRANGE
1210 A=CMSALT
1220 D=CMSTIME
1230 D$="CENTIMETRES"
1240 RETURN
1250 REM
1260 REM *FPS CALC*
1270 REM
1280 R=FPSRANGE
1290 A=FPSALT
1300 D=FPSTIME
1310 D$="FEET"
1320 RETURN
1330 REM
1340 REM *KPH CALC*
1350 REM
1360 R=KPHRANGE
1370 A=KPHALT
1380 D=KPHTIME
1390 D$="KILOMETRES"
1400 RETURN
1410 REM
1420 REM *MPH CALC*
1430 REM
1440 R=MPHRANGE
1450 A=MPHALT
1460 D=MPHTIME
1470 D$="MILES"
1480 RETURN
```


Ingenious Nevil! I don't think anyone's ever built a 115 megabyte floppy disk.

Publications Library Report

with Warren Welham

Well I am back and with quite a few additions to the library. Just a reminder that all books and overseas newsletters have to be back in by the Febuary meeting or on the Febuary meeting, remember there is no borrowing at the Febuary meeting. Also I am glad to my appeal working and getting old TNDS into the library.

As you may notice the library is getting quite a few cassettes and modules so I have decided to form another section of the library which will include Genuine Texas Instruments or 3rd Party Disks, Tapes or Modules but to do this I need people to donate them to the library (hopefully with books but not necessary). These would be available to all library members, especially country members who may still be using cassettes and are having trouble obtaining software. So people who are now on disks and do not use their cassettes anymore, why not dig them out and donate them, so all club members can get use out of them. (This does not include your own software)

New Arrivals of Overseas Publications

Group	Publications Name	Date
Channel 99 Hamilton	TI Focus	Sep 88
Hunter Valley 99ers User Group	Hunter Valley News	Jul 88
Hunter Valley 99ers User Group	Hunter Valley News	Aug 88
-	MICROpendium	Jun 86
-	MICROpendium	Jul 86
-	MICROpendium	Aug 86
-	MICROpendium	Sep 86
-	MICROpendium	Feb 87
-	MICROpendium	Apr 87
-	MICROpendium	May 87
-	MICROpendium	Jun 87
-	MICROpendium	Apr 88
-	MICROpendium	Jun 88
-	MICROpendium	Jul 88
-	MICROpendium	Aug 88
-	MICROpendium	Sep 88
Pittsburgh Users Group	The Pug Peripheral	Aug 88
T.I.U.P Inc. Perth	Tit Bits	Jul 88
Orange County User Group	ROM	Jun 88
Orange County User Group	ROM	Jul 88
Sacramento 99ers User Group	Network	Aug 88
LA 99er Computer Group	TopIcs	May 88
LA 99er Computer Group	TopIcs	Jun 88
-	TI Home	Aug 83
Canberra Home Computer Group	Chug-a-lug	Jul 88
Central Ohio Ninety Niners Inc.	Spirit of 99	Jul 88
International User Group	-	Dec 81
International User Group	-	Feb 82
Tacoma 99ers Users Group	The Tacoma Informer	Aug 88
Brisbane User Group	Bug Bytes	Jun 88
Brisbane User Group	Bug Bytes	Aug 88
Club Information Montreal	Cim 99	Jul 88
Club Information Montreal	Cim 99	Aug 88
Ottawa TI-99/4A Users Group	Newsletter	Sep 88
San Diego Computer Society	-	May 88
San Diego Computer Society	-	Jun 88
San Diego Computer Society	-	Jul 88
San Diego Computer Society	-	Aug 88
Northern New Jersey	-	Jun 88
Northern New Jersey	-	Jul 88

New TND's arrived this month

Apr 87	Nov 84	Apr 86	Jun 86	May 85	May 83	Jun 82
May 87	Oct 84	Jun 85	Jul 86	Sep 88	Apr 83	Jul 82
Sep 87	Sep 84	Oct 87	Mar 86	Aug 88	Jun 83	Aug 82
Oct 85	Mar 85	Mar 87	Apr 85	Mar 83	Aug 83	Sep 82
Oct 82	Dec 82	Nov 82	Feb 83	Dec 81	Feb 82	Mar 82
Apr 82	May 82	Mar 84	Aug 84	Jul 84	Jun 84	May 84
Apr 84	Feb 84	Jul 83	Nov 88	Vol.1 Nos.2,3,4,5,6,7,8		

New Books arrived this month

Code	Title	Author
00247	99'er HCM Vol.1 No.1 May 82	HCM
00248	99'er HCM Vol.1 No.2 Jun 82	HCM
00249	99'er HCM Vol.1 No.3 Jul 82	HCM
00250	99'er HCM Vol.1 No.4 Aug 82	HCM
00251	99'er HCM Vol.1 No.5 Sep 82	HCM
00252	99'er HCM Vol.1 No.6 Oct 82	HCM
00253	99'er HCM Vol.2 No.1 Nov 82	HCM
00254	99'er HCM Vol.2 No.2 Dec 82	HCM
00255	99'er HCM Vol.2 No.3 Jan 82	HCM
00256	Enthusiast'99 Vol.1 No.1	Inter/99-4
00257	Enthusiast'99 Vol.1 No.2	Inter/99-4
00258	Enthusiast'99 Vol.1 No.3	Inter/99-4
00259	Enthusiast'99 Vol.2 No.2	Inter/99-4
00260	Enthusiast'99 Vol.2 No.3	Inter/99-4
00261	Understanding Computer Science	T.Instrument
00262	Understanding " " Vol.2	T.Instrument
00263	Understanding Microprocessors	T.Instrument
00264	Tax/Investment Record Keeping	T.Instrument
00265	Compute's Guide to Assembly Language on the TI-99/4a	Compute
00266	Sams TI-99/4A 24 Basic Programs	C.Casciato
00267	Treasure Island	T.Instrument
00268	Hopper	T.Instrument
00269	Video Games 1	T.Instrument
00270	Addition and Subtraction 1	T.Insrtument
00271	Chisholm Trail	T.Instrument
00272	Early Reading	T.Instrument
00273	Facemaker	T.Instrument
00274	Car Wars	T.Instrument
00275	Alpiner	T.Instrument
00276	The Attack	T.Instrument
00277	Munchman	T.Instrument
00278	Division 1	T.Instrument
00279	Multiplication	T.Instrument
00280	Blackjack and Poker	T.Instrument
00281	Reading On	T.Instrument
00282	Parsec	T.Instrument
00283	Star Trek	T.Instrument
00284	Tombstone City	T.Instrument
00285	A-Maze-Ing	T.Instrument
00286	Video-Graphs	T.Instrument
00287	TIsHUG Tutorial Nov'88-Wire Accessory Interface	Ross Mudie
00288	HCM Vol.4 No.11	HCM
00289	TIsHUG Tutorial Jun'84	TIsHUG
00290	Computer Music Box	T.Instrument
00291	TI-Trek	T.Instrument
00292	Personal Financial Aids	T.Instrument
00293	Speak and Math Program	T.Instrument
00294	Music Maker Demonstration	T.Instrument
00295	Electrical Engineering Library	T.Instrument
00296	Speak and Spell Program	T.Instrument
00297	Programming Aids 1	T.Instrument
00298	Draw Poker	T.Instrument
00299	Bridge Building 1(with cassette)	T.Instrument
00300	Bridge Building 2(with cassette)	T.Instrument
00301	Bridge Building 3(with cassette)	T.Instrument
00302	Draw Poker (with cassette)	T.Instrument
00303	Ghost Town (with cassette)	T.Instrument
00304	AC Circuit Analysis (with tape)	T.Instrument
00305	Programming Aids 1 (with tape)	T.Instrument
00306	Saturday Night Bingo (with tape)	T.Instrument

Extended Display Package

by its author, Craig Sheehan

Having discussed the DISPLY and ACCEPT commands in the first two parts of this series, we are now in the position to examine one of XDP's more important features: windows. Windows should be nothing new, they exist in normal Extended BASIC. When using DISPLAY, PRINT, INPUT, etc., text is restricted to a window with a top row 1, bottom row 24 and left and right columns of 3 and 30 respectively. The top corner of this window was (1,1) despite the fact that its real co-ordinate was (1,3). XDP's windows are no different. The top left hand corner is now row 1, column 1 by definition and all other positions numbered relative to this position. Just like Extended BASIC's window, commands that use it cannot display data outside of it.

All of XDP's commands operate exclusively in the window with the exception of sprites and CLRS. The advantage with XDP's windows is that their boundaries are not fixed and the programmer can change them whenever there is a need. There are four ways to set the boundaries, of which we will first look at two. One of them you have seen before:

```
CALL LINK("XDP" [,mode [,top row [,bottom row
    [,left column [,right column]]]]])
CALL LINK("WINDOW" [,top row [,bottom row [,left column
    [,right column]]]])
```

Notice the similarity between the argument list for WINDOW and that for XDP if the mode were to be removed. The first argument of XDP allows the mode to the selected (either 32 or 40 columns) and besides that operates in an identical manner to WINDOW.

The top row, bottom row, etc. refers to the boundaries of the window and includes that row or column. For example,

```
CALL LINK("WINDOW",3,20,12,18)
```

defines a window with (1,1), the top left hand corner, corresponding the real position (3,12) and has a size 18 rows by 7 columns.

With both commands, any arguments that are left out will be given default values. The modes default is 32, and since if the mode is not specified, you cannot include a window list, the 32 column window defaults also take effect. Defaults for the window depend on the mode: top row 1, bottom row 24 with left and right columns of either 3 and 30 or 1 and 40 for 32 or 40 column mode respectively. As an example:

```
CALL LINK("XDP",40,12,16)
```

sets 40 column mode and the window to a top row of 12, bottom row 16 and defaults the left and right columns to 1 and 40 respectively.

Please note that the defaults can have a nasty consequence. If, whilst in 32 column mode,

```
CALL LINK("WINDOW",1,24,31)
```

is executed, a bad value error message will be returned. This is due to the left column, with a value of 31, being larger than that for the right column, defaults to 30. Any of the other commands that allow windows to be set will also return the same error.

There are two other ways of changing the size of the window. The first is with the MODE command, which has the same argument list as the XDP command. This command is used mainly to change between 32 and 40 column mode, and since there is a change in the number of columns, the option of setting the window boundaries is offered. The second method for changing the size of the window is via a specification command from DISPLY: '.WI'. It has four arguments which correspond to the top row, bottom row, left column and right column. Unlike the other methods of setting the window, all of the arguments must be given. When the window is changed, the next character will be displayed at position (1,1), the top left hand corner of the new window.

To get a feel for what effect the window has on the display command, the following program creates a string of ASCII characters from 33 to 126 and displays them in the window from (1,1). Initially the window is set to the boundaries of the 32 column display mode, i.e. (1,24,3,30). Each time a key is pressed, the screen is cleared and the size of the window reduced by one from all directions until it no longer exists. Each time the string will be redisplayed. Notice that text wraps around at the bottom right hand corner to the top left hand corner.

```
100 CALL LINK("XDP")
110 FOR CH=33 TO 126 :: A$=A$&CHR$(CH):: NEXT CH
120 FOR X=1 TO 12
130 CALL LINK("WINDOW",X,25-X,2+X,31-X)
140 CALL LINK("DISPLY",1,1,".VA.KD",A$)
150 NEXT X
```

Before typing in the program, do not forget to load the package as described in previous months. A new specification command appears on line 140: '.KD' pauses until a key is pressed.

An alternative method of achieving the same effect is to use the '.WI' specification command. To do this, delete line 130 and replace line 140 with:

```
140 CALL LINK("DISPLAY",1,1,
    ".WI(.VA,.VA,.VA,.VA).VA.KD", X,25-X,2+X,31-X,A$)
```

The '.VA' as a parameter for the specification command causes it to retrieve the next value from the variable list as the argument, just as the '.VA' used to display A$ gets the next item from the variable list to be displayed.

In last months article, the 'CLRS' command was shown as one way to clear the screen. There is a similar command to clear the window, which has the syntax:

```
CALL LINK("CLRW" [,character code])
```

This command works identically to the clear screen command except that it only clears the characters within the window. Optionally, the screen can be cleared with a character other than the space (ASCII 32) by supplying a character code as an argument. If it is not supplied, the space character is used as a default.

CLRW can be used by itself, with a bit of aid from the windows, to create some useful effects. One is make a border by first clearing the screen to a certain character and then clearing the window with the space character, as shown below:

```
100 CALL LINK("XDP",32,3,22)
110 CALL LINK("CLRS",35)
120 CALL LINK("CLRW")
130 GOTO 130
```

Notice that CALL LINK("CLRW") is equivalent to CALL LINK("CLRW",32).

More information on the new commands discussed in this article can be found in the XDP Utility Guide: CLRW (page 17), MODE (page 25) and WINDOW (page 36). The next part of this series will be in much more depth than previous articles. Due to this, it would be advisable to review all of the references given so far and ensure you understand them.

In the January/February TND: more applications for windows. In particular we will build up a set of subprograms to give interactive windows from Extended BASIC.

Techo Time

Super Module

by Lou Amadio and Geoff Trott

The following hardware project was first suggested by Rolf Schreiber and later designed and enhanced by Geoff Trott. Its purpose was to provide a unique and powerful hardware device by incorporating several facilities within the one module. I call the new device a SuperModule as it offers quite a lot for the power user: MiniMemory, Editor/Assembler and four 8Kbyte regions of battery backed static RAM. I hate to think what TI would have charged for such a module; the MiniMemory alone once retailed for over $100! The internal battery allows you to store up to 4 programs for use at any time. In effect, the new module could be used as a mini RAMdisk with, for example, a menu to run your favourite programs from floppy or even for playing ROM based games such as Ambulance.

Parts required:

1 of MiniMemory Module
1 of Editor/Assembler GROM chip
1 of 32Kbyte x 8 bit Static RAM (62256)
3 of SPDT miniature toggle switches - Dick Smith Electronics part # S-1173 or Tandy part # 275-662
1 of sub-miniature push button switch - Dick Smith Electronics part # S-1101 or Tandy part # 275-1571
2 of 1N4148 diodes
1 of 1.5K 1/4 watt resistor
1 of 10uF Tantalum 16V capacitor
6 mm wide Dymo label tape

How to Build it

The MiniMemory module is used as the basis for the SuperModule. Remove the PCB from the module and start by stripping off all components except for the 74LS04N, the MiniMemory GROM, the battery, the 100ohm RESET resistor and three 0.01uF capacitors as indicated.

Remove the label from the back of the MiniMemory module and drill four 6mm holes as indicated to suit 3 miniature SPDT toggle switches and one sub-miniature push button switch. The toggle switches must be mounted sideways and located high enough on the upper module case so that they do not interfere with the PCB during re-assembly. Locate SW4 within approximately 15mm from the side of the module case so as to clear the 74LS04 chip. During construction, ensure that no wiring etc is routed within approximately 12mm from the back of the PCB to allow room for the switches, which will be very close to the PCB (see diagram). A strip of insulating tape approximately 12mm wide must be used across the back of the PCB to prevent shorting to the switch case during final assembly.

As the push button switch is smaller than the toggle switches it should be mounted in the indicated position opposite the 74LS04N chip.

The 32Kbyte memory chip is mounted in the position vacated by the MiniMemory ROM chip. Again this position was chosen so as to allow room for the toggle switches. There is not enough room to use an IC socket, so be careful with all soldering around this chip (mind static electricity).

Carefully bend the pins of the 62256 RAM chip in a little to allow insertion into the PCB. Bend out the following pins for further connection:

1, 2, 20, 22, 23, 26, 27 and 28

Insert the IC with pins 1, 2 and 27, 28 "over hanging" the PCB holes vacated by the MiniMemory ROM. Ensure that the orientation is the same as for the other chips (towards the battery).

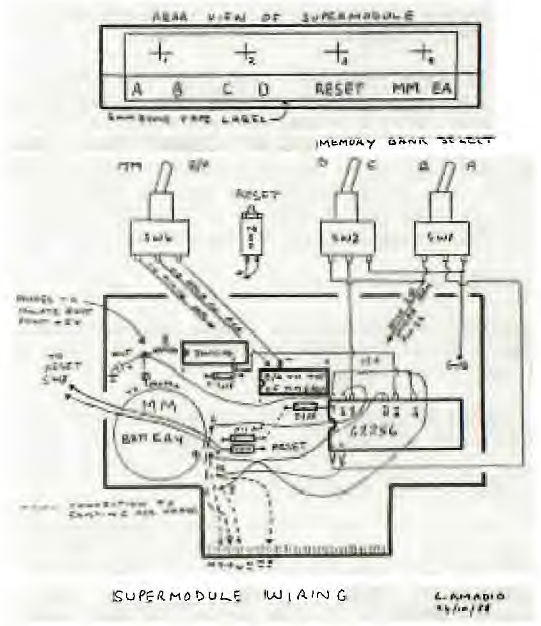
Desolder pin 14 (-5V supply) of the MiniMemory GROM and pull it clear of the PCB. Mount the Editor/Assembler GROM on top of the MiniMemory GROM (same orientation) and solder all pins except 14. Solder a wire from the hole in the PCB (left by pin 14 of the MiniMemory GROM) to the pole of switch SW4. This is the GROM select switch. Solder a wire from the remaining 2 contacts of SW4 to pin 14 of each GROM.

Solder wires between the contacts of the 2 memory bank select switches (SW1, SW2) as indicated. The poles of these switches are used to alternatively switch pins 1 or 26 of the 62256 high or low.

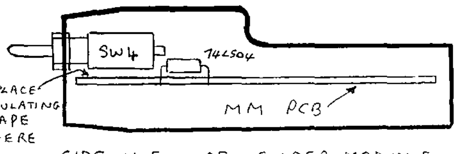
Make the following solder connections, using fine insulated single core copper wire where appropriate:

Connect pin 1 of 62256 to pole of SW1
Connect pin 2 of 62256 to contact 14 of module connector
Solder pins 3 to 19 of 62256 to PCB
Connect pin 20 of 62256 to contact 4 of module connector
Solder pin 21 of 62256 to PCB
Connect pin 22 of 62256 to pin 6 of 74LS04N
Connect pin 23 of 62256 to contact 8 of module connector
Solder pins 24 and 25 of 62256 to PCB
Connect pin 26 of 62256 to pole of SW2
Connect pin 27 of 62256 to contact 6 of module connector
Connect pin 28 of 62256 to junction of the two 1N4148 diodes as indicated
Solder the two 1N4148 diodes to the PCB as indicated. The diodes isolate the battery from the +5V supply.

Solder the 10uF tantalum capacitor (note polarity) from the junction of the diodes to earth. This capacitor reduces battery impedance.
Solder the 1.5K resistor between pins 20 and 28 of the 62256
Lift one end of the 100 ohm RESET resistor and re-connect via the push button SW3



Once the wiring has been completed, recheck everything against the diagrams and instructions. Check the battery voltage (~3V) and replace if necessary. Place a strip of insulating tape across the back of the PCB where there is a possibility of contact with the switches. Carefully re-assemble the module and route the wiring so as not to interfere with the toggle switches. Use 6mm Dymo tape to label the switch functions as per illustration above.



SIDE VIEW OF SUPER MODULE

How To Use The SuperModule

Insert the module into the console and use the switch to select Editor/Assembler or MiniMemory as required and press the reset button. Proceed as normal except that, since we have now removed the MiniMemory ROM, some of the features of MiniMemory are not yet available.

The two bank select switches (marked A B and C D respectively) are used to select one of four 8Kbyte memory banks at >6000. In order to assist with memory bank identification, the following table was printed out and fixed to the module with clear contact vinyl covering.

SuperModule		
Bank	SW1	SW2
1	A	C
2	B	C
3	B	D
4	A	D

The table above indicates the switch positions to select each memory bank. Programs or data may be stored in these 8Kbyte memory banks and will remain there until over written (or the battery goes flat). The memory banks are independent of the GROM selector switch. If MiniMemory is selected the 8Kbyte space may be used to save BASIC programs.

The MiniMemory LOAD AND RUN option which normally resides in ROM can be restored by first saving the ROM contents (from an unmodified MiniMemory) to disk and then transferring the file to one of the 8Kbyte memory banks. Contact your local club for the appropriate software.

During use, observe all of the normal precautions which are required with MiniMemory, such as switching the console power off before inserting or removing the SuperModule. This ensures that the RAM memory contents will not be corrupted.

Other uses for the SuperModule are as per suggestions for the SuperSpace and SuperCart modules found in MICROpendium. This includes applications as a mini RAMdisk to store and run your own programs. o

Taking the heat off the TI99/4A

by Lou Amadio

Have you noticed that the module port bay on your TI99/4A can get quite warm after a good session on the computer? If not, then you probably have a switch mode power supply which was fitted to some of the later consoles. Most likely, however, you will have the standard power supply and will probably have wondered if the excess heat is damaging your console. It is difficult to gauge the long term effects of heat on electronic components, but I am of the opinion that all unnecessary heat should be eliminated or reduced as much as possible.

I have seen several attempts at reducing excess heat, such as a specially constructed fan enclosure to draw air through the vents at the rear of the console or the complete removal of the power supply printed circuit board (PCB). I felt that most of these solutions were either too cumbersome or impractical in my situation.

Console Power Requirements

When TI designed the original power supply for the console they elected to use a linear IC voltage regulator for the +12 volt and -5 volt supplies and a pulse width modulated (PWM) supply for the +5 volt supply.

The current consumption of the console, with modulator connected, was estimated as follows using a Hall Current Probe connected to an oscilloscope:

+5 volts at	~ 900 mA
-5 volts at	~ 80 mA
+12 volts at	~ 330 mA

These currents are only approximate but will suffice for this exercise. The transformer supplied by TI is rated at 16 volts ac at 2 amps and 8 volts ac at 100 mA. When rectified, these windings produce 22 volts and 11 volts dc respectively.

The 16 volt ac winding is used for both the +5 and +12 volt regulators. Now 22 volts dc is fine for the +5 volts supply as this uses a PWM circuit. The relatively high voltage ensures that, even at high currents, the circuit runs efficiently.

Power dissipation in the linear voltage regulators can be calculated thus:

P = (22-12) * 0.33 = 3.3 watts for the +12 volt regulator, and

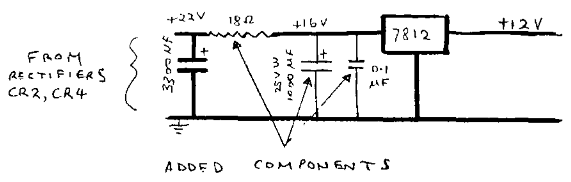
P = (11-5) * 0.08 = 0.48 watts for the -5 volt regulator.

It can be seen therefore that the +12 volt regulator (7812) is the one responsible for most of the unwanted heat generated inside the console.

My first attempt at reducing excess heat in the console was to insert a resistor in series with the 16 volt ac winding in the transformer case. Although it reduced some heat build up in the +12 volts regulator, it was probably not worth the effort as it increased heat dissipation in the +5 volts regulator (due to the longer average "on" time for the PWM circuit).

Some time later, having discussed the matter with Geoff Trott, I struck upon the best solution. This was to reduce the input voltage to the +12 volt regulator to the minimum required for correct operation. Linear IC regulators need approximately 3 to 4 volts above their rated output to work effectively, that means about 16 volts for a 7812.

I achieved this by interfacing a suitable resistor/capacitor network between the existing filter (3300 uF) and the 7812 regulator. An advantage of this modification over the alternatives mentioned above is that it is independent of other hardware.



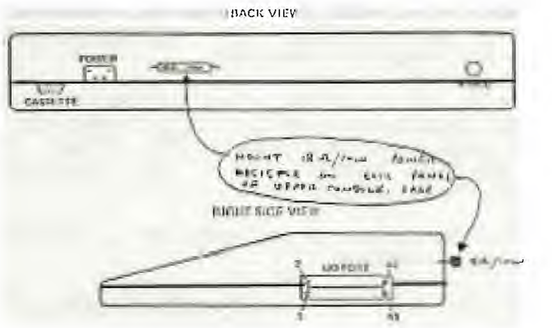
The value of the resistor is calculated using the following formula:

R = (22-16)/0.33 = ~18 ohms

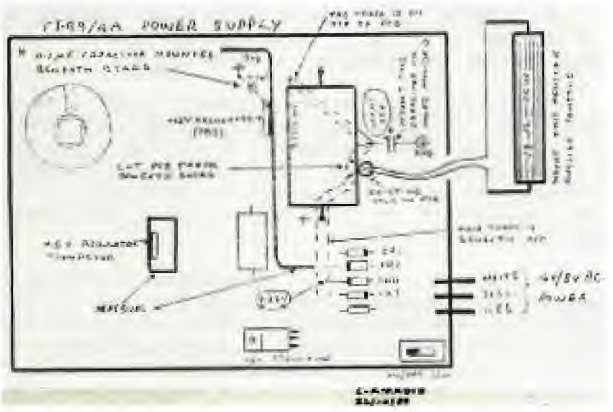
Power dissipation in the resistor is calculated as follows:

P = (22-16)*.33 = 2 watts

Although a 1 watt resistor would work, I found that a 10 watt unit was better able to dissipate the heat. Naturally the resistor must be mounted outside the console if we are to achieve our objective. The back of the console is probably the most logical place for this resistor. I mounted mine on the upper half of the case, approximately 30 mm from the power connector. Dismantle the console and drill 2 small holes in the case to suit the resistor. If the holes are made small enough, then the resistor will be self supporting. Mount the resistor clear of the console by approximately 3 mm to assist the heat dissipation. The easiest way to do this is to use a short piece of insulation from thick copper wire.



A twisted pair of wires is then fed from the resistor, along the back of the console, around the side, past the I/O port to the power supply PCB. Be careful that the wires do not interfere with I/O devices being inserted. A track on the power supply PCB must be cut prior to terminating the two wires on the board (see diagram).



A 1000 uF (25 volts) filter capacitor is used to reduce the impedance of the circuit supplying the 7812. Another capacitor (0.1 uF) is connected across the 1000 uF in order to prevent high frequency oscillation in the 7812 regulator. The 1000 uF capacitor is located on the power supply PCB in holes conveniently provided by TI! The 0.1 uF capacitor should be located as near as possible to the input of the 7812 regulator.

Once you have completed the modification you will be surprised at the temperature rise of the resistor which is now getting rid of heat previously trapped inside the console.

LOAD Interrupt

by Chris Lauhead, USA

One of the simplest hardware projects for our computer is the installation of a load interrupt switch. I imagine thousands of speech synthesizers and consoles have been modified to incorporate this switch. Once installed, however, most users realise they do not have the foggiest idea how to use this new "feature". Hopefully this article will help you understand what a load interrupt switch does and how it can be used. You will better understand the following discussion if you have at least some knowledge of assembly language, but you might find the information interesting even if you do not fully understand it.

One of the input pins on the TMS9900 microprocessor chip in the console is labeled NOT LOAD, which I will call -LOAD. The NOT means the signal is active if low. Since ground is low, every time you press the load interrupt switch (which grounds this pin), you activate this signal. If -LOAD is low when the microprocessor has just completed executing an instruction, the load interrupt will be executed instead of the next program instruction.

So, what does the microprocessor do when the -LOAD pin is low? You may find it surprising , but it executes only ONE instruction! That instruction is BLWP **>FFFC.**

That is it --- the rest is up to the programmer. Lest you fear that all the work of installing the load interrupt switch was for naught, let me hasten to point out that BLWP is an extremely powerful instruction. BLWP is the assembly language mnemonic for branch and load workspace pointer. This instruction is similar to mainframe instructions used to process many programs concurrently. It allows you to branch to another program, complete with its own set of registers, AND to get back to the original program where it left off.

In order to understand what the load interrupt can do for you, therefore, you have to understand the BLWP instruction. BWLP causes what TI calls a context change. This term is not very descriptive. Control Data calls their somewhat similar instruction an exchange jump. This is a bit more descriptive of what happens when a BLWP is executed. The data at addresses >FFFC and >FFFE (in the case of a load interrupt) are exchanged for what was in the workspace pointer register (WP) and program counter register (PC). The old values that were in WP and PC are put into the new R13 and R14. Also the status register is stored in the new R15. This is all the information that is necessary to return to the interrupted program. By "new" I mean the registers in the new workspace that was loaded from the data at >FFFC.

The chart that follows shows what happens if the load interrupt button is pressed while an instruction is being executed at address >A120 with workspace at >A000, assuming the instruction is only 2 bytes long and is not a branch or jump. Note that PC usually points to the NEXT instruction to be executed.

BEFORE			AFTER	
WP	>A000	-------	R13	>A000
PC	>A122	-------	R14	>A122
STATUS	>XXXX	-------	R15	>XXXX
>FFFC	>8300	-------	WP	>8300
>FFFE	>602C	-------	PC	>602C
			>FFFC	>8300
			>FFFE	>602C

Before the button is pressed the instruction at >A120 is being executed. If the button were not pressed, the instruction at address >A122 would be executed next. However the load interrupt in this case causes the program to continue at address >602C with a new workspace at >8300. Addresses >FFFC and >FFFE must, of course,have been loaded before the interrupt occurred. Those addresses can be set with a debugger or from a running program. continued on page 24

Jenny's Younger Set

Dear Jenny,

The three programs I send you are Good King Wenceslas in BASIC, the New Year's program in Extended BASIC and Anna Lee, Red River Valley and Amazing Grace in BASIC. I have scored 30 meteors intercepted with an average of 24 in Meteor. Is this a Younger Set Hall of Fame high score? With the New Year's program, the idea would be to let the "Press any key" message come up just before midnight and press the key on midnight on New Year's Day. However, wish every one reading this a very Merry Christmas.

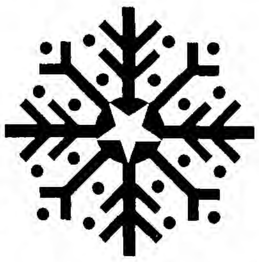
Vincent Maker

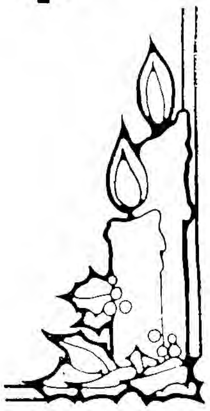
Spot on Vincent! You are certainly the Hall of Famer for Meteor. Thanks for the programs for the Christmas period. It is a pity that others are not able to think ahead and to send in something for their magazine.

