HUNTER VALLEY 99ERS USERS GROUP

HOME COMPUTER NEWSLETTER

JULY 1989

REGISTERED BY AUSTRALIA POST PUBLICATION

NUMBERNBG8023

YOUR COMMITTEE

PRESIDENT

Peter Smith
8 Glebe St.,
EAST MAITLAND 2322
Phone 336164 V/tl 493361640

VICE PRESIDENT

John Paton
1 Parien Close,
RUTHERFORD 2320
Phone 326014 V/tl 493260140

SECRETARY

Brian Woods
9 Thirlmere Pde.,
TARRO 2322
Phone 662307 V/tl 496623070

TREASURER

Noel Cavanagh
378 Morpeth Rd.,
MORPETH 2321
Phone 333764

Software Librarian

Stewart Bradley
14 Hughes St.,
BIRMINGHAM GARDENS 2287
Phone 513246

EDITOR

Allen (Joe) Wright
77 Andrew Rd.,
VALENTINE 2280
Phone 468120

PURCHASING/HARDWARE CO-ORDINATOR

Alan Franks
822 Pacific Highway,
MARKS POINT 2280
Phone 459170

SOCIAL SECRETARY

Robert (Bob) MacClure
75 Deborah St.,
KOTARA SOUTH
Phone 437431

PUBLICATIONS LIBRARIAN

Ken Lynch
9 Hall St.,
EDGEWORTH 2285
Phone 585983

COMMITTEE MEMBERS

Don Dorrington
36 NELSON St.,
BARNSLEY 2301
Phone 531220

Tim Watkins
36 The Ridgeway,
BOLTON POINT 2283
Phone 592836

CONTRIBUTIONS

Members and non members are invited to contribute articles for publication in HV99 NEWS.

Any copy intended for publication may be typed, hand written, or submited on tape/disc media as files suitable for use with TI Writer (ie. DIS/FIX 80 or DIS/VAR 80). A suitable Public Domain word processor program will be supplied if required by the club librarian.

Please include along with your article sufficient information to enable the file to be read by the Editor eg. File Name etc. The preferred format is 35 columns and page length 66 lines, right justified.

All articles printed in HV99 NEWS (unless notified otherwise) are considered to be Public Domain. Other user groups wishing to reproduce material from HV99 NEWS may feel free to do so as long as the source and author are recognised.

Articles for publication can be submitted to the Editor, ALL other club related correspondence should be addressed to The Secretary.

DISCLAIMER

The HV99 NEWS is the official newsletter of the HUNTER VALLEY NINETY NINE USER GROUP.

Whilst every effort is made to ensure the correctness and accuracy of the information contained therin, be it of general, technical, or programming nature, no responsibility can be accepted by HV99 NEWS as a result of applying such information.

The views expressed in the articles in this publication are the views of the author/s and are not necessarily the views of the Committee, Editor or members.

TEXAS INSTRUMENTS trademarks, names and logos are all copyright to TEXAS INSTRUMENTS

HV99 is non profit group of TI99/4A computer users, not affiliated in any way with TEXAS INSTRUMENTS.

From President PETE'S Quill

Another year has been negotiated successfully by your club, guided by a dedicated and talented executive. On 27th June a new executive was elected which I am sure is equally as dedicated and hopefully as talented as the one which has stepped down.

To all those past executive, especially our past president, let me, on behalf of the club-members congratulate and thank you for a job well done.

I thank the club for the confidence it has shown in me and wish to congratulate all the new committee on their appointment. I am sure we will work well together.

Looking forward at what I think this year may hold in store for this club, I feel a certain apprehension because I believe that a great amount of change must take place if we are to continue learning about and with our little orphan as well as providing members with the support and fellowship which now exists..

Our membership is rather unique in that the number of local active members is about the same as the number of "out of towners". This has many implications, the main ones being that the work of running our club is left to relatively few, and that it is often hard for the club to cater for the needs of people we rarely hear from, or seldom if ever see.