```
100 REM GOOD KING WENCESLAS
110 REM BY VINCENT MAKER.
120 CALL CLEAR
130 CALL SCREEN(1)
140 FOR T=2 TO 8
150 CALL COLOR(T,8,2)
160 CALL SCREEN(1)
170 NEXT T
180 DISPLAY :"********************* *GOOD KING
    WENCESLAS* *********************"
190 DISPLAY :"THIS IS AN ENGLISH TRADITIONAL
    CAROL,WHICH ALTHOUGH MAKES NO MENTION OFCHRISTMAS
    IN"
200 DISPLAY :"ITS WORDS,HAS BEEN REVERED BY CAROLLERS
    FOR MANY YEARS."
210 PRINT
220 PRINT "THE FIRST VERSE"
230 PRINT
240 PRINT "GOOD KING WENCESLAS LOOKED OUT, ON THE FEAST
    OF STEVEN, WHEN THE SNOW LAY ROUNDABOUT, DEEP AND
    THICK AND "
250 PRINT "EVEN.  BRIGHTLY SHONE THE MOON THAT NIGHT,
    THOUGH THE FROST WAS CRUEL, WHEN A POOR MAN CAME IN
    SIGHT,"
260 PRINT "GATHERING WINTER FUEL."
270 FOR T=1 TO 2
280 CALL SOUND(997,262,0)
290 CALL SOUND(250,294,0)
300 CALL SOUND(498,262,0)
310 CALL SOUND(250,196,0)
320 CALL SOUND(250,220,0)
330 CALL SOUND(250,196,0)
340 CALL SOUND(250,220,0)
350 CALL SOUND(250,247,0)
360 CALL SOUND(498,262,0)
370 NEXT T
380 CALL SOUND(250,784,0)
390 CALL SOUND(250,698,0)
400 CALL SOUND(250,659,0)
410 CALL SOUND(250,587,0)
420 CALL SOUND(250,659,0)
430 CALL SOUND(250,294,0)
440 CALL SOUND(250,262,0)
450 CALL SOUND(250,220,0)
460 CALL SOUND(250,262,0)
470 CALL SOUND(250,220,0)
480 CALL SOUND(250,247,0)
490 CALL SOUND(498,262,0)
500 CALL SOUND(498,196,0)
510 CALL SOUND(250,220,0)
520 CALL SOUND(250,247,0)
530 CALL SOUND(498,262,0)
540 CALL SOUND(250,294,0)
550 CALL SOUND(250,392,0)
560 CALL SOUND(250,349,0)
570 CALL SOUND(250,330,0)
580 CALL SOUND(250,294,0)
590 CALL SOUND(250,262,0)
600 CALL SOUND(250,349,0)
610 CALL SOUND(498,262,0)
620 END
```

```
100 REM ********************
110 REM *NEW YEAR'S PROGRAM*
120 REM *                  *
130 REM *BY VINCENT MAKER  *
140 REM *                  *
150 REM * 1988/89          *
160 REM ********************
170 DISPLAY AT(7,1):"THE NEW YEAR'S PROGRAM"
180 CALL CLEAR
190 DISPLAY AT(9,1):"BY V. MAKER."
200 PRINT "WHENEVER YOU ARE READY TO RUN THE NEW YEAR'S
    PROGRAM,PRESS ANY KEY"
210 CALL KEY(0,K,L)
220 IF L=0 THEN 210
230 CALL CLEAR
240 CALL MAGNIFY(2)
250 FOR T=0 TO 8
260 CALL COLOR(T,3,2)
270 CALL SCREEN(2)
280 NEXT T
290 DISPLAY AT(3,1):"*HAPPY NEW YEAR*"
300 CALL SPRITE(#1,49,16,50,50,0,0)
310 CALL SPRITE(#2,57,12,50,60,0,0)
320 CALL SPRITE(#3,56,13,50,72,0,0)
330 CALL SPRITE(#4,57,11,50,85,0,0)
340 DISPLAY AT(11,1):"LET OLD ACQUAINTANCE BE FORGOT,
    AULD LANGS AYE!!"
350 GOTO 350
```

```
100 REM *************
110 REM *RED  RIVER *
120 REM *VALLEY AND *
130 REM *AURA LEE   *
140 REM *AND AMAZING*
150 REM *GRACE BY   *
160 REM *VINCENT    *
170 REM *MAKER      *
180 REM *************
190 CALL CLEAR
200 DISPLAY :"1.AURA LEE                   2.RED RIVER
    VALLEY          3.AMAZING GRACE            4.ALL"
210 DISPLAY :"5.END"
220 CALL KEY(0,A,S)
230 IF S=0 THEN 220
240 IF A=49 THEN 310
250 IF A=50 THEN 650
260 IF A=51 THEN 1070
270 IF A=52 THEN 310
280 IF A=53 THEN 290
290 STOP
300 GOTO 220
310 FOR T=1 TO 2
320 CALL SOUND(250,196,0)
330 CALL SOUND(250,262,0)
340 CALL SOUND(250,247,0)
350 CALL SOUND(250,262,0)
360 CALL SOUND(250,294,0)
370 CALL SOUND(250,220,0)
380 CALL SOUND(500,294,0)
390 CALL SOUND(250,262,0)
400 CALL SOUND(250,247,0)
410 CALL SOUND(250,220,0)
420 CALL SOUND(250,247,0)
430 CALL SOUND(1000,262,0)
440 NEXT T
450 CALL SOUND(2250,330,0)
460 CALL SOUND(250,294,0)
470 CALL SOUND(250,262,0)
480 CALL SOUND(250,294,0)
490 CALL SOUND(1000,330,0)
500 CALL SOUND(250,330,0)
510 CALL SOUND(250,330,0)
520 CALL SOUND(250,349,0)
530 CALL SOUND(250,330,0)
540 CALL SOUND(250,294,0)
550 CALL SOUND(250,220,0)
560 CALL SOUND(250,294,0)
570 CALL SOUND(250,262,0)
580 CALL SOUND(250,262,0)
590 CALL SOUND(250,247,0)
```





continued on page 17

The Geneve is here, finally

part 4, by Jerry Boyer, USA

Well, this month I have decided to talk about some of the quirks that I have come up against with my Geneve 9640.
Some programs (not very many) do not work with the Geneve as they do with the TI99/4A. Funnel Writer or Funnelweb Utilities are some that lock up the Geneve and you have to shut down and restart all over again. TI-Artist and Graphx are two others that do not work. This might seem like a tragedy but it is not. The My-Word word processor and My-Art drawing package for the Geneve are far superior programs and easier to use. The programs written for the TI99/4A that use special memory mapping are the ones that lock up the Geneve. All the other programs as well as the modules saved to disks work extremely well, some are better in the Geneve mode than in the TI99/4A mode, as well as much faster.
There are some irritating quirks though. To load the My-Word program you first have to load M-DOS. Then you have to load the GPL interpreter program, and the you are ready to load the My-Word loader program which then loads the My-Word word processor. Myarc claims that in the future you will be able to load My-Word directly from the M-DOS but not now. Also when you switch back from GPL mode to M-DOS mode you reverse the functions of the CAPS LOCK and the NUM LOCK buttons. This can be very annoying. I have found that if you press the CTRL and the SHIFT keys then press the CAPS LOCK or the NUM LOCK keys you can correct the reversed situation. Another annoying feature is that if one of your programs locks up the Geneve and you have to shut it off and start over again, you just messed up the clock and date settings in your card. You then have to manually reset the correct date and time in the M-DOS mode. The first couple of times this happened I thought the battery in the card was defective but it was not. Another is the fact that as of now you cannot change the color of the screen in M-DOS mode. In My-Word you have 6 screen color choices. It would be nice to have others, especially if you have a composite monitor as i do. In the 80 column mode the letters bleed over a little. Not bad, just a little annoying at times.
Well, after 5 months enjoying all the benefits and speed of the Geneve, the above is all I can find fault with the Geneve. Not to bad I think. The Geneve is the best thing going right now to keep the TI system alive. You can run 99% of your programs and 100 % of your TI modules plus you now have the power to run large programs like the larger PC computers run with much better graphics and speed. It is a giant step up for the TI computer. The Geneve is well worth its price.

Part 5

Well, this month I have received the first mailing of the finished software from Myarc. It included the final versions of M-DOS v1.01; My-Word word Processor v1.10; and GPL Interpreter v.99 (needed to run the TI99/4A modules saved to disk). Also there was a Multiplan update v1.0 and a new Cartridge Saver program and a 13 page set of addenda that cover these programs. On the last page they state that the rest of the completed software will be scheduled for mailing by the 1st April. This is to include the long awaited Advanced BASIC; Pascal package; and any updates to the previously released software. I wonder why they picked April 1st? I hope it is not an April Fool's joke on us.
The Multiplan update is an 80 column by 26 line display, with 41K of memory set aside for data only. This will allow some very involved spread sheet set up, a definite upgrade. Also they have increased the speed to run at full Geneve speed which is now up to 5 times that of the TI99/4A. The calculations work very fast now, not long and drawn out like on the TI99/4A. Really neat.
The new M-DOS and GPL are fully compatible with all RS232, floppy disk controller cards, and the horizon RAMdisk. Please note that the Geneve will not currently operate with Myarc's own 512K RAMcard. The Geneve will handle up to 2 Horizon RAMcards set at CRU bases of 1400 and/or 1600 and have to be accessed as DSK6 and/or DSK7. The Horizon RAMdisks must be reformatted with DM1000 (not DM2 or DMIII) before it will operate correctly with the Geneve. It would be wonderful to be able to have a RAMdisk that could be set up as DSK1, so you could have the operating disk (M-DOS) and the GPL boot up immediately. None as yet are out on the market. GrandRAM from DataBioTics (expected out in February) is supposed to be compatible and be able to be set up as DSK1.
Included on the GPL disk is the upgraded Disk Manager III v2.10 set up for the Geneve. This version will now do single pass copying of a diskette and one pass copying of up to a 96K file. The DMIII will only run on the Geneve and only with a Myarc Disk Controller Card. My only gripe about it is that 18 sectors per track is the default of the formatting mode. I and most of the TI99/4A users around the world use only single density as there are less problems than with double density when your disks have to be read on someone else's machine. DS/SD (720 sectors) seems to be the standard format other user groups have for exchanging disks.
A great pile of software and hardware are scheduled for release soon. They include a 1.5Megabyte memory expansion from Myarc which will have up to 1 Megabyte of RAMdisk that is battery backed, and Videoflex and FrameGrabber from Miller Communications. These cards will be able to use signals from your VCR to print and use digitized pictures of anything your VCR can output. (Courtesy of LeHigh 99'er March 1988 page 2)

Part 6

Well, this month I have received an updated version of the Myarc My-Art system disk v1.4 dated March 1988. Also on the disk were 3 new pictures in the high resolution mode. They are: DRAGON; UNICORN; WOLF. These drawings are so good that they look like photographs. Along with the new disk were 13 pages of instructions showing how to use the new commands in addition to all of the regular My-Art commands. From what I can tell, the only new features are :
CTRL[H] = creates a horizontal mirror image of your picture
CTRL[V] = creates a vertical mirror image of your picture
CTRL[S] = deletes last letter of text entered with the Text command if the enter key has not been pressed
You can get the updated My-Art v1.4 by sending a photocopy of your My-Art package original sales receipt or invoice and $10 to:

Myarc Inc.
2624 Rainier Drive, NE
BIRMINGHAM, AL. 35215, USA

Now I think I will show some of the features most used in the M-DOS mode.

CTRL[C] = breaks the operation in progress
CTRL[S] = pauses the operation in progress
CLS = clears the screen
DIR = shows the directory of disk in DSK1.
DIR B: = shows the directory of disk in DSK2.
DIR >PRN = prints out directory of disk in DSK1.
DIR B: >PRN = prints out directory of disk in DSK2.
TYPE "filename" /M = prints D/V 80 file to screen with a pause feature using a space bar.
COPY "filename" PRN = prints D/V 80 file out to printer
COPY CON PRN = lets you print from console to printer directly.
CTRL[Z] = stops the above command and returns to M-DOS.
COPY CON "filename" = creates a D/V 80 file from console.
CTRL[Z] = stops the above command and returns to M-DOS.
COPY A:"filename" B:"filename" = copies file from DSK1. to DSK2.
DISKCOPY A: B: = copies DSK1. to DSK2. in one pass up to 720 sectors .
DATE = allows you to change the date in memory if incorrect.

continued on page 12

Multiplan Machinations

by Bill Harms, ROM, USA

In this article I will introduce you to a method to transfer data from a BASIC program to Multiplan. I use Multiplan to keep my budget and to estimate income taxes. I have a spreadsheet with 18 columns: 12 months, yearly total, year-to-date, weekly average, monthly average and two for taxes. Those last two have formulas to get various numbers from the spreadsheet. The rows include: pay, interest, expenses, loans and other. You can really do "WHAT-IFing" and "WHY-NOTing" with Multiplan.

I use a nice fast (I mean fast) Extended BASIC program I wrote to add all my transactions by category. Then I use a SYLK creator to quickly and correctly prepare them for loading into my Multiplan spreadsheet. SYLK (Symbolic Link) files are a little known feature of Multiplan. They can be written to disk by a BASIC program and read by Multiplan.

This material is based on a program I got from TI, a series of articles in the May (and later) 1985 SUPER 99 MONTHLY (now called THE SMART PROGRAMMER) and the Multiplan manual.

This bare bones program is based on the one I received from TI in 1984. The disclaimer was bigger than the program! It writes a disk file with a one cell spreadsheet that can be read by Multiplan.

```
100 OPEN #1:"DSK1.SYLKF",DISPLAY,OUTPUT,FIXED 128
110 CALL CLEAR
120 INPUT "ROW NUMBER: ";R$
130 INPUT "COLUMN NUMBER: ";C$
140 INPUT "CELL CONTENT: ";A$
150 FOR Q=1 TO 27-LEN(A$)
160 W$=W$&CHR$(0)
170 NEXT Q
180 X$=CHR$(34)&A$&CHR$(34) ! SURROUNDS CONTENTS WITH
    QUOTES
190 Z$=CHR$(13)&CHR$(10) !CARRIAGE RETURN AND LINE FEED
200 Y$="D;PMP"&Z$&"F;D606B"&Z$&"B;Y"&R$& ";X"&C$&Z$&
    "C;K"&X$&Z$&"W;N1;A1 1"&Z$&"D"&Z$&W$
    (This monster of a line has the symbolics needed
    for Multiplan to read the file. See page 205 of
    the Multiplan manual for explanations ).
210 PRINT #1:Y$
220 CLOSE #1
230 END
```

If you enter and run this program, you will find a file on your disk called "SYLKF". Before you can load this file, you must change it. It may seem a bit odd, but the file must be written as DISPLAY, FIXED 128 and then changed to INTERNAL, FIXED 128 in the file header. In other words, the file must use DISPLAY notation but must look like an internal file. There are two ways to do this. You will find an application of Barry Traver's RAW (read and write) in SUPER 99 MONTHLY. Or you can use Advanced Diagnostics to change the four hex characters of the first line of the file header to (0202).

Once you have done this you can load your file. First, boot Multiplan. Press (T)ransfer and then (O)ptions. Next press (S)ymbolic and then ENTER. Now press (T)ransfer again and this time (L)oad your file.
Here is the Multiplan spreadsheet

```
&-------------&
&             &
&        1    &
&  1  HARMS   &
&             &
&             &
&-------------&
```

This is what the data looks like on disk using Miller's Graphics great ADVANCED Diagnostics

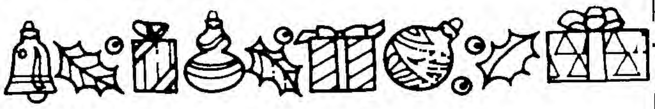
```
Drive : 2
Track : 3
Side : 1
Sector : 34
Byte : 0
Display : ASCII

ID;PMP$$F;D606
B$$B;Y1;%1$$C;
K"HARMS"$$W;W1
;A1 1$$E$$$$$$
```

Most of the $$'s stand for CR/LF (Z$ in line 190).

There are many ways you could input data besides the single INPUT in line 140. You could read data from DATA statements or from a disk file. That disk file could be created by almost anything: TI-Writer, RS232, another module or a Multiplan Print File.

You can create data in BASIC and then "dump" it into a spreadsheet 'en masse' instead of just keyboarding it. You can transmit the output SYLK file of your Multiplan Spreadsheet to others via RS232. The DIF (Data Interchange Format) used by Lotus 1-2-3 and Visicalc only accommodates the cell content, not the sheet parameters.

This is only a taste (bad?) of what you can create to load data into Multiplan. It really opens up Multiplan to other software. o

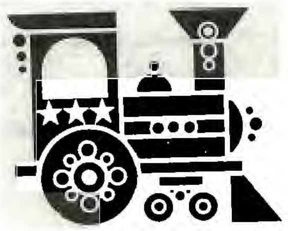


continued from page 11

FORMAT /V = formats disk in DSK1. (it asks for the new disk name)
TIME = allows you to set the correct time of the computer clock

One of the best features of M-DOS is that you can set up a D/V 80 file called AUTOEXEC to include your most used commands and put them in a menu which you can execute with a single letter input. AUTOEXEC automatically loads and executes as soon as M-DOS is booted up. This makes handling the most used commands a simple task.
My AUTOEXEC file is :

```
ECHO OFF
ECHO
ECHO
ECHO
ECHO
ECHO MENU
ECHO ====
ECHO
ECHO ============
ECHO A.  G.P.L.
ECHO B.  MY-ART
ECHO C.  M-DOS
ECHO D.  DIR B
ECHO E.  PRN DIR B
ECHO F.  FORMAT B
ECHO G.  COPY DISK
ECHO
ECHO ( type "N" space and selected letter )
```



Along with my AUTOEXEC file I have a D/V 80 file called "N" on my M-DOS boot disk. I have GPL also on the same disk. You need a DS/SD with 720 sectors for a boot up disk to get it all on. My "N" file looks like this.

```
IF %1==A DSK1.GPL
IF %1==B DSK2.MY-ART
IF %1==C ECHO M-DOS GO AHEAD
IF %1==D DIR B:
IF %1==E DIR B: >PRN
IF %1==F FORMAT B: /V
IF %1==G DISKCOPY A: B:
```

continued on page 20

A Solution to Hitchhiker's Guide to the Galaxy, Version 1.0

Presented by Sgt. Pepper and General Boy,
courtesy of OSB Systems

*Note: These are the exact commands needed to solve the version of Infocom's "Hitchhiker's Guide to the Galaxy" cracked by Fineous Fingers of The 1200 Club. These are not hints!

First, a bit of explanation. The reason this tutorial only works on versions of "Hitchhiker's Guide" cracked by FF is because this solver often utilizes several commands ("$FO", for example), that only work on test copies, which is what the cracked copy is. these '$' commands may or may not work on an original. The reason these commands are used in the solver is simply because the cracked game still has several bugs in it that prevent solving it smoothly. Therefore, these '$' commands are used to avoid random situations in the game which may negatively alter the outcome of the adventure. When a way to solve the game without using these special commands is found, this tutorial will be rewritten and re-distributed.

Commands are enclosed in quotes and should be typed in exactly as they are seen.

Directional movements are enclosed in brackets.

Comments are preceded by "->" or enclosed in parentheses.

Saving the game frequently is strongly recommended.

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

-> You (Arthur Dent) wake up in your bed on this Thursday, the worst day of your life.

"TURN ON LIGHT", "GET UP", "GET GOWN", "WEAR GOWN", "OPEN POCKET", "GET ASPIRIN AND FLUFF", "GET SCREWDRIVER AND TOOTHBRUSH", [S], "GET MAIL", [S], "BLOCK BULLDOZER"

-> You are now lying in the mud in front of the bulldozer, protecting your house from sure destruction. Your friend, Ford Prefect will appear and offer you a towel. Ignore him and keep waiting.

"WAIT" (several times until Prosser takes your place in the mud), [S,W], "EXAMINE SHELF", "BUY SANDWICH", "DRINK BEER", "DRINK BEER", "DRINK BEER", [E], "FEED DOG", "WAIT" (until the Vogon ships arrive and Ford drops his device), "GET DEVICE", "PUSH GREEN"

-> You are now in the "dark", aboard the Vogon constructor ship.

[E], "SMELL", "SMELL", "SMELL", "SMELL", "EXAMINE SHADOW", "EAT PEANUTS", "REMOVE GOWN", "HANG GOWN ON HOOK", "GET TOWEL", "COVER DRAIN WITH TOWEL", "GET SATCHEL", "PUT SATCHEL IN FRONT OF PANEL", "PUT MAIL ON SATCHEL", "PUSH BUTTON"

-> You should now have a babel fish in your ear.

"TURN SWITCH" (and note which word <first, second, or third> you will need from the Captain's favorite poem to open the case), "GET GOWN AND TOWEL", "WEAR GOWN", "WAIT" (until you are in the poetry appreciation chairs and the Captain is reading his poetry), "ENJOY POETRY", "WAIT" (until the Captain reads the second verse, and note the word you will need to type to open the vector plotter case), "WAIT" (until you are thrown into back into the hold while Ford tries to talk the guard into a sudden career change), "TYPE "XXXXX"" (the word from the poem), "GET PLOTTER", "WAIT" (until you and Ford are thrown into the airlock and blown out into space)

-> You and Ford are now aboard "The Heart of Gold", a new spaceship powered by the newly invented Improbability Drive.

[E], "HEAR", "HEAR", "HEAR", "HEAR", [S,S], (Ford will lead you to the bridge), "DROP PLOTTER AND GUIDE AND DEVICE", "GET CHISEL", [D,S,S,S,S,S,S], "LOOK", "LOOK", "GET PLIERS AND RASP", [N,D], "DROP CHISEL AND PLIERS AND RASP AND SCREWDRIVER AND TOOTHBRUSH", [U,S], "GET DRIVE", [N,N,W], "PUSH PAD", "GET CUP", [E,U], "DROP CUP AND DRIVE", "PLUG SHORT CORD INTO SMALL RECEPTACLE", "PLUG LONG CORD INTO LARGE RECEPTACLE", "PUT DANGLY BIT IN CUP", "$TR"

-> The '$TR' command has now placed you once again in "dark"... But you are now in the lair of the notorious Bugblatter beast of Traal.

[E], "SMELL", "SMELL", "SMELL", "SMELL", "EXAMINE SHADOW", "SAY "ARTHUR DENT"", [E], "GET STONE", "COVER HEAD WITH TOWEL", "CARVE "ARTHUR DENT" IN MEMORIAL", "REMOVE TOWEL", [W,SW], "GET INTERFACE", "DROP STONE", "WAIT" (until the men arrive, take you away, and after you are in "dark"...)

-> Surprise! You are back on The Heart of Gold!

[E], "HEAR", "HEAR", "HEAR", "HEAR", [S,S,W], "OPEN PANEL", "GET BOARD", "DROP BOARD", "PUT INTERFACE IN PANEL", "CLOSE PANEL", [E,S,D], "DROP CHIPPER", "$ZA"

-> Back in "dark", but you will soon take the part of Zaphod Beeblebrox on a very special mission.

[E], "SEE", "SEE", "SEE", "SEE", "EXAMINE LIGHT", "LOOK UNDER SEAT", "GET ALL", "STEER TOWARD CLIFF", "WAIT" (until the boat comes to a halt south of the ceremonial dias), "GET UP", [N], "WAIT" (UNTIL TRILLIAN COMES FORWARD AND PUTS A GUN TO YOUR HEAD), "GUARD, DROP RIFLES", "TRILLIAN, SHOOT RIFLES", [E]

-> Ta da! Back to The Heart of Gold!

[E], "HEAR", "HEAR", "HEAR", "HEAR", [S,S,S,D], "GET FLUFF AND KEY", "OPEN BOX WITH KEY", "DROP KEY", "$FO"

-> Now you will become Ford on earth, going through the same motions from the start of the game.

[E], "SEE", "SEE", "SEE", "SEE", "EXAMINE LIGHT", [N], "OPEN SATCHEL", "GIVE FLUFF TO ARTHUR", "OFFER TOWEL", TO ARTHUR", "IDIOT", "GO TO PROSSER", "PROSSER, LIE IN MUD", [S,W], "BUY BEER", "DRINK BEER", "DRINK BEER", [E], "GET DEVICE", "WAIT" (until the Vogon ships arrive and Arthur gets the thumb and presses the button)

-> "Dark", yet again. Back on the good ol' Heart of Gold.

[E], "HEAR", "HEAR", "HEAR", "HEAR", [S], "I" (you should have the satchel fluff in the pocket of your gown), "GET SATCHEL FLUFF", $PA

-> This time you become Trillian at a party on earth.

[E], "FEEL", "FEEL", "FEEL", "FEEL", "DRINK LIQUID", "OPEN BAG", "EXAMINE ARTHUR", "DROP GLASS", "GET FLUFF", "PUT FLUFF IN BAG", "GET GLASS", "WAIT" (or do what you can to avoid the hostess until "Phil" approaches you and takes you outside)

*Note: during this stage of the game do not talk to the hostess or an error will occur, and the game will end.

-> Getting used to "dark"? Do not worry, you will only have to do this a few more times.

[E], "HEAR", "HEAR", "HEAR", "HEAR", [S,S,U], "OPEN BAG", "GET TWEEZERS AND JACKET FLUFF", [D,S,D], "DROP TWEEZERS", "$FL"

-> This time you are actually Arthur, but you are now aboard an alien spaceship headed for earth.

[E], "HEAR", "HEAR", "HEAR", "HEAR", [S], "GET AWL", "LISTEN", "WAIT" (until after you have passed earth and the dog)

co · · on page 29

```
100 REM COUNTRY STILE BINGO
BY RC 6/4/83
110 RANDOMIZE :: OPTION BASE
 1 :: DIM A$(75),E$(5),B$(15
),I$(15),N$(15),G$(15),O$(15
)
120 FOR I=1 TO 8 :: CALL COL
OR(I,2,11):: NEXT I :: CALL
COLOR(9,5,10):: CALL COLOR(1
0,16,13)
130 CALL COLOR(12,7,16):: CA
LL COLOR(13,7,16):: CALL COL
OR(9,7,16):: CALL CHAR(96,RP
T$("F",16)):: CALL CHAR(97,"
0")
140 CALL CHAR(120,"00001F1F0
E0E0E0F0F0E0E0E1F1F0000F
0F838383838F0F038383838F8F")
150 CALL CHAR(124,"000007070
30303030303030303070700000C
0C08080808080808080808080C0C")
160 CALL CHAR(128,"00001E1E0
F0F0F0F0F0D0D0C0C0C1E1E00007
8783030B0F0F0F0F0F0F0707878"
)
170 CALL CHAR(132,"00000F1F1
F1C1C1C1C1C1C1C1C1F1F0F0000E
0F0F030300000F8F87070F0F0E")
180 CALL CHAR(136,"00000F1F1
C1C1C1C1C1C1C1C1C1F0F0000E
0F0707070707070707070F0E")
190 CALL CHAR(64,"3C4299A1A1
99423C"):: CALL CHAR(33,"000
000FFFF000000000000FFFFC0C0C
0000000FFFF030303")
200 CALL CHAR(36,"0000000000
00FFFFFFFF000000000000C0C0C0
C0C0C0C0C00303030303030303")
210 C1$="0C18002020002123267
EFFFEFEFF6160000000000F8FC2
62323FF232326FCF8"
220 C2$="00183000000002123267
EFEFEFEFF6160000000000F8FC0
68B5323538B06FCF8"
230 CALL MAGNIFY(3):: CALL C
HAR(140,C1$):: CALL CHAR(104
,C2$)
240 GOSUB 540 :: DISPLAY AT(
13,1):" SPEEK AND PLAY BINGO
 BY RC" :: DISPLAY AT(24,8):
"PRESS ANY KEY"
250 DISPLAY AT(4,8):"COUNTRY
 STYLE"
260 GOSUB 1410 :: GOSUB 560
:: GOSUB 540 :: CALL DELSPRI
TE(ALL)
270 CC$="TEN     ELEVEN  TWE
LVE  THIRTEENFOURTEENFIFTEEN
 6 TEEN  7 TEEN  8 TEEN  9 T
EEN"
280 DISPLAY AT(11,1):"DO YOU
 WISH TO USE A PRINT  DEVICE
? (Y/N) N"
290 ACCEPT AT(12,15)VALIDATE
("YN")SIZE(-1):P$ :: IF P$="
N" THEN 310 :: GOSUB 540
300 DISPLAY AT(11,1):"DEVICE
? TP.S" :: ACCEPT AT(11,9)SI
ZE(-20):D$ :: OPEN #1:D$,OUT
PUT
310 GOSUB 540 :: DISPLAY AT(
11,1):"INITIALIZING... PLEAS
E WAIT."
320 CALL CHAR(100,"FF0000FF0
000FF0000FF0000FF0000FFFF000
0FF0000FF0000FF0000FF0000FF"
)
330. CALL CHAR(116,RPT$("7F",
16)&RPT$("FE",16))
340 E$(1)="00070404070404070
080404080404080" :: E$(2)="0
00301010101010300800000000000
080"
350 E$(3)="00040605050404040
040404040C0C040" :: E$(4)="0
0030404040404030080400C0404
080"
360 E$(5)="00030404040404030
0804040404080"
370 FOR I=1 TO 15 :: B$(I)=S
EG$(STR$(I)&" ",1,2):: I$(I)
=STR$(I+15):: N$(I)=STR$(I+3
0):: G$(I)=STR$(I+45):: O$(I
)=STR$(I+60)
380 CALL PATTERN(#11,104+32*
(I-INT(I/2)*2)):: NEXT I ::
FOR I=1 TO 75 :: A$(I)=STR$(
I):: NEXT I
390 IF M$<>"1" THEN 400 ELSE
 M$="0" :: CALL DELSPRITE(AL
L):: GOSUB 540 :: CALL COLOR
(8,2,8):: DISPLAY AT(11,1):"
INITIALIZATION TAKING PLACE.
"
400 RN=1 :: FOR I=1 TO 75 ::
 N=INT(RND*75)+1 :: J=INT(RN
D*75)+1 :: S$=A$(N):: A$(N)=
A$(J):: A$(J)=S$ :: NEXT I
410 CALL SPRITE(#1,100,1,246
,193,0,0,#2,100,1,246,193,0,
0,#3,100,1,246,193,0,0,#4,10
0,1,246,193,0,0)
420 FOR I=1 TO 8 :: CALL COL
OR(I,2,8):: NEXT I :: CALL C
HAR(58,"0")
430 REM MAIN MENU******
440 GOSUB 540 :: DISPLAY AT(
4,5):"COUNTRY STYLE BINGO"
450 CALL HCHAR(2,6,36,22)::
CALL HCHAR(6,6,37,22):: CALL
 VCHAR(3,6,38,3):: CALL VCHA
R(3,27,39,3)
460 DISPLAY AT(8,2):"PRESS
   FOR":" ":"  1   AUTOMATIC
 GAME MODE":"  2   MANUAL GA
ME MODE"
470 DISPLAY AT(12,2):" 3   P
RINT BINGO CARDS":"  4   PRI
NT CALL LIST":"  5   EXIT"
480 GOSUB 560 :: IF (KEY>48)
AND(KEY<54)THEN 500
490 CALL SOUND(100,294,2)::
GOTO 480
500 IF (KEY=51 OR KEY=52)AND
 P$<>"Y" THEN 490
510 GOSUB 540 :: MENU=KEY-48
 :: ON MENU GOSUB 580,580,11
30,1290,530
520 ON MENU GOTO 390,390,430
,430
530 CALL CLEAR :: STOP
540 REM SCREEN CLEAR***
550 CALL CLEAR :: CALL SCREE
N(5):: CALL VCHAR(1,1,96,48)
:: CALL VCHAR(1,31,96,48)::
RETURN
560 REM CALL KEY***
570 CALL KEY(0,KEY,ST):: IF
ST=0 THEN 570 :: RETURN
580 REM BEGIN GAME***
590 CALL MAGNIFY(4):: CALL C
OLOR(2,4,8):: CALL COLOR(3,2
,4):: CALL COLOR(4,2,4):: CA
LL COLOR(8,2,14)
600 CALL CHARPAT(61,S$):: CA
LL CHAR(64,S$):: CALL CHAR(1
04,RPT$("10",8))
610 CALL CHAR(91,"FFFF000000
00FFFF"):: CALL CHAR(92,RPT$
("C0",14)&"FFFF"&RPT$("03",1
4)&"FFFF")
620 FOR I=2 TO 23 :: CALL HC
HAR(I,4,58,16):: NEXT I :: F
OR I=7 TO 16 STEP 3 :: CALL
VCHAR(8,I,104,15):: NEXT I
630 CALL CHAR(108,"030F1F3F7
F7FFFFFC0F0F8FCFEFEFFFFFFF7
F7F3F1F0F03FFFFFEFEFCF8F0C")
640 CALL CHAR(40,"030F1F3F7F
7FFFFFC0F0F8FCFEFEFFFFFFFF7F
7F3F1F0F03FFFFFEFEFCF8F0C")
650 CALL HCHAR(2,4,40):: CAL
L HCHAR(2,19,41):: CALL HCHA
R(23,4,42):: CALL HCHAR(23,1
9,43)
660 FOR I=3 TO 6 :: CALL HCH
AR(I,5,97,14):: NEXT I
670 CALL HCHAR(3,5,108):: CA
LL HCHAR(3,18,109):: CALL HC
HAR(6,5,110):: CALL HCHAR(6,
18,111)
680 CALL HCHAR(4,5,120):: CA
LL HCHAR(4,6,122):: CALL HCH
AR(5,5,121):: CALL HCHAR(5,6
,123)
690 CALL HCHAR(4,8,124):: CA
LL HCHAR(4,9,126):: CALL HCH
AR(5,8,125):: CALL HCHAR(5,9
,127)
700 CALL HCHAR(4,11,128):: C
ALL HCHAR(4,12,130):: CALL H
CHAR(5,11,129):: CALL HCHAR(
5,12,131)
710 CALL HCHAR(4,14,132):: C
ALL HCHAR(4,15,134):: CALL H
CHAR(5,14,133):: CALL HCHAR(
5,15,135)
720 CALL HCHAR(4,17,136):: C
ALL HCHAR(4,18,138):: CALL H
CHAR(5,17,137):: CALL HCHAR(
5,18,139)
730 CALL HCHAR(3,26,33,2)::
CALL HCHAR(3,25,34):: CALL H
CHAR(3,28,35)
740 CALL VCHAR(4,25,38,4)::
CALL VCHAR(4,28,39,4):: CALL
 VCHAR(5,24,36):: CALL VCHAR
(5,29,36)
750 CALL VCHAR(6,24,92,2)::
CALL VCHAR(6,29,94,2):: CALL
 HCHAR(8,25,91,4)
760 CALL VCHAR(8,24,93):: CA
LL VCHAR(8,29,95)
770 CALL HCHAR(10,26,33,2)::
 CALL HCHAR(10,25,34):: CALL
 HCHAR(10,28,35):: CALL HCHA
R(15,25,37,4)
780 CALL VCHAR(11,25,38,4)::
 CALL VCHAR(11,28,39,4)
790 DISPLAY AT(17,21)SIZE(8)
:"PRESS" :: DISPLAY AT(18,18
)SIZE(11):"SPACE"&CHR$(64)&"
PAUSE"
800 DISPLAY AT(19,18)SIZE(11
):"ENTER"&CHR$(64)&"START" :
: DISPLAY AT(20,18)SIZE(11):
"SHIFT R"&CHR$(64)
801 DISPLAY AT(21,18)SIZE(11
):"REPEAT"
810 DISPLAY AT(22,18)SIZE(11
):"SHIFT V"&CHR$(64):: DISPL
AY AT(23,18)SIZE(11):"NEW GA
ME"
820 IF MENU=1 THEN 830 ELSE
GOSUB 1380 :: IF KEY=12 THEN
 1070
830 FOR J=1 TO 3 :: GOSUB 13
40 :: IF KEY=12 THEN 1070 EL
SE CALL SPRITE(#5,100,2,25,1
93,-2,0)
840 FOR I=1 TO 62 :: NEXT I
:: NEXT J :: CALL MOTION(#5,
0,0):: CALL LOCATE(#5,26,193
)
850 REM SELECT NUMBER***
860 N=VAL(A$(RN)):: S$=SEG$(
"BINGO",INT((N-1)/15)+1,1)&"
 " :: IF N>9 THEN 870 :: S$=
S$&A$(RN):: GOTO 920
870 IF N>19 THEN 890 :: S$=S
$&SEG$(CC$,(N-INT(N/10)*10)*
8+1,8):: GOTO 920
```

continued on page 17