I do not imply that this situation is bad, but simply wish to point out that we need to know about you and your needs and ways you can help others too.

A hot issue which has had a little discussion recently is the subject of the club diversifying the range of computers embraced by the club. (some suggest that scope exists for both Amiga and IBM which could co-exist as special interest groups etc.).

How about a letter to the editor expressing your point of view.

A reminder to all. Fees are well and truly due for this 89/90 period. Please rejoin and support the club which has supported you so well.

A calendar outline has been drawn up for major activities by members of this club for the rest of 89. Most dates are tentative at the moment as we need to know if the activities will receive your support. Once again let us know and we will try to cater for your needs.

Regards

.... Peter

CURRENT AGENDA

The July meeting will be a demonstration of the use of PR-BASE as it is applied to the software library. The intention is to have a number of volunteers offer to take some of the library discs home and record data pertaining to the material on the discs. Currently about 60% of the library is cataloged. We want to complete this ASAP so that we can send out completed catalogS to all our members.

FORTH.
The forth SIG on again at 7:00 pm 1/8/89 at Richard Terry's Surgery at Whitebridge.

COMMITTEE MEETING At Peter Smith's house in Maitland. Starting at 7:00 pm on 8/8/89.

BASIC CLASS.
At Bob MacClure's house starting 7:00 pm on 15/8/89.

GENERAL MEETING.
At Jesmond Community Centre starting 7:00 pm. 22/8/89.

random bytes

with

BOB CARMANY

First of all, welcome to the new Committee members. You will probably find out that the 'easy' job that you were promised is hardly that!! There will be moments when you wish you could back out but you'll find that it is worth the effort in the final analysis.

What is the difference between a single and double sided disk? Give up? The answer is NOTHING!! When a disk is produced, it is stamped out of a giant sheet of mylar by what resembles a big cookie cutter. The mylar core is then coated with an iron oxide compound on both sides. At this point, double and single sided disks are exactly the same! The disks are then put into a "case" with the necessary hole pre-punched and the case is sealed either by heat or by crimping the edges together. It is at this point that the manufacturer decides whether the disk id going to be single or double sided. The ONLY difference is that a single sided disk is tested for reliability on just one side and a double-sided disk is tested on both sides. In other words, the manufacturer will only guarantee a single sided disk on one side --the other side is equally at good (usually). Why are double sided disks more expensive? Because of the time involved in testing the second side!

That brings us to the proper care of a disk. Every time I see someone lay a disk down on top of his monitor or PE-Box, I think to myself "I hope there weren't any irreplaceable programs on that disk!"

Disks are MAGNETIC media and, although they will take considerable abuse, can have their data destroyed by a magnetic field. All electric devices produce magnetic fields of varying degrees. Televisions and Monitors are notorious for magnetic "leakage". There is an easy way to check for a magnetic field around your electronic equipment. Take a portable radio and turn it to an unused frequency. Then, move it toward your television or PE-Box while it is turned on. Static indicates that a magnetic field is being produced and is escaping the shielding provided for the device. Most of the fields aren't strong enough to erase a disk but why take a chance! Here are a couple of rules that will help preserve your data and programs:

1) NEVER put disks on top of electronic devices such as Monitors, PE-Boxes or televisions.

2) NEVER write on a disk label after it is put on the disk. A ballpoint pen will instantly destroy a disk.

3) Keep disks away from extremes in temperature --they can adversely

affect data stored on a disk.

4) Avoid eating or drinking when handling disks (they don't like coffee!)

Here is a program that is hypnotic and a pleasure to watch. It was originally released to Users Groups several years ago.

```
100 REM !!!!!!!!!!!!!!!!!!!!
110 REM ! SNAKE DANCE BY !
120 REM ! VAUGHN SOFTWARE !
160 REM EXTENDED BASIC REQUIRED
180 RANDOMIZE :: CALL CLEAR :: CALL
SCREEN(2)
190 B=RND!190 :: CALL MAGNIFY(1)::
CALL CHAR(96,"8")::FOR A=1 TO 28 ::
CALL SPRITE(#A,96,8,95,10,10,SGN(95
-B)!A):: NEXT A
200 D=RND!20 :: FOR A=1 TO 28 ::
CALL MOTION(#A,D,A!SGN(10-D))::NEXT
A:: C=C+1 :: IF C<25 THEN 200
210 FOR A=1 TO 28 :: CALL
COLOR(#A,RND!14+2):: NEXT A :: C=0
:: CALL MAGNIFY(2)
220 FOR A=1 TO 28 :: B=RND!14+2 ::
CALL PATTERN(#A,96):: CALL POSITION
(#A,U,V):: CALL MOTION(#A,SGN(96-U)
!9,SGN(10-V)!9):: NEXT A
230 FOR A=1 TO 28 :: CALL PATTERN(#
A,96):: NEXT A :: CALL DELSPRITE(AL
L):: GOTO 190
```

That's it for this month. It's back to the gardening for awhile!

IN THE NEWS

compiled by
BRIAN WOODS

This month marks the beginning of the end of winter. It also marks the start of the last 6 months of the 1980's. And Today! It marks the beginning of the rest of our lives. What changes, if any, should we be making to our behavior as a race, a Nation, a State or a person?

The first step, my son, which one makes in the world, is the one on which depends the rest of our days.

VOLTAIRE.

AND!!!

Nothing is so firmly believed as what we least know.

MONTAIGNE.

BROWSE.
****
Peter Hoddie from the Boston User Group has releases a programme called BROWSE. The following is from the Genial computer catalog.

Browse is a utility to aid in the management of text files. Using Browse you can easily print, view, combine , and browse test files. You can easily select a group of files from a disk and have them printed. Browse has options which allow you to start each file on a new page, print the filename, use special modes of your printer, strip control characters and more.

Browse has an extremely friendly interface including pop up windows. File selection is handled through the familiar Disk Manager 1000 interface. Many functions are available through several different keypresses for maximum convenience of use on both the 99/4A and 9640.

Over 24000 bytes of text may be loaded into memory at one time. You can have several files loaded into memory at one time. These files may be viewed using a file viewer similar to the one found on Genial TRAVeLER. The viewer also includes keys to quickly move to the beginning and end of the file. The viewer allows for 80 column on the 9640. Select groups of files may be printed from memory, so that you can view a set of files before deciding which to print.

Browse allows files to be catalogued on floppy and RAM disks as well as hard drives. It is fully configurable, with all user settings saved as part of the programme for immediate availability whenever Browse is run.

Browse is an extremely useful and easy to use utility. Browse can read TI-Writer, Funnelweb and MY-Word files. It is a must for anyone who uses their TI for word processing. Because it is written in assembly language, Browse has an extremely quick response to you commands.

Browse requires Extended Basic, editor/Assembler, or TI-Writer to run. It runs on a TI-99/4A or 9640 computer.

The list price is US$10.00 plus US$1.00 shipping.

Genial Computerware,
P.O. Box 183,
Grafton,
MA 01519
USA.

GENIAL CATALOG.
****
BY dropping a line to the above address you can get a copy of their free Genial Computerware catalog.

99/4A UPGRADES
****
These have been mentioned in a previous article, here is a bit more detail.

TI DISK CONTROLLER
This upgrade kit does not upgrade to double density, but it does allow faster drive step time (TI set their card to 20ms), and does allow you to have a fourth disk drive. Other conveniences are also added. The cost is US$19.95 and you must specify whether or not the normal or fast head step is required.