```
100 REM *****************
110 REM *   GOLD RUSH    *
120 REM *****************
130 REM
140 REM 99'ER VERSION 2.2.1X
B-46
150 REM
160 CALL CLEAR :: CALL SCREE
N(16):: DIM DC(4)
170 RESTORE 2220 :: GOSUB 22
80 :: GOSUB 2280 :: GOSUB 22
80 :: GOSUB 2330
180 CALL CHAR(96,"FFFFFFFFF
FFFF00"):: CALL COLOR(9,2,5)
190 SM1=9 :: SM2=169 :: D1,D
3,D5=9 :: D2=193 :: D4=201 :
: D6=209 :: DC(2),DC(3),DC(4
)=10
200 CALL AR(0,0,0,3)
210 CALL CLEAR :: RESTORE 22
40 :: FOR X=1 TO 6 :: GOSUB
2280 :: NEXT X :: GOSUB 2300
 :: B=VAL(AN$)
220 D1$="UP" :: D2$="LEFT" :
: D3$="RIGHT" :: D4$="DOWN"
:: D5$="NOT THAT WAY" :: D6$
="TAKE DYNAMITE"
230 D7$="OK" :: D8$="NO DYNA
MITE" :: D9$="DROP DYNAMITE"
 :: D10$="LIGHT FUSE" :: D11
$="INPUT DIRECTION"
240 D12$="NO DYNAMITE..." ::
 D13$="CAN'T DO THAT" :: D14
$="FUSE BURNING" :: D15$="SH
AFT IS CLEAR"
250 D16$="GOLD!-VALUE$" :: D
18$="DIRT." :: D19$="SOLD RO
CK."
260 D20$="GRANITE" :: D21$="
FLOOD!" :: D22$="YOU'BEEN KI
LLED IN A" :: D23$="BLAST"
270 CALL CLEAR :: RESTORE 22
30 :: GOSUB 2280 :: RANDOMIZ
E
280 FOR X=1 TO B*30 :: P1=IN
T(RND*19)+1 :: P2=INT(RND*28
)+1 :: CALL AR(P1,P2,2,2)::
NEXT X
290 FOR X=1 TO B*20 :: P1=IN
T(RND*19)+1 :: P2=INT(RND*28
)+1 :: CALL AR(P1,P2,3,2)::
NEXT X
300 FOR X=1 TO B*5 :: P1=INT
(RND*18)+2 :: P2=INT(RND*27)
+1
310 CALL AR(P1,P2,5,2):: CAL
L AR(P1,P2+1,5,2):: NEXT X
320 P1=INT(RND*10)+10 :: P2=
INT(RND*26)+2
330 FOR FM=0 TO 2 :: NZ=INT(
RND*151)+100 :: CALL AR(P1,P
2+FM,NZ,2):: NEXT FM
340 P3=P1 :: P4=P2
350 FOR X=1 TO 10
360 P1=INT(RND*9)+P3-4 :: P2
=INT(RND*11)+P4-5
370 IF P1>19 OR P2<1 OR P2>2
8 THEN 360
380 CALL AR(P1,P2,Z,1):: IF
Z<4 THEN NZ=INT(RND*75)+26 :
: CALL AR(P1,P2,NZ,2)ELSE GO
TO 360
390 NEXT X
400 FOR X=1 TO 10
410 P1=INT(RND*10)+10 :: P2=
INT(RND*28)+1
420 CALL AR(P1,P2,Z,1):: IF
Z<4 THEN NZ=INT(RND*25)+10 :
: CALL AR(P1,P2,NZ,2)ELSE GO
TO 410
430 NEXT X
440 FOR X=1 TO 20
450 P1=INT(RND*15)+5 :: P2=I
NT(RND*28)+1
460 CALL AR(P1,P2,Z,1):: IF
Z<4 THEN NZ=INT(RND*25)+10 :
: CALL AR(P1,P2,NZ,2)ELSE GO
TO 450
470 NEXT X
480 FOR X=1 TO 30
490 P1=INT(RND*10)+2 :: P2=I
NT(RND*28)+1
500 CALL AR(P1,P2,Z,1):: IF
Z<4 THEN NZ=INT(RND*15)+10 :
: CALL AR(P1,P2,NZ,2)ELSE GO
TO 490
510 NEXT X
520 CALL CLEAR :: CALL HCHAR
(3,1,96,608):: CALL VCHAR(1,
31,117,96):: CALL COLOR(12,7
,16)
530 CALL CHAR(104,"06E6427E4
2470D090000001F7FDF000010080
43C3C3C")
540 CALL CHAR(112,"01071C74C
404040400000001071C70C000187
EC3")
550 CALL CHAR(115,"00000080E
0380E0380E0382E23202020FFFFF
FFFFFFFFFF")
560 CALL COLOR(11,13,1):: CA
LL CHAR(120,"1072401A0E604C0
1CB42598822449908")
570 CALL COLOR(13,13,1):: CA
LL CHAR(128,"7E42427E42427E4
2"):: CALL COLOR(14,5,5)
580 CALL HCHAR(2,24,112):: C
ALL HCHAR(2,28,116):: CALL H
CHAR(1,25,113)
590 CALL HCHAR(1,26,114):: C
ALL HCHAR(1,27,115)
600 CALL SPRITE(#2,106,2,D1,
D2,#3,106,2,D3,D4,#4,106,2,D
5,D6,#1,104,13,SM1,SM2)
610 RESTORE 2260 :: GOSUB 22
80
620 K=0 :: CALL KEY(0,K,S)::
 SP1=(SM1+7)/8-2 :: SP2=(SM2
+7)/8-2
630 DISPLAY AT(23,1):""
640 IF K=69 THEN GOSUB 730 :
: GOTO 620
650 IF K=83 THEN GOSUB 810 :
: GOTO 620
660 IF K=68 THEN GOSUB 900 :
: GOTO 620
670 IF K=88 THEN GOSUB 990 :
: GOTO 620
680 IF K=80 THEN 1050
690 IF K=76 THEN 1140
700 IF K=77 THEN 1810
710 IF K=15 THEN 2200
720 GOTO 620
730 A$=D1$ :: GOSUB 2320
740 IF SP1<1 THEN 770
750 IF SP1=1 THEN 780
760 CALL AR(SP1-1,SP2,Z,1)::
 IF Z=8 THEN 780
770 A$=D5$ :: GOSUB 2310 ::
RETURN
780 A$=D7$ :: GOSUB 2310 ::
SP1=SP1-1 :: SM1=SM1-8 :: CA
LL LOCATE(#1,SM1,SM2)
790 IF CAR=1 THEN CALL LOCAT
E(#CAR1,SM1,SM2)
800 RETURN
810 A$=D2$ :: GOSUB 2320
820 IF SP2=1 THEN 770
830 IF SP1=0 THEN 880
840 CALL AR(SP1,SP2-1,Z,1)::
 IF Z=7 OR Z=8 THEN 870
850 IF Z=6 THEN 870
860 A$=D5$ :: GOSUB 2310 ::
RETURN
870 IF SP1=19 THEN 880 ELSE
CALL AR(SP1+1,SP2,Z,1):: IF
Z=7 THEN 2190
880 SP2=SP2-1 :: SM2=SM2-8 :
: CALL LOCATE(#1,SM1,SM2)::
IF CAR=1 THEN CALL LOCATE(#C
AR1,SM1,SM2)
890 RETURN
900 A$=D3$ :: GOSUB 2320
910 IF SP2=28 THEN 770
920 IF SP1=0 THEN 970
930 CALL AR(SP1,SP2+1,Z,1)::
 IF Z=7 OR Z=8 THEN 960
940 IF Z=6 THEN 960
950 A$=D5$ :: GOSUB 2310 ::
RETURN
960 IF SP1=19 THEN 970 ELSE
CALL AR(SP1+1,SP2,Z,1):: IF
Z=7 THEN 2190
970 SP2=SP2+1 :: SM2=SM2+8 :
: CALL LOCATE(#1,SM1,SM2)::
IF CAR=1 THEN CALL LOCATE(#C
AR1,SM1,SM2)
980 RETURN
990 A$=D4$ :: GOSUB 2320
1000 IF SP1=19 THEN 770
1010 CALL AR(SP1+1,SP2,Z,1):
: IF Z<7 OR Z>8 THEN A$=D5$
:: GOSUB 2310 :: RETURN
1020 IF Z<>8 THEN GOTO 2190
1030 SP1=SP1+1 :: SM1=SM1+8
:: CALL LOCATE(#1,SM1,SM2)::
 IF CAR=1 THEN CALL LOCATE(#
CAR1,SM1,SM2)
1040 RETURN
1050 IF CAR=1 THEN 1100 ELSE
 A$=D6$ :: GOSUB 2320
1060 IF SM1=D1 AND SM2=D2 TH
EN CAR=1 :: CAR1=2 :: A$=D7$
 :: GOSUB 2310 :: D1,D2=0 ::
 GOTO 620
1070 IF SM1=D3 AND SM2=D4 TH
EN CAR=1 :: CAR1=3 :: A$=D7$
 :: GOSUB 2310 :: D3,D4=0 ::
 GOTO 620
1080 IF SM1=D5 AND SM2=D6 TH
EN CAR=1 :: CAR1=4 :: A$=D7$
 :: GOSUB 2310 :: D5,D6=0 ::
 GOTO 620
1090 A$=D8$ :: GOSUB 2310 ::
 GOTO 620
1100 A$=D9$ :: GOSUB 2320
1110 IF CAR1=2 THEN D1=SM1 :
: D2=SM2 :: CAR=0 :: GOTO 62
0
1120 IF CAR1=3 THEN D3=SM1 :
: D4=SM2 :: CAR=0 :: GOTO 62
0
1130 IF CAR1=4 THEN D5=SM1 :
: D6=SM2 :: CAR=0 :: GOTO 62
0
1140 A$=D10$ :: GOSUB 2320 :
: A$=D11$ :: GOSUB 2310
1150 IF CAR=0 OR DC(CAR1)=0
THEN A$=D12$ :: GOSUB 2310 :
: GOTO 620
1160 GOSUB 2330
1170 IF K=83 THEN 1210
1180 IF K=68 THEN 1260
1190 IF K=88 THEN 1290
1200 CALL SOUND(100,550,0)::
 GOTO 1160
1210 IF SP2<2 THEN A$=D13$ :
: GOSUB 2310 :: GOTO 620
1220 CALL AR(SP1,SP2-1,Z,1):
: IF Z=7 OR Z=8 THEN A$=D13$
 :: GOSUB 2310 :: GOTO 620
1230 CALL SPRITE(#5,105,7,SM
1,SM2-8):: DC(CAR1)=DC(CAR1)
-1
1240 P1=SP1 :: P2=SP2-1
1250 A$=D14$ :: GOSUB 2310 :
: GOTO 1320
1260 IF SP2>27 THEN A$=D13$
:: GOSUB 2310 :: GOTO 620
1270 CALL AR(SP1,SP2-1,Z,1):
: IF Z=7 OR Z=8 THEN A$=D13$
 :: GOSUB 2310 :: GOTO 620
```

```
1280 CALL SPRITE(#5,105,7,SM
1,SM2+8):: DC(CAR1)=DC(CAR1)
-1 :: P1=SP1 :: P2=SP2+1 ::
GOTO 1250
1290 IF SP1>18 THEN A$=D13$
:: GOSUB 2310 :: GOTO 620
1300 CALL AR(SP1+1,SP2,Z,1):
: IF Z=7 OR Z=8 THEN A$=D13$
 :: GOSUB 2310 :: GOTO 620
1310 CALL SPRITE(#5,105,7,SM
1+8,SM2):: DC(CAR1)=DC(CAR1)
-1 :: P1=SP1+1 :: P2=SP2 ::
GOTO 1250
1320 TD=5
1330 DISPLAY AT(22,17):TD ::
 GOSUB 1750
1340 TD=TD-1 :: IF TD>-1 THE
N 1330
1350 FOR EX=1 TO 10 :: CALL
PATTERN(#5,120):: CALL PATTE
RN(#5,121):: CALL SOUND(500,
INT(RND*-3)-4,0):: NEXT EX
1360 CALL DELSPRITE(#5):: IF
 ABS(SP1-P1)<3 AND ABS(SP2-P
2)<3 THEN 1740
1370 CALL AR(P1,P2,Z,1):: OR
E=Z
1380 IF K<>88 THEN 1400 ELSE
 CALL HCHAR(P1+2,P2+2,128)::
 CALL AR(P1,P2,8,2)
1390 CALL HCHAR(P1+1,P2+2,12
8):: CALL AR(P1-1,P2,8,2)::
GOTO 1410
1400 CALL HCHAR(P1+2,P2+2,32
):: CALL AR(P1,P2,7,2)
1410 IF ORE<4 THEN A$=D15$ :
: GOSUB 2310 :: GOTO 620
1420 IF ORE>8 THEN GOLD=GOLD
+ORE :: DISPLAY AT(24,21):GO
LD :: A$=D16$ :: GOSUB 2310
:: DISPLAY AT(22,24):ORE
1430 IF ORE<=8 THEN 1450
1440 RESTORE 2270 :: FOR CS=
1 TO 5 :: READ GS :: CALL SO
UND(200*CS,GS,0):: NEXT CS :
: GOTO 620
1450 IF ORE=4 THEN A$=D17$ :
: GOSUB 2310 :: GOTO 620
1460 CALL HCHAR(P1+2,P2+2,13
6)
1470 A$=D21$ :: GOSUB 2310 :
: CALL SOUND(1000,110,0,-8,0
):: GOSUB 1750
1480 CALL SOUND(500,110,0,-5
,0):: GOSUB 1750
1490 IF P2<=1 THEN 1500 ELSE
 CALL AR(P1,P2-1,Z,1):: IF Z
=7 OR Z=8 THEN 1520
1500 IF P2>27 THEN 1510 ELSE
 CALL AR(P1,P2+1,Z,1):: IF Z
=8 OR Z=7 THEN 1580
1510 CALL HCHAR(P1+2,P2+2,13
6):: GOTO 620
1520 P2=P2+1 :: CALL HCHAR(P
1+2,P2+2,136):: GOSUB 1750
1530 CALL AR(P1,P2,5,2)
1540 IF P1=SP1 AND P2=SP2 TH
EN 2190
1550 IF P1=19 THEN 1560 ELSE
 CALL AR(P1+1,P2,Z,1):: IF Z
=8 THEN GOTO 1640
1560 IF P2>0 THEN CALL AR(P1
,P2+1,Z,1):: IF Z=7 OR Z=8 T
HEN 1520
1570 CALL HCHAR(P1+2,P2+2,13
6):: GOTO 620
1580 P2=P2+1 :: CALL HCHAR(P
1+2,P2+2,136):: GOSUB 1750
1590 CALL AR(P1,P2,5,2)
1600 IF P1=SP1 AND P2=SP2 TH
EN 2190
1610 IF P1=19 THEN 1620 ELSE
 CALL AR(P1+1,P2,Z,1):: IF Z
=8 THEN GOTO 1640
1620 IF P2<28 THEN CALL AR(P
1,P2+1,Z,1):: IF Z=7 OR Z=8
THEN 1580
1630 CALL HCHAR(P1+2,P2+2,13
6):: GOTO 620
1640 P1=P1+1 :: CALL HCHAR(P
1+2,P2+2,136):: GOSUB 1750
1650 CALL AR(P1,P2,5,2)
1660 IF P1=SP1 AND P2=SP2 TH
EN 2190
1670 IF P1=19 THEN 1680 ELSE
 CALL AR(P1+1,P2,Z,1):: IF Z
=8 THEN 1640
1680 RANDOMIZE
1690 IF P2>27 OR P2<=1 THEN
1720
1700 CALL AR(P1,P2+1,Z,1)::
CALL AR(P1,P2-1,Z1,1):: IF (
Z=7 OR Z=8)AND(Z1=7 OR Z1=8)
THEN FL3=INT(RND*2)+1
1710 IF FL3>0 THEN ON FL3 GO
TO 1520,1580
1720 IF Z=7 OR Z=8 THEN 1580
 ELSE IF Z1=7 OR Z1=8 THEN 1
520
1730 CALL HCHAR(P1+2,P2+2,13
6):: GOTO 620
1740 DISPLAY AT(22,1):D22$:D
23$ :: GOTO 2200
1750 CALL KEY(0,K1,S1):: IF
S1=0 OR S=-1 THEN RETURN
1760 IF K1=69 THEN GOSUB 730
 :: RETURN
1770 IF K1=83 THEN GOSUB 810
 :: RETURN
1780 IF K1=68 THEN GOSUB 900
 :: RETURN
1790 IF K1=88 THEN GOSUB 990
 :: RETURN
1800 RETURN
1810 DISPLAY AT(23,1):"MINE
IN WHAT DIRECTION?"
1820 CALL KEY(0,KE,SE):: IF
SE=0 THEN 1820
1830 IF KE=83 THEN 1870
1840 IF KE=68 THEN 1970
1850 IF KE=88 THEN 2070
1860 CALL SOUND(100,550,0)::
 GOTO 1810
1870 DISPLAY AT(24,1)SIZE(16
):"MINE LEFT"
1880 IF SP2<2 OR SP1<1 THEN
CALL SOUND(50,220,5):: GOTO
620
1890 CALL AR(SP1,SP2-1,Z,1):
: IF Z<>1 THEN 1920 ELSE CAL
L AR(SP1,SP2-1,7,2):: CALL H
CHAR(SP1+2,SP2+1,32)
1900 SM2=SM2-8 :: SP2=SP2-1
:: CALL LOCATE(#1,SM1,SM2)::
 IF CAR=1 THEN CALL LOCATE(#
CAR1,SM1,SM2)
1910 A$=D18$ :: GOSUB 2310 :
: GOTO 620
1920 IF Z=2 THEN A$=D19$ ::
GOSUB 2310 :: GOTO 620
1930 IF Z=3 THEN A$=D20$ ::
GOSUB 2310 :: GOTO 620
1940 IF Z=7 OR Z=8 THEN DISP
LAY AT(22,1):"ALREADY MINED.
" :: GOTO 620
1950 CALL HCHAR(SP1+2,SP2+1,
32):: ORE=Z :: IF Z>8 THEN C
ALL AR(SP1,SP2-1,7,2)
1960 P1=SP1 :: P2=SP2-1 :: G
OTO 1420
1970 DISPLAY AT(24,1)SIZE(16
):"MINE RIGHT"
1980 IF SP2>27 OR SP1<1 THEN
 CALL SOUND(50,220,4):: GOTO
 620
1990 CALL AR(SP1,SP2+1,Z,1):
: IF Z<>1 THEN 2020 ELSE CAL
L AR(SP1,SP2+1,7,2):: CALL H
CHAR(SP1+2,SP2+3,32)
2000 SM2=SM2+8 :: SP2=SP2+1
:: CALL LOCATE(#1,SM1,SM2)::
 IF CAR=1 THEN CALL LOCATE(#
CAR1,SM1,SM2)
2010 A$=D18$ :: GOSUB 2310 :
: GOTO 620
2020 IF Z=2 THEN A$=D18$ ::
GOSUB 2310 :: GOTO 620
2030 IF Z=3 THEN A$=D19$ ::
GOSUB 2310 :: GOTO 620
2040 IF Z=7 OR Z=8 THEN DISP
LAY AT(22,1):"ALREADY MINED.
" :: GOTO 620
2050 CALL HCHAR(SP1+2,SP2+3,
32):: ORE=Z :: IF Z>8 THEN C
ALL AR(SP1,SP2+1,7,2)
2060 P1=SP1 :: P2=SP2+1 :: G
OTO 1420
2070 DISPLAY AT(24,1)SIZE(16
):"MINE DOWN."
2080 IF SP1=19 THEN CALL SOU
ND(50,220,5):: GOTO 620
2090 CALL AR(SP1+1,SP2,Z,1):
: IF Z<>1 THEN 2130 ELSE CAL
L AR(SP1+1,SP2,8,2):: CALL H
CHAR(SP1+3,SP2+2,128)
2100 SP1=SP1+1 :: SM1=SM1+8
:: CALL LOCATE(#1,SM1,SM2)::
 IF CAR=1 THEN CALL LOCATE(#
CAR1,SM1,SM2)
2110 A$=D18$ :: GOSUB 2310 :
: IF SP1<1 THEN GOTO 620
2120 CALL AR(SP1-1,SP2,8,2):
: CALL HCHAR(SP1+1,SP2+2,128
):: GOTO 620
2130 IF Z=2 THEN A$=D19$ ::
GOSUB 2310 :: GOTO 620
2140 IF Z=3 THEN A$=D20$ ::
GOSUB 2310 :: GOTO 620
2150 IF Z=7 THEN CALL AR(SP1
+1,SP2,8,2):: CALL HCHAR(SP1
+3,SP2+2,128):: GOTO 620
2160 IF Z=8 THEN DISPLAY AT(
22,1):"ALREADY MINED" :: GOT
O 620
2170 CALL VCHAR(SP1+2,SP2+2,
128,2):: ORE=Z :: IF Z>8 THE
N CALL AR(SP1+1,SP2,8,2):: C
ALL AR(SP1,SP2,8,2)
2180 P1=SP1+1 :: P2=SP2 :: G
OTO 1420
2190 DISPLAY AT(22,1):"YOUR
DEAD!" :: CALL SOUND(2000,22
0,0):: CALL SOUND(4000,110,0
)
2200 CALL DELSPRITE(ALL):: D
ISPLAY AT(23,13):"SCORE:";GO
LD :: DISPLAY AT(24,1):"PLAY
 AGAIN(Y/N)?"
2210 ACCEPT AT(24,17)VALIDAT
E("YN"):PA$ :: IF PA$="Y" TH
EN SCORE=0 :: CAR=0 :: GOTO
160 ELSE CALL CLEAR :: RUN "
DSK1.LOAD"
2220 DATA 2,9,GOLD RUSH,4,5,
BY CHRIS W GOOBER,24,1,PRESS
 ANY KEY TO BEGIN
2230 DATA 12,7,STANDBY PLEAS
E
2240 DATA 1,1,LEVEL OF DIFFI
CULTY:,3,3,1.MINE COOK,5,3,2
.MINER'S HELPER,7,3,3.APPREN
TICE MINER,9,3,4.MINER
2250 DATA 11,3,5.MINE FOREMA
N,24,1,YOUR CHOICE?,12345
2260 DATA 24,17,"GOLD"
2270 DATA 440,660,550,660,77
0
2280 READ XC,YC,A$ :: DISPLA
Y AT(XC,YC):A$ :: RETURN
2290 READ XC,YC,A$ :: DISPLA
Y AT(XC,YC):A$ :: ACCEPT AT(
XC,YC+LEN(A$)):AN$ :: RETURN
```

continued on page 31

continued from page 14

```
880 IF N>19 THEN 890 :: S$=S
$&SEG$(CC$,(N-INT(N/10)*10)*
8+1,8):: GOTO 920
890 S$=S$&SEG$("TWENTY THIRT
Y FORTY  FIFTY  SIXTY  SEVEN
TY",(INT(N/10)-2)*7+1,7)
900 IF N/10=INT(N/10)THEN 92
0 :: I=(N-INT(N/10)*10-1)*5+
1
910 S$=S$&" "&SEG$("ONE  TWO
  THREEFOUR FIVE SIX  SEVENE
IGHTNINE",I,5)
920 SY$=S$
930 REM DISPLAY NUMBER IN CA
GE***
940 S$=STR$(N)&" " :: CALL C
HARPAT(ASC(SEG$(S$,1,1)),S1$
,ASC(SEG$(S$,2,1)),S2$):: I=
INT((N-1)/15)+1
950 CALL COLOR(#7,1,#9,1,#10
,1):: CALL LOCATE(#7,25,194,
#9,25,193,#10,22,193)
960 CALL CHAR(140,SEG$(E$(I)
,1,16)&S1$&SEG$(E$(I),17,16)
&S2$)
970 CALL SPRITE(#6,116,8,26,
193,0,0,#7,140,2,25,194,0,0,
#8,116,8,22,193,0,0,#9,116,7
,25,193,0,0)
980 CALL SPRITE(#10,116,7,22
,193,0,0):: CALL MOTION(#6,-
5,0,#8,-5,0,#5,-5,0)
990 FOR I=1 TO 300 :: NEXT I
 :: CALL MOTION(#5,0,0,#6,0,
0,#8,0,0):: CALL SAY(SY$)
1000 CALL MOTION(#9,9,0,#7,9
,0,#10,9,0):: FOR I=1 TO 8 :
: GOSUB 1340 :: IF KEY=12 TH
EN 1070
1010 CALL MOTION(#9,9,0,#7,9
,0,#10,9,0):: NEXT I
1020 CALL MOTION(#9,0,0,#7,0
,0,#10,0,0):: CALL LOCATE(#7
,81,194,#9,78,193,#10,81,193
):: CALL SAY(SY$)
1030 CALL COLOR(#9,12,#10,12
):: S$="" :: IF N>9 THEN 104
0 :: S$=CHR$(58)
1040 S$=S$&STR$(N):: I=N-INT
((N-1)/15)*15+7 :: J=INT((N-
1)/15)*3+5
1050 CALL HCHAR(I,J,ASC(SEG$
(S$,1,1))):: CALL HCHAR(I,J+
1,ASC(SEG$(S$,2,1)))
1060 RN=RN+1 :: IF RN<76 THE
N 820
1070 M$="1" :: RETURN
1080 REM PRINT BINGO***
1090 PRINT #1:"  XXXX   XXX
 X   X  XXX    XXX":"  X   X
  X    XX  X X    X X    X":"
X   X    X   X X X X     X
X"
1100 PRINT #1:"  XXXX    X
 X X X X XXX X   X":"  X   X
  X   X   XX X   X X   X"
1110 PRINT #1:"  X   X   X
 X  XX X   X X   X":"  XXXX
  XXX  X    X  XXX    XXX"
1120 RETURN
1130 REM PRINT BINGO CARDS**
*
1140 DISPLAY AT(11,1):"NUMBE
R OF CARDS?" :: ACCEPT AT(11
,18)VALIDATE(NUMERIC):I :: G
OSUB 540
1150 DISPLAY AT(11,4):"CARDS
 BEING PRINTED...":"
PLEASE WAIT."
1160 FOR J=1 TO I :: FOR N=1
 TO 5
1170 K=INT(RND*15)+1 :: L=IN
T(RND*15)+1 :: S$=B$(K):: B$
(K)=B$(L):: B$(L)=S$ :: K=IN
T(RND*15)+1 :: L=INT(RND*15)
+1 :: S$=I$(K):: I$(K)=I$(L)
1180 I$(L)=S$ :: L=INT(RND*1
5)+1 :: K=INT(RND*15)+1 :: S
$=N$(K):: N$(K)=N$(L):: N$(L
)=S$ :: K=INT(RND*15)+1 :: L
=INT(RND*15)+1
1190 S$=G$(L):: G$(L)=G$(K):
: G$(K)=S$ :: K=INT(RND*15)+
1 :: L=INT(RND*15)+1 :: S$=O
$(L):: O$(L)=O$(K):: O$(K)=S
$ :: NEXT N
1200 GOSUB 1080 :: FOR N=1 T
O 5
1210 PRINT #1:" ";RPT$("*",3
1):" ";RPT$("*     ",5);"*":
" ";RPT$("*     ",5);"*"
1220 IF N=3 THEN S$="* FREE"
 ELSE S$="*  "&N$(N)&" "
1230 PRINT #1:" *  ";B$(N);"
 ";"*  ";I$(N);" ";S$;"*  ";
G$(N);" ";"*  ";O$(N);" *"
1240 PRINT #1:" ";RPT$("*
  ",5);"*":" ";RPT$("*      "
,5);"*"
1250 NEXT N
1260 PRINT #1:" ";RPT$("*",3
1):" ":" ":" ":" ":" "
1270 NEXT J
1280 RETURN
1290 REM PRINT CALL LIST***
1300 DISPLAY AT(11,1):"CALL
LIST BEING PRINTED."
1310 PRINT #1:"         BING
O CALL LIST":"         -----
----------"
1320 FOR I=1 TO 75 :: PRINT
#1: :"          ";SEG$("
BINGO",INT((VAL(A$(I))-1)/15
)+1,1);" ";A$(I):: NEXT I ::
 PRINT #1:" ":" ":" ":" "
1330 RETURN
1340 REM CHECK KEYS***
1350 CALL KEY(0,KEY,ST):: IF
 ST=0 THEN 1400
1360 CALL MOTION(#5,0,0,#9,0
,0,#7,0,0,#10,0,0)
1370 IF KEY<>32 THEN 1390
1380 CALL KEY(0,KEY,ST):: IF
 ST<=0 THEN 1380
1390 IF KEY=6 THEN CALL SAY(
SY$)ELSE IF KEY=13 THEN 1400
 ELSE IF KEY<>12 THEN 1380
1400 RETURN
1410 REM TRACTOR CONTROL***
1420 CALL SPRITE(#11,140,2,4
0,240,0,-6)
1430 FOR K=1 TO 5
1440 CALL SPRITE(#(11+K),116
+K*4,2,32+K*8,256,0,-6):: FO
R I=1 TO 8 :: FOR J=1 TO 6 :
: NEXT J :: CALL PATTERN(#11
,104+36*(I-INT(I/2)*2))
1450 NEXT I :: NEXT K :: FOR
 I=1 TO 20 :: FOR J=1 TO 6 :
: NEXT J :: CALL PATTERN(#11
,104+36*(I-INT(I/2)*2)):: NE
XT I
1460 CALL MOTION(#12,0,0,#13
,0,0,#14,0,0,#15,0,0,#16,0,0
)
1470 FOR I=1 TO 20 :: FOR J=
1 TO 6 :: NEXT J :: CALL PAT
TERN(#11,104+36*(I-INT(I/2)*
2)):: NEXT I
1480 CALL COLOR(#11,1)
1490 RETURN
```

continued from page 10

```
600 CALL SOUND(250,330,0)
610 CALL SOUND(250,294,0)
620 CALL SOUND(1000,262,0)
630 IF A=52 THEN 650
640 GOTO 170
650 CALL SOUND(250,196,0)
660 CALL SOUND(250,262,0)
670 CALL SOUND(750,330,0)
680 CALL SOUND(250,294,0)
690 CALL SOUND(500,262,0)
700 CALL SOUND(250,294,0)
710 CALL SOUND(250,262,0)
720 CALL SOUND(250,220,0)
730 CALL SOUND(1250,262,0)
740 CALL SOUND(250,196,0)
750 CALL SOUND(250,262,0)
760 CALL SOUND(500,330,0)
770 CALL SOUND(250,262,0)
780 CALL SOUND(250,330,0)
790 CALL SOUND(500,392,0)
800 CALL SOUND(250,349,0)
810 CALL SOUND(250,330,0)
820 CALL SOUND(1250,294,0)
830 CALL SOUND(250,392,0)
840 CALL SOUND(250,349,0)
850 CALL SOUND(500,330,0)
860 CALL SOUND(250,330,0)
870 CALL SOUND(250,294,0)
880 CALL SOUND(500,262,0)
890 CALL SOUND(250,294,0)
900 CALL SOUND(500,262,0)
910 CALL SOUND(250,294,0)
920 CALL SOUND(250,294,0)
930 CALL SOUND(250,330,0)
940 CALL SOUND(250,392,0)
950 CALL SOUND(1000,349,0)
960 CALL SOUND(250,220,0)
970 CALL SOUND(250,220,0)
980 CALL SOUND(500,196,0)
990 CALL SOUND(250,247,0)
1000 CALL SOUND(250,262,0)
1010 CALL SOUND(500,294,0)
1020 CALL SOUND(250,330,0)
1030 CALL SOUND(250,294,0)
1040 CALL SOUND(1250,262,0)
1050 IF A=52 THEN 1070
1060 GOTO 170
1070 CALL SOUND(250,147,0)
1080 CALL SOUND(500,196,0)
1090 CALL SOUND(125,247,0)
1100 CALL SOUND(125,196,0)
1110 CALL SOUND(500,247,0)
1120 CALL SOUND(250,220,0)
1130 CALL SOUND(500,196,0)
1140 CALL SOUND(500,196,0)
1150 CALL SOUND(250,156,0)
1160 CALL SOUND(500,147,0)
1170 CALL SOUND(250,196,0)
1180 CALL SOUND(500,196,0)
1190 CALL SOUND(125,247,0)
1200 CALL SOUND(125,196,0)
1210 CALL SOUND(500,247,0)
1220 CALL SOUND(250,220,0)
1230 CALL SOUND(1000,294,0)
1240 CALL SOUND(250,247,0)
1250 CALL SOUND(500,294,0)
1260 CALL SOUND(125,294,0)
1270 CALL SOUND(125,247,0)
1280 CALL SOUND(500,196,0)
1290 CALL SOUND(250,147,0)
1300 CALL SOUND(250,165,0)
1310 CALL SOUND(250,196,0)
1320 CALL SOUND(125,196,0)
1330 CALL SOUND(125,165,0)
1340 CALL SOUND(500,165,0)
1350 CALL SOUND(250,165,0)
1360 CALL SOUND(500,196,0)
1370 CALL SOUND(125,247,0)
1380 CALL SOUND(125,196,0)
1390 CALL SOUND(500,247,0)
1400 CALL SOUND(250,220,0)
1410 CALL SOUND(1000,196,0)
1420 GOTO 170
```

Questprobe featuring Spiderman

Part 2, Description of characters

by Larry Saunders

SPIDER-MAN - friend

Real name: Peter Parker.
Occupation: Freelance photographer, adventurer.
Identity: Secret.
Base of operations: New York City.
Origin: Parker was bitten by a massively irradiated spider and hugely increased strength. Using his new found powers, Parker started a short lived show business career that was soon given up in favor of crime fighting. Peter Parker now works for the Daily Bugle as a freelance photographer, mainly selling photographs of Spider-Man in action.
Height: 5'10"
Weight: 165lbs
Eyes: Brown
Hair: brown
Powers: Spider-Man possesses superhuman strength, reflexes and equilibrium, the ability to cling to almost any kid of surface, and a subconscious danger sense (Spider sense). He can lift up to 10 tons, and his reflexes are on average 15 times faster than a normal man.
Weapons: Spider-Man has developed a spider like web spinning device and a silk like compound that mimics a spider's natural abilities.

MADAME WEB - friend

Real name: Cassandra Webb.
Occupation: Professional medium.
Identity: Publicly known.
Base of operations: New York City.
Origin: Cassandra Webb has been blind since birth but discovered at an early age that she possessed clairvoyant abilities. She became a professional medium later in life but was striken by a disease of the nervous system which left her totally dependent upon a large spider web like life support system equipped with robot arms that take the place of her useless limbs.
Height: 5'6"
Weight: 110lbs
Eyes: Pale Grey
Hair: Black and silver
Powers: Madame Web possesses a number of psychic sensory powers. With great concentration she is able to scan people's thoughts or project her own thoughts into the minds of other. She also has the ability, to a limited extent, to predict future events.

ELECTRO - foe

Real name: Maxwell Dillon.
Occupation: Professional criminal.
Identity: Publicly known.
Base of operation: Mobile.
Origin: While working for an electrical power company, Dillon was struck by lightning which caused a mutagenic change in his nervous system. This resulted in him becoming a human electrical capacitor.
Height: 5'11"
Weight: 165lbs
Eyes: Blue
Hair: Red-brown
Powers: Electro has the ability to generate electrostatic energy (up to 1,000 volts per minute) and is able to store up to 100,000 volts at any time. This can be discharged at a controlled rate, with anything from one volt, to the full 100,000 being released at a time (capable of killing a man at a range of ten feet). He can release a bolt of lightning which can travel up to 100 feet at a speed of 1,100 feet per second. Electro is also capable of travelling at great speed along electrical power lines simply by creating imbalances in his magnetic field devices to a limited extent.

SANDMAN - foe

Real name: William Baker.
Occupation: Former professional criminal.
Identity: Publicly known.
Base of operations: Mobile.
Origin: Baker was on the run from the police after escaping from prison in New York. He sheltered in a nuclear testing site where he was exposed to a massive dose of radiation. This caused his body to take on the properties of animated sand.
Height: 6'1"
Eyes: Brown
Weight: 450lbs
Hair: Brown
Powers: Sandman can convert all or part of his body to sand, compact his body to make is as hard as sandstone or loosen it to make him invulnerable to physical attack.

MYSTERIO - foe

Real name: Quentin Beck.
Occupation: Former Hollywood special effects designer, now professional criminal.
Identity: Known to the police, secret to the public.
Base of operations: Mobile.
Origin: Beck was an extremely accomplished stunt man and special effects designer before he set himself the task of killing Spider-Man and taking his place, but as Mysterio, he was defeated and ended up in prison .
Height: 5'11"
Eyes: Blue
Weight: 175lbs
Hair: Black
Powers: Mysterio is a skilled fighter and athlete, a master hypnotist and illusionist. He wears a 'fishbowl' type helmet with an oxygen supply; carries portable projectors for the creation of large scale illusions, and has canisters attached to his back which emit a thick gas which not only obscures vision but also acts against Spider-Man's spider sense.

RINGMASTER - foe

Real name: Maynard Tiboldt.
Occupation: Circus ringmaster and professional criminal.
Identity: Public known.
Base of operations: Mobile.
Origin: Born into a circus family in Austria, Tiboldt became the master of his own circus after the second world war and came to America. However he soon discovered that his small circus could not make profits while in competition with huge American circus troups, so he turned to crime. He now runs an outfit called the 'Circus of Crime' which travels across America hypnotising and robbing its audiences.
Height: 6'1"
Eyes: Green
Weight: 190lbs
Hair: Grey-black
Powers: Ringmaster has some skill in acrobatics and hand to hand combat but his main skills lie in hypnosis through the 'nullatron' device, concealed in his top hat. The hat is also equipped with projectors which transmit bright lights through a spinning disc mounted in the front. These lights can daze and disorient potential victims, making them less able to resist his mind control.

DOCTOR OCTOPUS - foe

Real name: Otto Octavius.
Occupation: Ex-atomic research consultant, criminal mastermind.
Identity: Publicly known.
Base of operations: New York City.
Origin: Octavius constructed a chest harness with four tentacle like arms to enable him to manipulate radioactive substances at a safe distance. In a freak accident the harness became bonded to his skin and nervous system due to exposure to radiation.
Height: 5'9"
Eyes: Brown
Weight: 245lbs

continued on page 26

Trajectories

from SUBFILE99, USA

Whenever you want to write programs that calculate the flight of objects (rockets, tennis balls, cars jumping ramps, etc.), you need to be able to calculate the trajectory of that object under variable conditions. In this issue, we cover the most common formulas used in calculating those trajectories.

The Mathematical Information

For those of us who remember our mathematics lessons, the following formulas will act as a reminder, for the rest of us, they will probably act as an irritant. Still, we include the mathematics formulas for the sake of completeness. If you don't really understand them, that's OK, the DEF statements will do all the work anyway!

Calculating the Trajectory

When calculating the trajectory of a moving object you have three measurements: 1) Altitude (the highest point reached while in flight; 2) Range (the distance covered while in flight) and 3) Duration (the length of the flight measured in time - usually seconds). These three measurements depend on two important variables: 1) angle of trajectory; and 2) speed of travel.

The three formulas are expressed in mathematical terms as follows:

θ = Angle of trajectory
V = Speed of object (velocity)
g = acceleration due to gravity

Altitude:

$$A = \frac{V^2 \sin\theta}{2g}$$

Range:

$$R = \frac{V^2 \sin 2\theta}{g}$$

Time of flight:

$$T = \frac{2V}{g\sqrt{\sin\theta}}$$

Creating Nested DEF Statements

In our demonstration program at the end of this article we calculate the Altitude, Range and Travel Time of the object in either centimetres, feet, kilometres, or miles. This means we need three DEFs for each measuring unit. We can however, use parts of the DEF statements interchangeable . And TI BASIC lets us "chain" DEFs together by referencing a "lower" DEF in our new "higher" DEF. This is how it works:

The calculation for the distance traveled can be expressed in BASIC as follows:

```
DEF RANGE=SIN(T*3.49E-02)
```

where T=trajectory angle.

The calculation for the highest altitude reached can be expressed as follows:

```
DEF ALT=SIN(T*1.745E-02)
```

These two "primary" DEFs can then be used in the "secondary" DEFs as follows:
To calculate Range, Altitude and Duration in centimetres per second:

```
DEF CMSRANGE=RANGE*V*V*1.094E-03
DEF CMSALT=ALT*V*V*5.097E-04
DEF CMSTIME=SQR(ALT)*V*2.039E-03
```

Note in inclusion of the DEFs RANGE and ALT in the above DEFs. This is a kind of "nested" DEF structure and can be very useful when constructing complex DEFs.

Putting it all Together

The complete set of DEFs for figuring Range, Altitude and Flight Time for centimetres, feet, kilometres and miles can be found in the first part of the demonstration program below. Experiment with the program. Try out various angles of trajectory and velocity to see how the formulas work. Then, if you are really interested, try writing a graphic demonstration program that incorporates these calculations.

```
100 REM *****************
110 REM *               *
120 REM * TRAJECTORIES  *
130 REM *               *
140 REM * DEMONSTRATION *
150 REM *               *
160 REM *****************
170 REM
180 REM      4/85
190 REM     SUBFILE99
200 REM
210 REM **************
220 REM *HOUSEKEEPING*
230 REM **************
240 REM
250 DEF RANGE=SIN(T*3.49E-02)
260 DEF ALT=SIN(T*1.745E-02)
270 REM
280 DEF CMSRANGE=RANGE*V*V*1.0194E-03
290 DEF CMSALT=ALT*V*V*5.097E-04
300 DEF CMSTIME=SQR(ALT)*V*2.039E-03
310 REM
320 DEF FPSRANGE=RANGE*V*V*3.0864E-02
330 DEF FPSALT=ALT
340 DEF FPSTIME=SQR(ALT)*V*6.173E-02
350 REM
360 DEF KPHRANGE=RANGE*V*V*7.866E-06
370 DEF KPHALT=ALT*V*V*3.933E-06
380 DEF KPHTIME=SQR(ALT)*V*5.663E-02
390 REM
400 DEF MPHRANGE=RANGE*V*V*1.2574E-05
410 DEF MPHALT=ALT*V*V*6.287E-06
420 DEF MPHTIME=SQR(ALT)*V*9.053E-02
430 L$="----------------------------"
440 REM
450 REM **************
460 REM *TITLE SCREEN*
470 REM **************
480 REM
490 CALL CLEAR
500 FOR L=1 TO 17
510 READ M$
520 PRINT TAB(4);M$
530 NEXT L
540 INPUT "  PRESS ENTER TO START":A$
550 REM
560 REM ************
570 REM *MAIN INPUT*
580 REM ************
590 REM
600 CALL CLEAR
610 PRINT "ENTER DATA":"----------": : :
620 INPUT "ANGLE OF TRAJECTORY:":T
630 PRINT
640 INPUT "VELOCITY OF OBJECT: ":V
650 PRINT : :
660 REM
670 PRINT "MEASURE IN:":
680 PRINT :"1 - CENTIMETRES PER SECOND"
690 PRINT :"2 - FEET PER SECOND"
700 PRINT :"3 - KILOMETRES PER HOUR"
710 PRINT :"4 - MILES PER HOUR"
720 PRINT : : :
730 INPUT "ENTER CHOICE:":C
740 IF (C<1)+(C>5)THEN 730
750 REM
760 ON C GOSUB 1200,1280,1360,1440
770 REM
780 REM ***************
790 REM *PRINT RESULTS*
800 REM ***************
```

continued on page 4

c99 Routines

by Stephen Shaw, UK

Some short c routines to get you used to using c99 and maybe show how some things are done and how some things are used. These routines are by Donald L Mahler and come from the Boston Computer Society. They have been printed from tested source code.
Remember:
*s means "pointer to s" while
&s means "the address of s"

File prf is as follows:

```
/* file DSK1.PRF */
/* PRINTF REFS */
#asm
   REF  PRINTF
#endasm
```

Save it to disk, calling it PRF.

```
/* 1;C            */
#include dsk1.prf
int table[]={3,5,2,9,6};
/* sets up an array */
main()
{
  int i; i=0;
/* first term of array is "0th" */
  while (i<5)
  {
    printf("The address of the %dth \n",i);
    printf("element of table is %u.\n",&table[i]);
/*              */
/* "&table[i]" = "address of ith term of array" */
/*              */
/* addresses are unsigned integers that is why we */
/* use 'u'      */
/*              */
    printf(".. and the value stored there\n");
    printf("is %d\n",*(&table[i]));
/*              */
    ++i;
/*              */
/*  increment i  */
/*              */
    putchar('\n');
  }
}
```

Type this in using the Assembler Editor, save it and then compile it, assemble the result, to say 1;O. Do not select any assembly options!
To run using LOAD AND RUN load:
DSK1.1;O
DSK1.PRINTF
DSK1.CSUP
then start with program name START.
CSUP and PRINTF are supplied with the c99 package.

This second routine uses strings and also requires the file prf defined above!

```
/* 2;C         */
#include dsk1.prf
main()
{
  char *ptr1, *ptr2 ;
/* two character pointers */
  ptr1="Boston/Computer/Society";
/*             */
/* the address of a string is the address of first */
/* letter */
/*             */
  ptr2=ptr1 ;
  while (*ptr2)
/*              */
/*  "*ptr2 !=0" */
/*              */
  {
    putchar(*ptr2++) ;
  }
/*              */
/* spell out the string letter by letter    */
/*              */
  puts("\n \n Now let's reverse it! \n\n");
/*              */
/* ptr2 is now address of last letter of string!!! */
/*              */
  while (--ptr2 >= ptr1)
/*              */
/* decrease address until back at original starting */
/* address */
/*              */
  {
    putchar (*ptr2);
  }
  putchar('\n');
}
```

And here is another short example of c99 in action. Try it out now!

```
/* 3;C            */
#include dsk1.prf
main()
{
  char x;
  puts("Enter any letter : \n\n");
  x=getchar();
  putchar('\n');
  printf("The upper case form of %c is",x);
  caps(&x);
  putchar(x);
  putchar('\n');
}
caps(ptr)
char *ptr;
{
  if (*ptr <= 'z' & *ptr >= 'a')
  *ptr = *ptr + 'A' - 'a' ;
}
/*              */
/* if letter is lower case then decrease ASCII */
/* value by the difference between 'A' (65) */
/* and 'a' (97)       */
```

Now compile, assemble (remember, no options!) and load and run:
DSK1.3;O
DSK1.PRINTF
DSK1.CSUP
and program start name is START.
If however you wish to transform your program to memory image format, to use with RUN PROGRAM FILE, then load these files, using LOAD AND RUN:
DSK1.C99PFI
DSK1.3;O
DSK1.PRINTF
DSK1.CSUP
DSK1.C99PFF
DSK1.FWSAVE
and now choose the program named SAVE.
Now you will have a single "PROGRAM" file which you can load in one piece, instead of having to load lots of other files. ©

continued from page 12

The "N" file is the executing commands for the AUTOEXEC MENU file. Therefore, any time at all, while I am in M-DOS, all I need to press to get my most used commands is an N, a space, and a letter (A to G). You can customize either file to suit your own special needs. This menu system will be even better if you have a RAMdisk with a battery so you can keep your most used files ready to load. Just substitute whatever disk drive numbers that your using for my disk numbers. This setup saves a lot of time when in M-DOS. When you create your own AUTOEXEC and "N" files use the: COPY CON "filename" command in M-DOS. This creates a D/V 80 file without any control characters in it like when you create a file with TI-WRITER and print it to disk using C DSK1."filename" as a device name in the print file command.
Well, that is all I have to pass along this month. Bye now. (Courtesy of LeHigh 99'er April 1988 page 3.) ©

Turbo Pasc 99

by Stephen Shaw, UK

At press time, I was still awaiting a positive price quotation from Texaments, prior to ordering. However, I have managed to obtain a German copy of the program and have done some benchmark tests as below.

A full review will appear as soon as I have the English version, with English documentation! I understand that Texaments have the source code and are making one or two amendments. These should not alter the benchmarks or general comments too much.

Error handling is excellent, with many syntax errors caught on compilation. Use of defaults when you ask for the impossible mops up many run-time errors, and the range of error messages for run-time is greater than that produced by the Editor/Assembler alone. An error caught on compilation will place you back in the editor with the cursor near the error. Most helpful!

This is not Borland's Turbo Pascal(tm). It is close to "standard" ISO Pascal, amended to make it more friendly for programmers used to the TI99/4A.

ISO Pascal items not supported are:
  file, in, packed, record, set, type, with, char, ord, pred, round, sqr, succ, trunc, odd, reset, rewrite, dispose, new, pack, unpack.
Items included but not in ISO Pascal (some replacing items above):
  Block, Module, Relative, Stream, String, Open, Seek, Append, Close, Asc, Cursor, Key, Screen, minint, pi, graphics, text, putln, cls, cir, cis, cri, crs, csi, csr, len, int, rnd, seg, tan, randomize.
Variations on ISO Pascal are:
  Strings are within double quotes "string" instead of 'string'. REAL must be specified as 4, 6 or 8 bytes; if you select 8 you have normal Extended BASIC precision.

Some of the portability has gone with these changes of course, but they seem to make the language easier to enter for a TI99/4A user with no Pascal experience!

In addition, disks containing additional procedures are available, one for Windows, and one for our familiar TI99/4A extensions with sprites, sound and so on. If the language is popular, expect more in due course.

NB: The German version I have requires the Editor/Assembler Module to function.

Books worth looking at:
A CRASH COURSE IN PASCAL (about 8 pounds) by D M Munro, Arnold; ISBN: 0-7131-3553-0
PASCAL USER MANUAL AND REPORT by Jensen and Wirth, Springer-Verlag; ISBN: 0-387-96048-1 (specify third edition)

There are also books by Peter Lottrup (COMPUTE!) and Zaks (Sybex) which may be worth a look. Take care: most Pascal books these days seem to be for Turbo Pascal (not Turbo Pasc 99 or ISO Pascal).

Turbo Pasc 99 benchmarks.

These benchmarks were detailed in TI*MES issue 15, and have appeared several times in PERSONAL COMPUTER WORLD magazine.

For Turbo Pasc 99, the Pascal code will be given first, followed by the timing and possibly some notes. (Remember: you do not need a P-Code card to run Turbo Pasc 99!)