RS232 UPGRADE.
The upgrade for the TI RS232 allows you to have your TI card emulate a thermal printer (TP) which is useful for some programmes, which only support graphics output to the antiquated thermal printer. There are also software changes so that you can specify long RS232 names (RS232/2.BA=4800.DA=8.PA=0" with something shorter like "SIO" for serial I/O. This kit is only US$14.95. so that the kit may be customised to your system. Guion has a series of questions that must be answered so write him for the official order form before ordering. Both the RS232 and Disk Controller upgrades require some soldering. More details on all these packages are available by writing to the address below.

John Quion
11923 Quincy Lane,
Dallas.
TX 75230
U.S.A.

FUNNELWEB.
****
At our June meeting Tony McGovern gave a sneak preview of his lastest effort with Funnelweb for both the Digit 80 Column card and the standard TI 40 column format . Don't want to steal Tony's thunder but the 80 Column version, particularly the SD and view features are REALLY good. Wish I could afford that Digit card, still might if I can pick up a CHEAP monitor.

THATS ALL.
****
That is about it for this month.

AL'S MARKET PLACE UPDATE

WITH *Alan Franks*

As we start another membership year, here is one bit of news that might brighten your day. We are now managing to fix 99% of all crashed consoles we come across. So if you have one in need of repair get it to me either at the meetings or at home and I will do my best to get it up and running again. The only charge is for parts and a five dollar donation to the club so we can invest some money in spare sockets and chips.

The other thing I have to mention is the second hand market. In the last year or so we have managed to find good homes for a large amount of hardware and software by running adds in the local papers and buying systems then splitting it between members. In doing so we have just about cured the problem of not being able to go down to the local supermarket and buy it.

However lately there has been more people selling than I can find buyers for so here is a list of people you can contact if your in need of anything.

No (1)

Stewart Bradley our software librarian has a console, PE box with 32k, RS232, and disk controller cards for sale. You can contact Stewart at 513246 he is prepared to except any reasonable offer.

No (2)

Paul Slowey an ex member of our club also has a console, joysticks, and the following modules for sale parsec, buckrogers, tunnels of doom, hangman, ti-invaders .
You can contact Paul on 591569.

No (3)

Joanne Bowness has a console, cassette player, extended basic, and three other modules for sale.
Joanne has not got the phone on but her address is Flat 2 No. 2 Penyland st, Cardiff North.

No (4)

Mrs Barnes has a console and the following modules for sale alpiner, startrek, indoor soccer, football, attack, tunnels of doom.
You can find out more about this one by ringing 327220 the asking price is £50 ono for the lot.

No (5)

Mrs lucas has a console and speech synthesizer with about eight various modules for sale. Just ring her at 546339 with any reasonable offer.

No (6)

Richard Blakemore has a console in mint condition with joysticks and the following modules munch man, multiplication, division 1, ti-invaders, moonmine, carwars, video games 1 and extended basic all for £50 ono. Ring 828334 and ask for Richard.

No (7)

Eddy has a console with TI joysticks plus a computec joystick with the ti adapter along with the following modules fractional numbers, reading on, meteor multiplication, demolition division, hunt the wumpus, TE 2, touch typing, munchman, alligator mix, phys fitness , car wars, ti-invaders, attack, number magic, minus mission, multiplication, personal record keeper, dragon mix, alien addition, blasto, division 1, addition subtraction 2, tombstone city, early learning fun, hangman, video games 1 and a few cassette programs. All the modules are in a special carry case and the manuals have been put in a folder. You can make eddy an offer at 485598.

No (8)

Bill has a console, joysticks, ex basic about four assorted modules along with manuals plus a book on basic programing . Just ring 438024 and make Bill an offer.

No (9)

Robert Adams has a console with joysticks, ex basic, video games 1, hangman, video chess, touch typing, plus graphics pack on cassette.
Ring Robert on 665631 and make an offer.

OUT OF TOWNERS.
****
If you are looking for any hardware in particular please feel free to write to me. I keep a list of people who are wanting equipment. As the equipment becomes available I contact those people with a view to them buying it if they still want it.