```
PROGRAM intmath;
  VAR t,i,x,y : INTEGER;
BEGIN
  writeln("........");
  t := 0;
  FOR t := 1 TO 100 DO
   BEGIN
     x := 0;
     y := 9;
     writeln("start");
     FOR i := 1 TO 1000 DO
      BEGIN
       x := x + (y * y - y) DIV y
      END;
    END;
  writeln("---",x);
END.
```

The timing on this program equates to a benchmark of 0.337 seconds for 1000 loops, which compares well with a benchmark of 0.48 seconds for c99.

```
     PROGRAM realmath;
     VAR t,
         i : INTEGER;
         x,y : REAL[4];
     BEGIN
       writeln(".........");
       FOR t := 1 TO 5 DO
        BEGIN
         x := 0.0;
         y := 9.9;
         FOR i := 1 TO 1000 DO
          BEGIN
           x := x + (y * y - y) / y;
          END;
        END;
       writeln("***END...",x);
     END.
```

Real numbers using 4 bytes, took 8.20 seconds for 1000 loops, compared to 17.7 seconds for plain ordinary TI BASIC!

```
PROGRAM triglog;
VAR i : INTEGER;
    x,y : REAL[4];
BEGIN
  writeln("........");
  x := 0.0;
  y := 9.9;
  FOR i := 1 TO 400 DO
   BEGIN
    x := x + sin(arctan(cos(ln(y))));
   END;
  writeln("***",x);
END.
```

This was slow and the equivalent of 1000 loops took 625 seconds, which compares badly to 360 seconds in Extended BASIC!

```
PROGRAM textscrn;
VAR i : INTEGER;
BEGIN
  text;
  writeln("START");
   FOR i := 1 TO 1000 DO
    writeln("1234567890qwertyuiop",i);
  writeln("***",i);
END.
```

This one took 69 seconds for 1000 loops, again comparing badly with c99, which took just 38.7 seconds.

```
PROGRAM store;
VAR i : INTEGER;
    f : STREAM[80];
BEGIN
  writeln("START");
  open(f,"DSK2.TEXT",output);
  FOR i := 1 TO 1000 DO
   putln(f,"1234567890qwertyuiop");
  close(f);
  writeln("******");
END.
```

This took 61.4 seconds, compared to 83 seconds in Myarc Extended BASIC and 131 seconds in TI Extended BASIC. Notice how easy disk access can be!

RESULTS:
Benchmarks are not the be all and end all of a language, although an advertiser can give the impression a language is incredibly fast by telling you what it is fast at, and not telling you what it is slow at! Turbo Pasc 99 seems to be comparable with c99 overall, sometimes better sometimes not.

Personally I found it much easier to use Turbo Pasc 99; I even made it write to disk! Look how short the Store program is!
If you are a TI99/4A P-Code user you may find these codes a little odd. I can assure you they work exactly as printed here. If you have a P-Code card, why not run comparative tests and let me know the times?

The answers printed out by the math results were: intmath 8000, realmath 8.9E3 (both the same as TI BASIC) and triglog -2.2021E2, compared to -2.20497E2 from TI BASIC. This minor inaccuracy is due to using only 4 bytes for the variable rather than 8, but I could have used 8 if I had wanted such accuracy. o

Turbo Pasc 99 Program

from Stephen Shaw, UK

Very very few of our readers will own Turbo Pasc 99 right now, but for those of you with disk drives, here is a lengthy sample of a program written in Turbo Pasc 99, to enable you to see how it appears, and for those of you with TI Pascal, you can see how the syntax varies. Professional Pascal users may be interested to see how arrays are passed to procedures!

This is the first ever Pascal program written by me, and is based on a BASIC program in the book "Dynamic Games for your TI99/4A" by Scott Vincent. I have wanted a flexible Life simulation in machine code, but not found any suitable, so I have taken advantage of Pascal to see what I can do!

In this version, you first set up the Life starting position by keying A for life and <SPACE> for no life, to fill a 12x12 grid. I have not put any cursor control in, it just moves over each line in turn, left to right.
The outer frame has fixed status, it will affect the inner 10x10 grid, but no changes will occur in the frame. If you put life there it will stay there! This enables you to experiment with variations on the normal isolated life universe. To help spot the frame, the cursor is a + in the frame and a ? in the 10x10 grid.

You have the option of allowing automatic changes in generations, or keying them in. If you select automatic, you may return to the set up section just by pressing a key. If you select keyed progress, to get back to the set up screen involves quickly pressing two different keys, one after the other.

There is one "bug" left in. The counter G is of integer type but there is no test for the maximum integer, so after some 32000 odd generations the program will bomb out!

LIFE patterns may:
- Die out.
- Reach a stable pattern.
- Cycle between patterns: changing between two patterns is the most common, but I have created a 4-pattern cycle.
- As a variant of the above, a pattern has been found in which a "bullet" is regularly produced from a cluster, and moves off screen.

Here is the entire Pascal code and as successfully compiled.

```
{ Pascal CODE ==================================}

PROGRAM life;

 VAR k,z:INTEGER;
    m,n:ARRAY[12,12] OF INTEGER;

 PROCEDURE rules;
  VAR k,status:INTEGER;
  BEGIN
   cls;
   writeln(" LIFE");
   writeln(" first set up start position");
   writeln(" Put life forms into a");
   writeln(" 10 x 10 grid ");
   writeln(" by pressing A or space");
   writeln(" (The 10x10 grid is in a frame-");
   writeln(" making a 12x12 grid in total:");
   writeln(" If you place life in the frame");
   writeln(" it will never die, and can ");
   writeln(" constantly create new life if");
   writeln(" the proper life conditions are met)");
   writeln(" ");
   writeln(" There is no cursor control.");
   writeln(" Generation starts after 144th");
   writeln(" key press");
   writeln(" When prompted indicate if you");
   writeln(" want to prompt each generation");
   writeln(" with a Keypress, or let the");
   writeln(" program go full tilt.");
   writeln(" ");
   writeln(" Press any key to continue");
   writeln(" Stephen Shaw Jan 1988");
   REPEAT
    key(3,k,status)
   UNTIL status=1;
  END;

 PROCEDURE draw(VAR m,n:ARRAY[12,12] OF INTEGER);
  VAR row,col:INTEGER;
  BEGIN
   FOR row:=1 TO 12 DO
    BEGIN
     FOR col:=1 TO 12 DO
      BEGIN
       m[row,col]:=n[row,col];
       cursor(row+2,col+5);
       IF m[row,col]=1 THEN
        write("O")
       ELSE
        write("~");
     END;
    END;
  END;

 PROCEDURE think(VAR m,n:ARRAY[12,12] OF INTEGER);
  LABEL rerun;
  VAR c,g,k,s,t,y,z:INTEGER;
  BEGIN
   cls;
   cursor(2,2);
   writeln(" Press K to key generations");
   writeln(" Or any other key for auto");
   writeln(" ");
   writeln(" When all life ceases,");
   write(" Press a key to restart");
   REPEAT
    key(3,y,c)
   UNTIL c=1;

 { y = 75 to key prompt }

   cls;
   g:=0;
   REPEAT
    g:=g+1;
    FOR k:=2 TO 11 DO
     BEGIN
      FOR z:=2 TO 11 DO
       BEGIN
        c:=m[k-1,z-1]+m[k-1,z]+m[k-1,z+1];
        c:=c+m[k,z-1]+m[k,z+1];
        c:=c+m[k+1,z-1]+m[k+1,z]+m[k+1,z+1];
        IF (m[k,z]=0) AND (c=3) THEN
         n[k,z]:=1;
```

continued on page 28

The Evolution of a Failure

The Story of the IBM PC,

by Chris Bobbitt, Asgard Software, USA

The story of IBM's efforts in the microcomputer industry is a story of a company's inability to adapt to a marketplace. In short, IBM has ultimately been a failure in imposing its will on the computer industry. This story is a parable of the dangers of corporate gigantism, and an illustration, yet again, of how a properly motivated David can knock of a Goliath any time and anywhere.

In 1982 IBM placed its seal of approval on the microcomputer industry with the introduction of the IBM PC. Microcomputers had been in constant use, particularly by small business and by professionals and farmers since the introduction of CP/M in 1977 by DRI. The early microcomputers, by the standards of today (which have in no part been set by IBM, but I am getting ahead of myself now), were clunky and slow. They used little graphics or sound (these were reserved for "home" computers like the Apple II, Atari 400, Commodore 64, and TI99/4A), but were very functional. Visicalc, the forerunner of all spread sheets (now considered an indispensable tool by business), was invented and propagated on such machines. Word processors, data bases and telecommunications flourished years before IBM ever sold a single computer for less than $25,000. Until 1979, basically, the computer industry was IBM and everyone else. By 1982 it was IBM, everyone else, and Apple, Atari, Texas Instruments, Radio Shack, etc.

It was at this point that IBM decided to make their own entry into the microcomputer world. Like a dinosaur sensing its own extinction unless moved to action, IBM set up an "entrepreneurial corporate unit", basically a company within a company, to design and build a computer aimed at small business and individuals. This machine, by corporate dictum, was to "break no new ground". The designers succeeded with flying colours, later to the ultimate dismay of the very corporation that sponsored it. They basically designed a machine that was obsolete 3 years before it was introduced. Most aspects of the machine were basically a retrogression to an earlier era. IBM used strictly off the shelf components that they could purchase cheaply (the better the profit margin, my friend), and an obsolete design based on a horrible misfit of a chip manufactured by Intel, a then half bankrupt microprocessor house. The only saving grace of the chip was that it had "16 bits". This made the IBM PC, as it was dubbed, the second widely available 16 bit machine, the other was based on a strange chip called the TMS9900, by TI, which was so off the wall and unique in its design approach that normally conservative computer system designers could not make head or tail of it, even though it was the most innovative chip ever released at that time. It was a chip that only a hard boiled techie could love, and has since been relegated to same. The TMS9900 later become a commercial failure, due to TI's inability to sell to anyone but the government, but that is a different story.

When it came time to find an operating system for this machine, IBM went to a small company called Microsoft, whose major claim to fame at that time was its Microsoft BASIC, the standard for BASIC languages now and then. Microsoft was told to get an operating system fast, but not to tell anyone who it was for. Bill Gates, president of Microsoft, approached RDI (then, second largest software company in the world after Visicorp, maker of Visicalc), and asked them to do a custom operating system for an unspecified manufacturer within 3 months. Dr. Kildall of DRI basically told him to take a number, since he was busy porting CP/M to 20 or so machines at the time. Gates' response was to dust of a CP/M clone that had been lying on a back shelf since 1978 (purchased for $4000 from 2 brothers in Oregon), rename it MS-DOS, and offered it to IBM. IBM's response was "we don't care what it is as long as it works". Thus was born MS-DOS, and the source of Microsoft's fortune since.

The IBM PC was released in the early part of 1982 to generally bad reviews. Most of the computer magazines of the time were run by people who had a good knowledge of computers, and they recognized the PC for what it was; a real piece of garbage (or at least nothing compared to your average S-100 system). Despite the efforts of a number of anti-establishment heroes of the early computer revolution, this was to become the standard, at least until recently.

While the computer press wrote the machine off, the big business press went wild over it. Companies who had never dreamed of purchasing a computer for less than $300,000 were now given the green light to buy a machine for, gosh, individual use, for about $3,000. A popular phrase at the time, "no one was ever fired for buying IBM", took on new significance as suddenly the Fortune 500 set woke up and realised that microcomputers could do things, useful productive things. A small company called "Lotus Development Corporation", sensed an opportunity and introduced a Visicalc knock-off called "Lotus 1-2-3" for the PC, and the machine took off. Soon every big accounting company in New York had hundreds of the things, and were wildly recommending their usefulness to their big clients. Within 2 years, all the early critics of the machine were either out of business, or sold to the new faith. A standard was born!

The seeds of the PC rise were later its doom. While the PC itself was a machine of marginal, even poor capability, it had what is known as an "open architecture". Since all the components for the machine were easy to find as well as technical information for it, by 1984 there were hundreds of devices on the market to correct the many short comings of the machine. With an infusion of cash, the failing Intel took-off and produced the 8086 (thus was born the IBM PC XT) and then the 80286 (the IBM PC AT). While the other chips corrected the many problems with the 8086 (which was reportedly so ill designed that it had bugs in the built in controlling software that sometimes made a program never run the same each time), they basically expanded upon a design ideology that was reaching its physical limit of expansion.

Another difficulty was rising in the east; the far east actually. Because the components of the IBM PC were basically low technology items in the scale of things, the clone was born. This is a machine that is basically a copy of the IBM PC. Because many of the makers of the clones included things as standard that were optional, fixed for the problems with the PC series (such as faster memory and processors) it soon became apparent that the IBM PC family was technically inferior to its imitators (not too difficult a thing to be). Because components were standardized so much and the construction process downright simple, the only real basis of comparison between machines became price, not quality, features, or technology. As a result, within a two year period most computer component production in the United States fled overseas where it was cheaper to build the machines. The IBM standard had a stranglehold on 70% of the market, and the only way to compete with this 70% was if you were cheaper than the other guy. Hence was born the "competitiveness" issue so popular recently in political circles. IBM basically stabbed the American high tech industry in the back. Production shifted almost completely overseas in response to a more price sensitive situation (early computers were almost always completely constructed in the US. As a matter of fact the last microcomputer to be wholely made over here was the TI99/4A in Lubbock). Between the years of 1982 and 1985, more than 50,000 jobs in the high-tech industry fled overseas because of the "IBM standard".

So, a number of simultaneous events were happening. The IBM standard was approaching the limits of its inherent expandability, production was rapidly shifting off-shore as the basis of competition became price, and the number of clone manufacturers jumped from 10 to 500. At the same time, IBM's market share dropped from 50% of the microcomputer market to 25%, and their profit margin fell in half due to competition. If this was not bad enough, the Fortune 500 crowd soon became familiar enough with microcomputers so that they were now not afraid to buy clones that were technically superior; so much so that the catch phrase today is "you may be fired for buying IBM". Even the US government was purchasing Zeniths, Compaq, and WP clones! The last nail in the coffin was struck by a small competitor of IBM, Apple.

In 1982 Apple had 40% of the microcomputer market. By 1984 it had dropped to 12%. However, through this period Apple's profitability trebled, and it introduced a machine that was so obviously technologically superior that it has become a legend, the Macintosh. While never having the success of the PC in big business, the Macintosh founded an industry. Electronic publishing (the art of using a computer to combine words and text on a page in typeset quality), and a niche in art departments, newspapers, and in small businesses all over the country. While software development for the Macintosh never began to approach that of the IBM world, what has been produced is so obviously of high quality that John Dvorak of the industry newspaper, Infoworld, which is dominated by IBM standard, was so moved to declare that "all innovative software being produced today is on the Macintosh". The Macintosh has a profitable niche market that is expanding to other areas as users in big business have begun to realise its utility elsewhere, showed the possibilities of graphics, sound, and ease of use. This was the straw that broke the camels back so to speak. The basic IBM PC lacks all those things.

By 1986 it was obvious that the IBM standard was going to fade into oblivion, and IBM with it if IBM's profits were any indicator. As a matter of fact, in late 1985 IBM posted its first loss ever, including during the great depression. All due to the failure of its microcomputer standard. "SOMETHING HAD TO BE DONE!" was heard throughout the hallowed halls of the corporate bureaucracy. IBM had stepped on its cape. It had discovered that even the largest computer company in the world cannot ignore a basically technology driven market forever. The reaction was the birth of the rumors of a "clone killer".

These rumors materialized a few weeks ago with the introduction of the PS/2 series of computers. Within the next 18 months, it will become laughably obvious how much of a failure IBM is. If the PC was IBM stumbling, the PS/2 is the outright fall. The PS/2 is IBM's attempt to imitate the Macintosh.

The net affect of the PS/2 series is to raise the costs of producing a clone of it dramatically. Since it uses proprietary technology called "gate arrays" (which cost anywhere between $20-100,000 each to produce), it effectively raises the cost of producing a clone out of reach of 99% of clone manufacturers. However, the PS/2 is never going to worry much about clones. In fact, it is a failure on arrival.

IBM will sell thousands of them to large corporations, but for the foreseeable future no company is ever again going to dominate the microcomputer market so. As the PC standard gets older and its limitations get more obvious, the number of companies producing computers will rise, while clone makers will drop like flies. By the fact that IBM has used so much proprietary technology, the new companies by necessity will produce non-IBM machines. For the first time since 1981, computer companies will be owned by engineers again, and will look to the future, and not the past. The types of computers available will increase dramatically. There will again be several different popular chips available on the market supporting hundreds of operating systems.

One thing is for sure, though, many of the machines will run PC software, even though they do not resemble a PC in the least on the inside or out. All will run clones of PC software. The number of niche markets will rise dramatically as again computers will be marketed specifically to particular market segments. The total effect of all this is the verticalization of the marketplace. The world, as we know it, will never be the same.

Oh yes, the PC has a future, but the same future afforded other abandoned computers - ignominity and the eventual death of major support. If you buy a PC now, you are either crazy, or you really do not care about support. If the latter is the case, then you can be proud to be known as an "orphan". The joke is definitely on you if you bought a PC to escape that possibility!!!!!!

o

continued from page 9

The above sample is a real one. I use it quite often to jump into SuperBug II in my Super Cartridge for debugging assembly programs. The data in addresses >FFFC and >FFFE can be set to anything you wish. They are often set to point to a screen dump program in memory, but can be used any way you like. As an experiment, set >FFFC to >8300 and >FFFE to >A000 with a debugger. Now load Fast-Term or any other program that starts at address >A000. Press QUIT to return to the color bar screen. Now press the load interrupt button. You should immediately see the Fast-Term initial display. Whatever address is in the address >FFFE when the button is pressed will be executed immediately. The load interrupt is a non maskable interrupt. That is, it can not be disabled even by a LIMI 0 instruction. Only RESET has a higher priority than -LOAD. If RESET and -LOAD occur simultaneously, reset will happen first, then the load interrupt will be processed.

With the simple load interrupt switches most of us have installed it is not possible to return to the interrupted program without using special software techniques. The switch bounce causes multiple interrupts which corrupt R13, R14 and R15. Remember, the return information was stored in those 3 registers by the BLWP instruction. In many cases you will not want to return anyway, so this would not be a problem. If you have a load interrupt switch that produces only one interrupt, you can return to the original program by simply executing a RTWP (return with pointer) instruction, assuming you have not reloaded the workspace pointer register. RTWP takes the value in R13 and places it in WP register, R14 and places it in the PC register, and R15 and places it in the STATUS register, This has the effect of restoring the environment present when the load interrupt occurred. The interrupted program will then resume execution where it left off.

The load interrupt switch circuit to produce only one interrupt is quite simple. It consists of 2 chips, 3 resistors, 1 diode and a push button switch. If there is enough interest in this circuit, we could publish it in a future newsletter. By the way, do not waste your time building the load interrupt circuit described in the March issue of MICROpendium (page 12). It simply gives precisely timed switch bounces. It does not give you a one shot load interrupt! Meanwhile, if you come up with any interesting load interrupt software, I would very much appreciate receiving a copy of it. Very little has been done in this area. You could become famous. o

Tips from the Tigercub #52

by Jim Peterson, Tigercub Software, USA

Over 120 original programs in BASIC and Extended BASIC, available on cassette or disk, now reduced to just $1 each!, plus $1.50 per order for cassette or disk and postage and packing. Minimum order of $10. Cassette programs will not be available after my present stock of blanks is exhausted. The Handy Dandy series, and Color Programming Tutor, are no longer available on cassette. Descriptive catalogs, while they last, $1 which is deductable from your first order.

Tigercub Full Disk Collections, reduced to $5 postpaid. Each of these contains either 5 or 6 of my regular catalog programs, and the remaining disk space has been filled with some of the best public domain programs of the same category. I am not selling public domain programs; they are a free bonus!

TIGERCUB'S BEST, PROGRAMMING TUTOR, PROGRAMMER'S UTILITIES, BRAIN GAMES, BRAIN TEASERS, BRAIN BUSTERS!, MANEUVERING GAMES, ACTION GAMES, REFLEX AND CONCENTRATION, TWO-PLAYER GAMES, KID GAMES, MORE GAMES, WORD GAMES, ELEMENTARY MATH, MIDDLE/HIGH SCHOOL MATH, VOCABULARY AND READING, MUSICAL EDUCATION, KALEIDOSCOPES AND DISPLAYS

Nuts & Bolts disks

These are full disks of 100 or more utility subprograms in MERGE format, which you can merge into your own programs and use, almost like having another hundred CALLs available in Extended BASIC. Each is accompanied by printed documentation giving an example of the use of each. Nuts & Bolts (No. 1) has 100 subprograms, a tutorial on using them, and 5 pages of documentation. Nuts & Bolts No. 2 has 108 subprograms, 10 pages of documentation. Nuts & Bolts #3 has 140 subprograms and 11 pages of documentation. Now just $15 each, postpaid.

Tips from the TIGERCUB

These are full disks which contain the programs and routines from the Tips from the Tigercub newsletters, in ready-to-run program format, plus text files of tips and instructions. Tips (Vol. 1) contains 50 original programs and files from Tips newsletters No. 1 through No. 14. Tips Vol. 2 contains over 60 programs and files from Nos. 15 through 24. Tips Vol. 3 has another 62 from Nos. 25 through 32. Tips Vol. 4 has 48 more from issues No. 33 through 41. Now just $10 each, postpaid.

```
**********************************
*           NOW READY            *
*  Tips from TIGERCUB Vol. 5     *
* Another 49 programs and files  *
* from issues No. 42 through 50. *
*     Also $10 postage paid      *
**********************************
```

TIGERCUB CARE DISKS #1, #2, #3 and #4. Full disks of text files (printer required). No. 1 contains the Tips news letters #42 thru #45, etc. Nos. 2 and 3 have articles mostly on Extended BASIC programming. No. 4 contains Tips newsletters Nos. 46 to 52. These were prepared for user group newsletter editors but are available to anyone else for $5 each postpaid.

This one should come in handy for bowling league captains and Little League coaches.

```
100 DIM M(29,29),T$(30)
110 GOTO 130
120 N;Q$;J;I;X;P$;S$;K
130 !@P-
140 DISPLAY AT(3,7)ERASE ALL:"LEAGUE SCHEDULER":;:"by
    the Burwells         adapted by Tigercub"
150 DISPLAY AT(8,1):" This program sets up a":"schedule
    for up to 30 teams":"so that each plays
    each":"other once and only once."
160 DISPLAY AT(12,1):" If an odd number of teams":"are
    scheduled, each gets one":"bye."
170 DISPLAY AT(16,1):"Number of teams?" ::
    ACCEPT AT(16,18)VALIDATE(DIGIT):N :: IF N>30 THEN
    DISPLAY AT(18,1):"LIMIT OF 30!" :: GOTO 170
180 DISPLAY AT(18,1)ERASE ALL:"Schedule teams by name?
    Y" :: ACCEPT AT(18,25) SIZE(-1)VALIDATE("YN"):Q$ ::
    IF Q$="N" THEN 200
190 FOR J=1 TO N :: DISPLAY AT(20,1):"Team
    no.";J;"name?" :: ACCEPT AT(22,1):T$(J):: NEXT J ::
    GOTO 210
200 FOR J=1 TO N :: T$(J)="Team No. "&STR$(J):: NEXT J
210 IF N/2<>INT(N/2)THEN N=N+1 :: T$(N)="bye"
220 DISPLAY AT(23,1):"Schedule by day, week, month":"or
    what?" :: ACCEPT AT(24,10):S$ :: FOR J=1 TO N-1 ::
    M(1,J)=J+1
230 NEXT J :: FOR J=1 TO N-1 STEP 2 :: GOSUB 260
240 NEXT J :: FOR J=2 TO N-2 STEP 2 :: GOSUB 330
250 NEXT J :: GOSUB 390 :: STOP
260 FOR I=1 TO N-2 :: IF M(I,J)=N THEN 280
270 M(I+1,J)=M(I,J)+1 :: GOTO 290
280 M(I+1,J)=M(I,J):: GOTO 300
290 NEXT I
300 X=I+1 :: FOR I=X TO N-2 :: M(I+1,J)=M(I,J)-1
310 NEXT I
320 RETURN
330 FOR I=1 TO N-2 :: IF M(I,J)=2 THEN 350
340 M(I+1,J)=M(I,J)-1 :: GOTO 360
350 M(I+1,J)=M(I,J):: GOTO 370
360 NEXT I
370 X=I+1 :: FOR I=X TO N-2 :: M(I+1,J)=M(I,J)+1
380 NEXT I :: RETURN
390 DISPLAY AT(12,1)ERASE AL L:"Output to - 2":;:" (1)
    Sc reen":" (2) Printer" :: ACCEPT AT(12,13)SIZE(-1)
    VALIDATE("12"): K :: IF K=1 THEN 440
400 DISPLAY AT(18,1):"Printer? PIO" ::
    ACCEPT AT(18,10)SIZE(-18):P$ :: OPEN #1:P$ :: PRINT
    #1:"LEAGUE SCHEDULE": : :: FOR I=1 TO N-1 :: PRINT
    #1:S$;" #";I :: PRINT #1:T$(1);" vs ";T$(M(I,1))
410 FOR J=2 TO N-2 STEP 2 :: PRINT #1:T$(M(I,J));" vs
    "; T$(M(I,J+1))
420 NEXT J :: PRINT #1:"": :
430 NEXT I :: RETURN
440 FOR I=1 TO N-1 :: PRINT TAB(7);"LEAGUE SCHEDULE": :
    :: PRINT "WEEK #";I: : :: PRINT T$(1);" vs
    ";T$(M(I,1)):: FOR J=2 TO N-2 STEP 2 :: PRINT
    T$(M(I,J));" vs ";T$(M(I,J+1))
450 NEXT J :: PRINT "": : :: PRINT "PRESS ANY KEY FOR
    NEXT WEEK"
460 CALL KEY(0,K,S):: IF S=0 THEN 460
470 CALL CLEAR
480 NEXT I :: RETURN :: END
```

Some folks seem to think that the subprograms on my Nuts & Bolts disks are just flashy screen displays. Not so! This one will be on the next disk full, if I ever get it full, which is most unlikely.

ACCEPT AT with a negative size is useful to accept a default string from the screen, but the length of the string is limited to 28 characters; and if you want something other than the default, you must be sure to delete any extra characters. CALL DEFAULT(R,C,M$,R$), where R and C are the row and column to accept at, M$ is the default string which can be up to 254 characters long, and R$ is the string accepted, will display the default string, accept it if Enter is pressed, or accept any other string without having to blank out the extra characters. Just do not type too fast!

```
100 M$="TESTING" :: CALL CLEAR
110 CALL DEFAULT(12,1,M$,R$) :: DISPLAY AT(24,1):R$ ::
    GOTO 110
10000 SUB DEFAULT(R,C,M$,R$) :: R$="" :: X=ASC(M$)
10001 DISPLAY AT(R,C):M$
10002 CALL HCHAR(R,C+2,ASC(SEG$(M$,1,1))) :: CALL
    HCHAR(R,C+2,30)
10003 CALL KEY(0,K,S):: IF S=0 THEN 10002 ELSE IF K=13
    THEN R$=M$ :: SUBEXIT ELSE DISPLAY
    AT(R,C):CHR$(K):: ACCEPT AT(R,C+1):R$ ::
    R$=CHR$(K)&R$
10004 SUBEND
```

CALL DEFAULT(R,C,N,RN), with N as the default value and RN as the value accepted, will do the same for numeric input, and will reject any non-numeric input. Errors due to fast typing can be prevented by omitting the DISPLAY AT(R,C):CHR$(K) in line 1002.

```
100 N=176453.897 :: CALL CLEAR
110 CALL DEFAULTN(12,1,N,RN) :: DISPLAY AT(24,1):RN ::
    GOTO 9999
10000 SUB DEFAULTN(R,C,N,RN) :: DISPLAY AT(R,C):N ::
    N$=SEG$(STR$(N),1,1)
10001 CALL HCHAR(R,C+2,ASC(N$)):: CALL HCHAR(R,C+2,30)
10002 CALL KEY(0,K,S):: IF S=0 THEN 10001 ELSE IF K=13
    THEN RN=N :: SUBEXIT ELSE DISPLAY AT(R,C):CHR$(K)::
    ACCEPT AT(R,C+1):R$ :: R$=CHR$(K)&R$
10003 ON ERROR 10004 :: RN=VAL(R$):: GOTO 10005
10004 CALL SOUND(200,110,5,-4,5):: DISPLAY AT(R,C):N ::
    ON ERROR STOP :: RETURN 10002
10005 SUBEND
```

Ed Machonis discovered an easy way to count the words in a TI-Writer file, using TI-Writer itself. Just put in a line before line 0001, with .LMO;RM1;FI;PLnnn with nnn being the sector length of the file multiplied by 40. Save it, go into the Formatter and print it to disk under a different filename. Return to Editor, load the resulting file, page through it with FCTN[4] counting any blank lines, subtract the number of blanks from the last line number, and that is it! The Formatter takes about one minute to count 1000 words. If the resulting file is very large, you may have to load it in two sections.

```
100 M$="POS WILL FIND THE FIRST OCCURRENCE OF A
    SUBSTRING WITHIN A STRING BUT I OFTEN NEED TO FIND
    THE LAST OCCURRENCE SO I WROTE THIS SUBPROGRAM"
105 INPUT "SUBSTRING?":L$
110 CALL LAST(M$,L$,P):: IF P=0 THEN PRINT "NOT FOUND"
    :: GOTO 105 ELSE PRINT SEG$(M$,P,255):: GOTO 105
120 SUB LAST(M$,L$,P):: X=1
130 Y=POS(M$,L$,X):: IF Y=0 THEN P=0 :: SUBEXIT ELSE
    Z=Y
140 X=Y+1 :: Y=POS(M$,L$,X):: IF Y=0 THEN P=Z ::
    SUBEXIT ELSE Z=Y :: GOTO 140
150 SUBEND
```

Here is a new way to make music. The algorithm in 110 sets up a 3-octave chromatic scale - note the N(1)=F, I have erroneously omitted it when I previously published that algorithm.

To change the key of the music you have programmed, just change the value of F. Lines 190-220 contain the part of the music that is repeated within the melody. A is the subscript of the melody note, B is the subscript number of the chord. These must be above 13, as the frequency is divided by 2 in the subroutine.

Each beat of the music has a GOSUB, to 230 to play a bass accompaniment with the first note of each bar, to 260 for the other notes of the bar. The chord note is divided by different values to play the three notes of the chord in succession, and multiplied by 3.75 in the 3rd voice to produce a bass note two octaves lower in the -4 noise. The melody note is multiplied by 1.01 in the second voice to give a richer tone.

```
100 DISPLAY AT(12,3)ERASE ALL:"THE MAORI FAREWELL SONG"
    ! programmed by Jim Peterson
110 F=110 :: DIM N(36):: FOR J=1 TO 36 ::
    N(J)=INT(F*1.059463094^(J-1)):: NEXT J :: N(1)=F ::
    T=-999
120 GOSUB 190 :: A=30 :: B=23 :: GOSUB 230 :: GOSUB 260
    :: GOSUB 260 :: A=32 :: B=28 :: GOSUB 230 :: GOSUB
    260 :: GOSUB 260 :: A=28
130 GOSUB 230 :: GOSUB 260 :: GOSUB 260 :: A=30 :: B=23
    :: GOSUB 230 :: GOSUB 260 :: A=28 :: GOSUB 260 ::
    A=27 :: GOSUB 230 :: GOSUB 260
140 A=28 :: GOSUB 260 :: A=30 :: GOSUB 230 :: GOSUB 260
    :: GOSUB 260 :: GOSUB 230 :: GOSUB 260 :: GOSUB 260
    :: GOSUB 190
150 A=30 :: B=23 :: GOSUB 230 :: GOSUB 260 :: GOSUB 260
    :: A=32 :: B=16 :: GOSUB 230 :: GOSUB 260 :: A=28
    :: GOSUB 260
160 A=33 :: B=23 :: GOSUB 230 :: GOSUB 260 :: A=32 ::
    GOSUB 260 :: A=25 :: B=13 :: GOSUB 230 :: GOSUB 260
    :: GOSUB 260
170 A=27 :: B=23 :: GOSUB 230 :: GOSUB 260 :: GOSUB 260
    :: A=28 :: B=16 :: GOSUB 230 :: GOSUB 260 :: GOSUB
    260
180 B=28 :: GOSUB 230 :: GOSUB 260 :: GOSUB 260 :: B=16
    :: GOSUB 230 :: GOSUB 260 :: GOSUB 260 :: GOTO 120
190 A=32 :: B=28 :: GOSUB 230 :: GOSUB 260 :: GOSUB 260
    :: A=28 :: B=16 :: GOSUB 230 :: GOSUB 260 :: A=30
    :: GOSUB 260
200 A=32 :: B=28 :: GOSUB 230 :: GOSUB 260 :: GOSUB 260
    :: B=16 :: GOSUB 230 :: GOSUB 260 :: GOSUB 260 ::
    B=28 : : GOSUB 230 :: GOSUB 260
210 A=30 :: GOSUB 260 :: A=33 :: B=23 :: GOSUB 230 ::
    GOSUB 260 :: A=27 :: GOSUB 260 :: A=28 :: B=16 ::
    GOSUB 230 :: GOSUB 260 :: GOSUB 260
220 B=28 :: GOSUB 230 :: GOSUB 260 :: GOSUB 260 :: B=16
    :: GOSUB 230 :: GOSUB 260 :: GOSUB 260 :: RETURN
230 CALL SOUND(T,N(A),5,N(B)/1.585,9,N(B)*3.75,30,-4,9)
    :: GOSUB 290
240 CALL
    SOUND(T,N(A),5,N(B)/1.334,9,N(B)*3.75,30,-4,9)::
    GOSUB 290
250 CALL SOUND(T,N(A),5,N(B)/2,9,N(B)*3.75,30,-4,9)::
    GOSUB 290 :: RETURN
260 CALL SOUND(T,N(A),5,N(A)*1.01,5,N(B)/1.585,9)::
    GOSUB 290
270 CALL SOUND(T,N(A),5,N(A)*1.01,5,N(B)/1.334,9)::
    GOSUB 290
280 CALL SOUND(T,N(A),5,N(A)*1.01,5,N(B)/2,9)
290 FOR D=1 TO 20 :: NEXT D :: RETURN
```

Memory full...... Jim Peterson

continued from page 18

Hair: Brown
Powers: Dr. Octopus can mentally control his four electronically powered telescopic limbs to great effect. Each tentacle is five inches in diameter, can be extended from a length of six feet up to a maximum of twenty four feet. The arms each end in three pincers capable of gripping with force of up to 175 pounds per square inch, and rotating a full 360 degrees in a screwdriver like fashion. Each tentacle can lift 3 tons and travel at speeds of up to 90 feet per second. By spinning his arms like a giant fan Dr. Octopus can create a wind of up to 50 miles per hour. The limbs can be separated from their harness and controlled by Octopus at distances up to 900 miles.

LIZARD - foe
Real name: Dr. Curtis Conners.
Occupation: Research biologist.
Identity: Secret.
Base of operations: New York City and West Palm Beach, Florida.
Origin: Conners was an army surgeon until his arm was amputated following a wound in the Korean war. He turned to the study of reptiles and became a leading authority. He discovered the chemical that allows reptiles to regenerate, and when he applied it to himself he found that not only was his arm regenerated but his whole body took the form of a reptilian humanoid.
Height: 5'11"
Eyes: Blue
Weight: 175lbs
Powers: The lizard can lift up to 12 tons, jump over 18 feet, and run at 45 miles per hour. His reactions are about twice the speed of normal man, and his 6'6" tail can be moved at a speed of 100 feet per second.

HYDRO-MAN - foe
Real name: Morrie Bench.
Occupation: ?
Identity: Unknown.
Origin: ?
Powers: Hydro-man powers known up to date are that he is a man who can transform all or part of his body into water.

CHIEF EXAMINER - ?
The mysterious overseer of the QUESTPROBE Series, as yet very little is known about him.

The Forth Column

Forth Forum <8>
by George L. Smyth

This month I am going to quickly go over infinite loops and add a few things that I have skipped the past few months. I will also mention that Forth does not have a GOTO word and then turn around and introduce GOTO word. There is a good reason for the lack of this statement in Forth, since it promotes unstructured programming. BASIC produces spaghetti code, code which pops all over the place and makes reading large programs at times next to impossible. Actually, BASIC can be structured, but few programmers seem to feel that this is important from the programs I have read. The lack of this statement in Forth helps force (like it or not) modularized structures. Then again, far be it for me to keep you from using bad programming practices.

Infinite loops

I have never really been fond of infinite loops. I guess the reason is because it is so easy to lose control of the system and be forced to reboot. But there are practical applications for these constructs, such as writing a Forth system. To clarify, an infinite loop is a loop which has no predetermined number of loops. Last month we went over the DO..LOOP construct, in which a predetermined number of loops is established before execution of the construct (although we can change that with LEAVE). The BEGIN..UNTIL construct loops forever unless a condition arises which will force it out of its loop.

The word BEGIN acts as a point of return. The word UNTIL examines the value at the top of the stack and acts on it as a Boolean flag. If the flag is false ("0"), then execution returns to the word BEGIN. If you have a case where you always want the words between BEGIN and UNTIL to repeat, you can place a "0" on the stack just before the word UNTIL to ensure this. For example:

```
: GREATEST BEGIN ." I am the greatest" CR 0 UNTIL ;
```

Of course, like I said, you lose control of the system when you do this because there is no way to "get out" of the loop short of turning off the computer (although, as I said, sometimes you may want that). If you can remember to put it in, there is a word which will allow you to leave this loop the same way one might leave Extended BASIC.

?TERMINAL (--- f) This word looks to see if the break key (FCTN[4], CLEAR) is being pressed and leaves a flag on the stack to indicate this. This word is best placed just before the word UNTIL to test for activation of the break key. If the break is not in the process of being pressed when the word ?TERMINAL is executed, a false flag is placed on the stack and the word UNTIL returns execution to the word following BEGIN. If you are pressing the break key when ?TERMINAL is executed, a true flag is placed on the stack in front of UNTIL, and the construct is left.

```
: GREATEST BEGIN ." I am the greatest" CR
?TERMINAL UNTIL ;
```

Similar to this construct is the BEGIN..WHILE..REPEAT construct. The word WHILE looks on the stack to check for a Boolean value. If the value is true, then execution continues. If the flag is false, the construct is left. The word REPEAT always passes execution back to the word following BEGIN.

```
: GREATEST BEGIN ." I am " ?TERMINAL 0= WHILE ." the
greatest" UNTIL ." very vain !!!" ;
```

Potpouri

Next I will continue with a couple of words which have kind of fallen through the cracks during the course of the last half dozen articles.

GOTOXY (c r ---) No, Forth does not have a GOTO statement. This word places the cursor on the designated column and row of the screen. Remember that the screen row and column numbers begin with '0'.

LIST (scr# ---) This word lists one specified Forth screen to your display. One good use for this word comes into play when you do not have the editor loaded and you want to just look at a screen or two of information. If you have selected your printer to be your output device the screen will be dumped to your printer. The screen does not have to be written in Forth, but may contain any series of characters. After all, how do you think I get hints when I play adventure games?

SWCH (---) If you have loaded -PRINT into the system, execution of this word will send the output normally sent to your screen to your printer. Whatever printer parameters you set up on screen 72 will be the output device accessed when this word is invoked.

UNSWCH (---) This word (also requiring -PRINT to be loaded) directs output back to the screen. If you are already sending to the screen, this word will have no effect. These past three words can be combined to send disk information to your printer. The word I have in my system is:

```
: SLIST ( scr# --- ) SWCH LIST UNSWCH ;
```

TRIAD (scr# ---) This word is part of fig-Forth's system which sends three Forth screens to your printer, as long as you have -PRINT entered. When the screen number is entered, that number is considered part of a triad of screens, with the initial screen being evenly divisible by three. For instance, if '17 TRIAD' is entered, screens 15-17 will be listed to the screen, as screen 17 is part of the 15, 16, 17 triad, screen 15 being evenly divisible by three. You will note that most programs written by Forth programmers begin with a screen evenly divisible by three to take advantage of this feature. A couple other features: blank screens will not be printed and after printing three screens, the text on line 15 of screen 4 will be printed at the bottom of the page before the form feed. On our system, unless you change it, the text "TI Forth --- a fig-Forth extension" will be printed

You may have noticed that the last several words I mentioned are the first words that I have mentioned that require any type of extension. If you followed my first article explaining how to configure a quick booting system disk, -PRINT is already loaded. If you decided not to load that particular extension, entering the word -PRINT will load screens 72 and 73, which contain the definitions to these words. As a matter of fact, now might not be too bad a time to go to your Forth manual and lookup screens 72 and 73 in Appendix I. You will notice that one of the neat things about Forth is that some of it is written in Forth. When you are feeling a bit braver, you can begin modifying the source code for the extensions. The general rule of programming is first to get your program working and then optimize it. The Forth extensions certainly are not optimized! Of course nobody can fault TI or the authors for this, as this was a project tossed into the bit bucket when TI pulled out of the home computer market. If nothing else, you can get a bit of flavor of how a Forth word is written. By the way, you will also get an example of the output produced by the word TRIAD.

Well, I guess that that is about it for this month. Next month I will go over a few more miscellaneous words and try to think of something else to fill these pages up with. If you started reading "Starting Forth" by Leo Brody (very highly recommended) and had trouble and have been patiently following these tutorials (whew!), you can return to that book. I have gone over most of the information presented in the first six chapters, so reading it now will be close to review. Actually, the tutorial is close to ending, so I hope that sometime in the not too distant future I can get my hands on some interesting material to play with. Till next month, keep Forthing!

The Forum

Thought for the month:
"Program complexity grows until it exceeds the capacity of the programmer to maintain it!"

Before getting my hands on Forth I wrote an adventure game. I felt that it was fairly well written but I ran across one problem, I bought a disk drive.

My attempt to run the program from disk was not successful because the program already filled the 16K memory to the gills and now the disk drive was commanding even more memory. I decided to cut the program back just enough to allow for the memory that my drive had taken away. After working numerous hours I decided to scrap the program and start again from scratch.

My problem was that I knew nothing about modularized programming and when I went back to rework my code, I found that my program jumped all over the place. I had an incredible amount of trouble just following program execution. As it turned out, I stumbled into the technique of programming in reuseable modules and completed the adventure game (let me know if you want a copy). If I had written the game in Forth, not only would my code have been more compact, but if well written, could have been modified much more easily (this is one of the major contentions presented in Leo Brodie's "Thinking Forth").

The culprit is (mainly) the GOTO statement. When programming in BASIC, we tend to rely on it too much because the language (designed for beginners) cuts us slack to the point that poor programming practices can be developed. BASIC programs often jump all over the place like a grasshopper. Forth does not allow this. The language was developed in an attempt to force programmers to write efficiently.

This is why I was so amused when I was browsing through the magazine rack several months ago and noted in Dr. Dobbs Journal that someone had written a Forth word which acted like a GOTO statement. Someone else, however, took this seriously and presented it on Compuserve. I thought that it was interesting in that it did something I mentioned last month you should rarely do (unless you are doing something tricky, which we are here), take a value off the return stack without replacing it. Here is the word:

```
    : GOTO ( pfa --- ) CFA R> DROP EXECUTE ;
```

The idea here is to place the parameter field address of the word to be executed on the stack. 'CFA' converts this into a code field address and places that on the stack. 'R> DROP' removes the address on the top of the return stack and places it on the parameter stack to be discarded. 'EXECUTE' executes the definition of the word who's CFA is found on the stack.

This is how the word was given to me. This did not seem convenient so I decided to hack the word a bit and make it a bit more useful. I came up with two words: GOTO and GTO. Here are my definitions:

```
    : FNDWRD -FIND 0= IF ." Word not found" QUIT THEN ;
    : GOTO FNDWRD DROP R> DROP CFA EXECUTE ;
    : GTO FNDWRD DROP CFA EXECUTE ;
```

FNDWRD uses the word -FIND to look through your dictionary for the word that follows it (see your TI Forth manual). If the word is not found, this fact is printed out and execution ends. If it is a legal word then execution continues. Since -FIND leaves a value we are not using we DROP it. If we only want to execute the named word we include 'R> DROP'. This drops the return value on the return stack which, in effect, says that there is nothing else for the system to execute, so say "ok" and stop. If we do not include this pair of words, execution continues to whatever follows the word after GOTO. I have not worked with these words other than to write them, so perhaps someone can find good use for them. In the meantime, I will not hold my breath. All in all, these words are pretty useless, but then again this is an opportunity to write unstructured Forth programs. Let me know how it goes.

Well, that concludes this month's edition. Next month I will fill in couple more cracks and then begin talking about how to write a program in Forth. Then again I might try forcing some graphics on you. Forth's graphics will convert any BASIC programmer with an open mind who is interested in the graphics available with the TI. If anyone has any comments on what they would like to read about or anything in general concerning Forth, go ahead and give me a call any evening (other than when the Caps are playing) or write to:

George L. Smyth
3017 Sylvan Drive
Falls Church, VA 22042

continued from page 22

```
         IF (m[k,z]=1) AND ((c=2) OR (c=3)) THEN
          n[k,z]:=1;
         IF (m[k,z]=1) AND ((c>3) OR (c=0)) THEN
          n[k,z]:=0;
         key(3,t,s);
         IF s=1 THEN
          GOTO rerun;
      END;
     END;
    cursor(24,2);
    write("generation ",g);
    draw(m,n);
    IF y=75 THEN
     REPEAT
      key(3,k,s)
     UNTIL s=1;
    UNTIL false;
   rerun:
  END;

 PROCEDURE setup(VAR  k,z:INTEGER;VAR  m,n:ARRAY[12,12]
OF INTEGER);
  VAR j,jj:INTEGER;
  BEGIN
   REPEAT
    REPEAT
     cursor(k+2,z+5);
     IF (k=1) OR (k=12) OR (z=1) OR (z=12) THEN
      write ("+")
     ELSE
      write("?");
     REPEAT
      key(3,j,jj);
     UNTIL ((j=32) OR (j=65)) AND (jj=1);
    IF j=65 THEN
     BEGIN
      cursor(k+2,z+5);
      write("O");
      m[k,z]:=1;
      n[k,z]:=1;
     END
    ELSE
     BEGIN
      cursor(k+2,z+5);
      write("~");
      m[k,z]:=0;
      n[k,z]:=0;
     END;
    z:=z+1;
   UNTIL z=13;
   z:=1;
   k:=k+1;
  UNTIL k=13;
 END;

 { This is the actual program!:- }

 BEGIN
  cls;
  rules;
  REPEAT
   k:=1;
   z:=1;
   cls;
   text;
   setup(k,z,m,n);
   think(m,n);
  UNTIL false;
 END.
```

It takes about a second for each generation, much faster than is possible in TI BASIC! It would be simple to amend the size of the life universe, but the larger the universe the longer the creation time!

continued from page 30

ATICC (September1988) (Adelaide User Group): Page 1, Coordinator mentions a hardware item which he calls a Data Bank by Colin Cartright which uses a Z80 CPU and interfaces with the TI99/4A. Page 3 has a warning on using versions of DM1000 over 3.5. Page 8, user written subprograms in Extended BASIC. Pages 9 and 19 have programs written in the new ATICC "G" language. Page 13, the use of the "edge" character in BASIC and another article on the RS232 connection and finally page 19 has a comparative test of 10(!) languages for the TI99/4A.

Melbourne TIMES (September 1988): Pages 4 to 12 have a handy TI-Writer Reference Guide. Page 13, "UCSD Pascal" by PR Gleed. Page 15, Funnelweb 4.10/4.11/4.12 configuration instructions and finally Page 21 is about PR Gleed's trip to the USA.

BUG BYTES (October 1988) (Newsletter of TIBUG): Page 4 has an article on "Tape User's Word Processor" using the built in editor ie TI BASIC. Page 6, "Geneve Corner" has the latest news on the Geneve and the HFDCC including a hint on adding an extra 32 Kbyte of no-wait memory for the Geneve.

Hunter Valley 99ers (September 1988): Al Lawrence reports on changes to Funnelweb, TIsHUG/HV99ers get together for November (which somehow never eventuated), Albert Anderson talks about Neil Quigg's QUEST RD200 32K RAMdisk for the PE Box. The 512Kbyte card (expandable to 1 Mbyte) includes 32Kbyte CPU memory and is battery backed. Cost is $140 for an operational (socketed) board with 32Kbyte installed. The HV club has also been advertising for consoles and equipment. A timely reminder that Fairware authors should be compensated by program users eg Barry Boone (Archiver) has received a contribution from only ONE person in Australia. What can we do about this? "In The News" by Joe Wright mentions the 64Kbyte upgrade to the Peter Schubert Multifunction card for the PE Box. Two banks of CPU 32Kbyte are available by switching CRU address. This could easily be made available on an upgraded DSR ROM. TI Keys (Fairware) allows quick input of commands from a single keystroke. Keys may be re-assigned and macros are able to co-exist with some programs eg Funnelweb. Random Bytes (Bob Carmony) mentions a very useful tip for the TI-Writer Formatter: when prompted for the pages to be printed, you can enter, eg, 3 - E (with spaces either side of the "-") to print out only pages 3 to the end of the document. He also writes on how to modify BASIC programs to suit your own special requirements. Tony McGovern contines his articles on "Assembly Squeezing" on moving blocks of information and ties with the TI99/4A community in Sweden and a review of John Birdwell's Disk Utilities and its compatability with other hardware items such as controller cards and RAMdisks. A review of "Genealogy Records and the Computer" by Beryl Thompson (Canberra Group).

HV99ers (October 1988): Al Lawrence calls for early orders for the QUEST RD200 RAMdisk (compatable with the Horizon) to reduce costs through economies of scale. Bare board with full instructions is only $60. Tips on Funnelweb V4.1 through V4.12, another mention of the TIsHUG/HV99ers get together, an updated QED loader for the 32Kbyte Supercart and mention of a new program from David Caron to convert the TI99/4A keyboard into a piano. Random Bytes from Bob Carmony outlines a Extended BASIC program which converts DV80 to DF80 file for creating object code ready to run, a hint on cleaning the GROM port using Tandy degreaser (#64-2322) said to cure 90% of lockup problems, a call from Ron Kleinschafer for payment to authors for Fairware programs in use and a re-appraisal of the EPROM Programmer and software first published in August 1987, Tony McGovern on Assembly Squeezing and string handling, the Funnelweb operating system and how it interfaces with other programs and finally Jack Sughrue writes on "Adventuring". o

continued from page 13

-> Now you are in a strange maze which is actually Arthur's brain. Just move a few directions or so, until a particle blocks your way. and be sure not to drop anything!

(Once you are by the particle type...) "GET PARTICLE"

-> And it is back to The Heart of Gold we go!

[E], "HEAR", "HEAR", "HEAR", "HEAR", [S,S,S,D], "DROP AWL", [U,N,W], "PUSH PAD", [E,U], "WAIT" (until the ship's computer announces that Magrathean missiles have been launched at the ship), "TURN SWITCH"

-> Now the missiles should have changed their form slightly.

[D,W], "GET TEA", "GET NO TEA", [E,S], "SHOW TEA AND NO TEA TO DOOR", "DRINK TEA", [N,U]

-> Now would be a good time to save the game.

"TURN SWITCH", (if you see "BUG #60", that is good. If not, restore the game and keep turning the switch until you do.)

(After "BUG #60"), [E], "FEEL", "FEEL", "FEEL", "FEEL", "DRINK LIQUID", "PLANT ALL FLUFF IN POT", "$HE"

-> You will soon be back on The Heart of Gold, but not for long.

[E], "HEAR", "HEAR", "HEAR", "HEAR", [S,S,U], "PLUG SHORT CORD INTO SMALL RECEPTACLE", "PUT DANGLY BIT IN CUP"

-> Save the game again.

"TURN SWITCH", (restore and try again until you see "BUG #60" again), [E], "FEEL", "FEEL", "FEEL", "FEEL", "DRINK LIQUID", "EXAMINE POT", (there should be a small stalk growing there), "GET POT", "$HE"

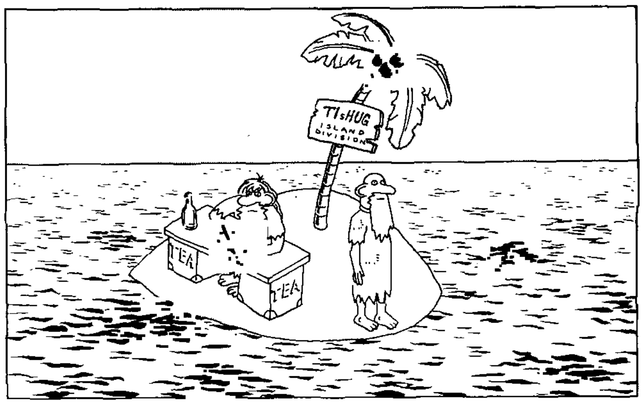
-> Relax, because this is the last time you will be visiting the "dark". Enjoy it while it lasts.

[E], "HEAR", "HEAR", "HEAR", "HEAR", [S,S,U,W,D,S,W], "MARVIN, OPEN HATCH", "EAT FRUIT", (eating the fruit will tell you which random tool Marvin needs to open the hatch), [E,D], "DROP ALL", "GET XXXXX" (whichever tool Marvin needs), [E], "WAIT", (until Marvin shambles in and asks you for the tool), "GIVE XXXXX" (tool), [W,D]

-> Congratulations! You have stepped onto the surface of the legendary planet of Magrathea and have solved "The Hitchhiker's Guide to the Galaxy" with all 400 points!.

Watch for more Infocom games in the "Hitchhiker" series soon to come.

— End of session — o



"Dear Editor. Joe and I are writing down our comments for the Survey. We both strongly agree that there should be far more cartoons!"

continued from page 1

Last month I mentioned that I would like Terry to ask Jim Peterson for his Tips from the Tigercub #29 to #42 next time he wrote to him. Well clearly Jim Peterson reads his copy of the TND as today (18 November) a parcel arrived from Terry with a disk and letter from Jim saying "here they are". When I thought about it, I realised that the postal services of both countries should be congratulated on their speedy service. The TND was air mailed to Jim on 26th October and his letter is dated 7th November. The parcel from Terry was mailed yesterday in Sydney so the round trip for mail which was not just a standard letter took just over 3 weeks, including all handling delays, for 3 separate mailings. Apart from that I must thank Jim for his very prompt response, which tells me that he values our efforts at presenting information. I hope he appreciates what I do to his originals to fit them into our format. I do not know about you but I think that all Jim's articles are full of useful information and tips. The main problem I have is finding the time to take advantage of all that information. We now have enough Tips articles to print one a month for the next year and a half!

****

In his letter, Jim Peterson takes Terry to task for saying that Library Disk A270 contains files giving the contents of Jim's public domain library. This is not true. Please note that the files have the contents of the <u>copyright</u> programs which Jim is wanting to sell to us. We will be printing this catalog when I have time to reformat it and then you can use it to order some quality software from Jim. He does have a public domain catalog which Terry probably has, if you ask him, which describes about 3400 programs.

****

While on explanations, Lou was a bit upset with what I did to the article on the 64Kbytes on the 16 bit bus. The problem related to the origin of various signals needed by the memory. To hopefully clear things up (or perhaps to really confuse you) here is a list of some signals which you can obtain at more than one point. Pin 8 of the 74LS21 is connected to pin 20 of both 62256s and then goes to pin 12 of U606 or to pin 11 of U608 and U609 (system RAM which has been removed). Similarly, pin 22 of both 62256s are connected to DBIN(L), which is available at pin 9 and pin 10 of U508 or pin 2 and pin 11 of U602 or pin 9 of U603 or pin 1 of U614 or pin 9 of U604. Pin 27 of both 62256s are connected to WE(L) which is available at pin 5 of U602 or pin 16 of U608 and U609. The preferred attachment points are shown on the diagrams, except that U602 is shaded with no connections shown to it. The connections to the holes formerly occupied by U608 and U609 are not shown. Just to whet your appetite, Lou and I are hoping to give the circuitry to allow the wait states to be introduced at the flick of a switch, to allow all those troublesome games to work once more. Stay tuned to this channel!

****

Included with this TND are the forms for nomination to the board. Please discuss this at your regional group meetings and try to work out where you want TIsHUG to be heading and whether you can afford the time to help it to get there. This year has been hard on our small group and we feel that we have shown in some small part how we want TIsHUG to appear to the outside world. Now we would like to see some others take up the challenge and make TIsHUG better for all its members so that it will remain strong and vital for years to come. Support and encouragement needs to be given to those who are putting things back into the group. I firmly believe that there has not been a consistent policy followed by the board in this regard. If there has been then there is a communications problem in that there is no reporting of what the board is doing on our behalf. I hear from Ross that he has finally been allowed to upgrade the BBS disk controller to double density. Why did not the board announce this as part of publicity about how great the BBS is and how confident the board is in the future? From the overwhelming lack of information and excitement I could be excused for wondering if those at the top do not believe that TIsHUG will last more than 6 months. I do not believe that we are approaching the end of the useful life of our computer. Goodness, most of the "successful" computers cannot do all that our computer can do and it costs far more to do what they can do. Of course they may run a bit faster and have a bit more memory, but with a RAMdisk or two our computer is fast enough for home use. It will only take a while and some bright young (anyone under 90) programmer to use c99 (for example) to write good software which uses swapping to RAMdisk to expand the abilities of many programs like word processing. In fact it looks like Larry Saunders has found just such a program. Our computer has been so well engineered that it allows for expansions so readily and yet still retains its essential good character. What we need are a few people in charge of TIsHUG who are not scared of the future and who are willing to take a chance on what looks a good thing. End of sermon for this year.

Newsletter Roundup

by Lou Amadio

<u>Overseas Newslet</u>

The PUG Peripheral (October : This club has decided to purchase a printer ribbon re-inker (what a good idea, TIsHUG). Mention of BATCH processing for the TI99/4A (from St Louis 99ers) and Super Extended BASIC. Page 2, hardware news, where Gary Taylor talks about the problems of providing backup power for the Myarc 512Kbyte card. Page 3, Audrey Buchar on "Multiplan Part III", the Index command, the IF statement and Work Tables. Page 4, Mickey Schmitt on "Getting the Most Out of Cassette System - No 18", understanding, creating and using cassette files. TI-Writer Part 9 by Stan Katzman discusses the Formatter "dot" commands. Page 6 has the 3rd tutorial on Forth for the novice. Page 7 has a Funnelweb Tips column. Page 10, article on "Computing For Your Retirement" from St Louis 99ers. Page 11, Hardware article on a very interesting disk drive tester to assist in fault finding.

SNUG (September 1988) (Sacramento 99er User Group): Page 2, a little history on TI, Page 3, BASIC to FORTRAN conversion By Mike Morrow, Page 5, Death of a Computer Part 1D by Ralph Nocera.

The Tacoma Informer (October 1988) (Tacoma 99ers Users Group): Page 3, Boyd Shugert's article on printing graphics from BASIC. Page 5, Jack Sughrue on the "Good Old Days - Part 3 - The Dark Ages. Page 7, Jim Lugue on a "Jiffy Flyer" and page 9 has a questionaire for club members.

ROM (September 1988) (The Users Group Of Orange County CA): Page 3, "Assembly Language" by Adrian Robinson gives a screen dump for the Jiffy Flyer. Page 5, "And So Forth #34" By Earl Raguse on menu driven Forth utilities. Page 7, "TI Bits Number 20" is an article on how your TI99/4A saves text files. Finally a reminder on ensuring that we compensate Fairware authors for their programs.

TI FOCUS (October 1988) (Channel 99 Users Group, Ontario): Page 3, changing colours in Disk Utilities V4.1 by John Van Weelie. Page 4, Archiver III V3.02 by Barry Boone. Page 6, "Sector Editing For The Novice" by Tom Arnold and Page 9 has a Funnelweb and TI-Writer tutorial on printer control codes by Tor Hansen and Dwayne Verhey.

The Ottawa TI-99/4A Users Group Newsletter (October 1988): Page 5, "How to Print a Sheet From Multiplan" by Jane Laflamme. Page 6 has a review (by Charles Earl) of Myarc's New HFDC Card. Overall impression was quite good apart from a bug with Archiver V3.02. Page 8, "Expanding Extended BASICs Powers - Writing Assembly Language Routines: Part 1" by David Caron. Page 10 has a BASIC "screen" calendar by Lucie Dorrias .

<u>Local Newslet s</u>

TI-UP TITBITS (September 3): Page 7 has an article on the RS232 connection (from Manners Newsletter). Page 10, TI-Writer codes table (from PUG Peripheral), and the Geneve choice by Garry Christensen.

continued on page 29

Regional Group Reports

Meeting summary.

Banana Coast	11/12/88	Sawtell
Carlingford	21/12/88	Carlingford
Central Coast	10/12/88	Toukley
Glebe	8/12/88	Glebe
Illawarra	19/12/88	Keiraville
Liverpool	9/12/88	???
Northern Suburbs	22/12/88	Davidson
Sutherland	16/12/88	Jannali

BANANA COAST Regional Group
(Coffs Harbour area)

Regular meetings are held in the Sawtell Tennis Club on the second Sunday of the month at 2 pm sharp. For information on meetings of the Banana Coast group, contact Kevin Cox at 7 Dewing Close, Bayldon, telephone (066)53 2649, or John Ryan of Mullaway via the BBS, user name SARA, or telephone (066)54 1451.

CARLINGFORD Regional Group.

Regular meetings are usually on the third Wednesday of each month at 7.30pm. Contact Chris Buttner, 79 Jenkins Rd, Carlingford, (02)871 7753, for more information.

CENTRAL COAST Regional Group.

Meetings are normally held on the second Saturday of each month, 6.30pm at the Toukley Tennis Club hall, Header St, Toukley. Contact Russell Welham (043)92 4000

GLEBE Regional Group.

Regular meetings are normally on the Thursday evening following the first Saturday of the month, at 8pm at 43 Boyce St, Glebe. Contact Mike Slattery, (02)692 0559.

ILLAWARRA Regional Group.

Regular meetings are normally on the third Monday of each month, except January, at 7.30pm, Keiraville Public School, Gipps Rd, Keiraville, opposite the Keiraville shopping centre. Contact Bob Montgomery on (042)28 6463 for more information.

LIVERPOOL Regional Group

Regular meeting date is the Friday following the TIsHUG Sydney meeting at 7.30 pm. Contact Larry Saunders (02)644 7377 (home) or (02)759 8441 (work) for more information.

NORTHERN SUBURBS Regional Group.

Regular meetings are held on the fourth Thursday of the month. If you want any information please ring Dennis Norman on (02)452 3920, or Dick Warburton on (02)918 8132.

SUTHERLAND Regional Group.

Regular meetings are held on the third Friday of each month at the home of Peter Young at Jannali at 7.30pm. Group co-ordinator is Peter Young, (02) 528 8775. BBS Contact is Gary Wilson, user name VK2YGW on this BBS.

The October meeting welcomed Steven Carr, from the Club Shop, who will become a regular visitor to all Regional Meetings, to make the shop merchandise more readily available.

At future meetings we will prevail upon Peter Pederson to show us a few of the features and tricks associated with Multiplan.

Many thanks to Peter Schubert who demonstrated some of his hardware including the AT music card and the RAMdisk EPROM Operating System.

TIsHUG in Sydney

Regular meetings are normally at 2pm on the first Saturday the month, except January and possibly other months with public holidays on that weekend, at the Woodstock Community Centre, Church Street, Burwood.

Meetings planned for the next few months.

December 3 - Christmas Party.

Given a fine day, this will be one of the major events of the year, with plenty of food and drink available and a great chance to chat with fellow members in a social relaxed atmosphere. There will be plenty of software released for this meeting, so you will have plenty to keep you occupied over the Christmas/New year holiday break.

January - no meeting scheduled. This is the month when we can all relax and enjoy the summer break.

February 4 (1989) - Annual General Meeting. Providing the club auditorium is available, this meeting will be held at the Burwood RSL Club, Shaftsbury Road, Burwood. This meeting will see the election of a board of Directors to run the group for the following 12 months. Full details and nomination forms will appear in the December issue of the TND.

*** Stop Press ***

Just in from Russell Welham from the Central Coast Regional Group. Their next meeting will be on Sunday 11th December and will be a Christmas Party at the Toukley Tennis Club at 10am. Ross Mudie will be there with his train set and all are welcome to bring their food and drinks and join in for a Barbeque. The energetic can play tennis or go swimming.

TI-Writer Mnemonic memory Tricks

from ROM, USA

CTRL	Mnemonic	Function	Alternate
A	Advance Down	Roll Down	FCTN 4
B	Back Up	Roll Up	FCTN 6
C	Command Mode	Command Mode	FCTN 9
F	Flyaway Character	Delete Character	FCTN 1
G	Make a Hole for Char	Insert Character	FCTN 2
H	Hop Back to Last	Last Paragraph	CTRL 6
I	Indent	Tab	FCTN 7
J	Jump to Next	Next Paragraph	CTRL 4
K	Kill to End of Line	Delete to Line End	
L	Leap Home	Home Cursor	
M	Make New Paragraph	New Paragraph	CTRL 8
N	No More Line	Delete Line	FCTN 3
O	Open Blank Line	Insert Blank Line	FCTN 8
P	Page Beginning	New Page	CTRL 9
R	Reformat	Reformat	CTRL 2
T	Tab Back	Back Tab	
V	Veer to Left	Cursor to Line Start	
W	Word Tab	Word Tab	CTRL 7
Y	Yank Margin Control	Left Margin Release	
Z	Zip Back	OOPS!	CTRL 1
		Screen Color	CTRL 3
		Dupe Line	CTRL 5
		Next Window	FCTN 5
		Word Wrap	CTRL 0
		Line Numbers On/Off	FCTN 0

Note: The arrow keys work with both the Function Key or the Control Key.

continued from page 16

```
2300 READ XC,YC,A$,B$ :: DIS
PLAY AT(XC,YC):A$ :: ACCEPT
AT(XC,YC+LEN(A$))VALIDATE(B$
):AN$ :: RETURN
2310 DISPLAY AT(22,1):A$ ::
RETURN
2320 DISPLAY AT(24,1)SIZE(16
):A$ :: RETURN
2330 CALL KEY(0,K,S):: IF S=
0 THEN 2330 ELSE RETURN
2340 SUB AR(P1,P2,Z,M)
2350 IF M<3 THEN 2370
2360 DIM M$(19):: MN$=RPT$(C
HR$(1),28):: FOR X=1 TO 19 :
: M$(X)=MN$ :: NEXT X :: SUB
EXIT
2370 IF M=2 THEN 2390
2380 Z=ASC(SEG$(M$(P1),P2,1)
):: SUBEXIT
2390 M$(P1)=SEG$(M$(P1),1,P2
-1)&CHR$(Z)&SEG$(M$(P1),P2+1
,29-P2):: SUBEND
```