Alan Franks.

EDITOR'S LAMENT

This is my first newsletter! Have to admit that I was more than a bit hesitant about taking over those duties. I felt a bit like the army recruit, who, when volunteers were called, found that he was the only person in the parade who had not taken one step backwards. Have to admit now, that it is all together that I find it very rewarding. Thanks to Brian for his assistance and advise, thanks to Al Franks, both really helpful gents.
Since I am not of a flamboyant nature you will no doubt notice that reflected in the newsletter. You will also notice that I do hold very strong feelings about this planet and it's people. I will on occasions have to beg you indulgence if I prattle on a bit.

Remember that for the newsletter to be a success it needs **INPUT**.

Joe Wright

BEATING AROUND THE BUSH

WITH Ron Klienschafer

It is some time since I wrote an article for this newsletter!, so what's been going on up in the sticks?, I suppose the major thing to keep things busy here since December is the fact that since that particular month a Quest Ramdisk was purchased and pluged into the old bucket of bolts, whats wrong with that you say?, well, after getting the Quest it was decided that if there was going to be a Ramdisk hooked to the machine then it may as well be used to its maximum capacity so it was stuffed full of memory IC's, it was only after this that it was found out that the DSR was only temporary affair untill a DSR was written to suit this particular RamDisk, and the Ramdisk could only be used to a maximum of 1600 sectors, or around 400K, it looked like the only way out was to have a "bash" at the thing to get it working.

Now writing a program for some other application is one thing but tackling a DSR is a different kettle of fish, the whole exersize taught this black duck so much more about the machine, and in particular just how one cant always avoid the GPL associated with it, it also bought to light just how valauble a Debugging program is and taught me another one of those little tricks in writting Assembly programs. When a problem is encountered I found that a simple solution is to write a short routine about the problem, load it to memory then go in with the Debugger and work through the routine, changing values, inspecting registers and buffers etc untill the routine worked fine, it could be then included into the main source file knowing that it would work ok, this removed the problem of finding the routine in a full program file, and it didnt matter if the test routine was loaded into a readily accessable area of memory, the "guts" of the routine was what was really wanted and it made things much easier than continually changing code in a full file, assembling it, loading it, then finding out that a bug still existed.

After getting the Quest up and running, there still remained the need for a Program to load the DSR, and manage the RamDisk. To kick off there, a lot of help was had by having Tony's source files for Q/D, not that the finally assembled version for the program to do the job resembles anything like the original source files written by Tony, (nor as elegant) but it sure made life a lot easier by pointing out just what was needed, I must say here that I was impressed by Tony's code that checks each CRU address allowed for the Quest then tests the memory if something is found and determines if WHAT is found is a Quest or not, very neat. If this code were incorporated into the Configure Program for the Horizon RamDisk to sort out if the Ramdisk found was an Horizon or not then the two could very happily live together in the PE Box, although not having an Horizon Ramdisk I dont know how they get on when installed together anyway.

One piece of code in Tony's file that has been retained unchanged is the inverting routine to test all of the Ramdisks memory, take it from personal experience, if a test is done and a fault is reported then a fault DOES exist, even if subsequent tests show OK, the main problem encountered was bad connections between the IC and its socket, despite having top quality sockets fitted, usually by removing the IC then reinserting it fixed the problem.

One other area that problems were encountered with is the heatsink on the voltage regulator, maybe its because I have such a HUGE variation in supply voltage from the old diesel generator that I find that the fan has to be on all the time, maybe someday I will get around to fitting a much larger heat sink. During winter most "in towners" would probably not have any problems but during summer when the ambient temperature jumps the problem could occur, this problem can be noted by the fact that different characters

than should be, start appearing in a text or source file when loaded from the Quest, also if during assembly of a file on the Quest the Assembler starts to show many errors that you know are not there then its time to "cool it".
I can only say that after using only floppies for years the transformation is dramatic, if anyone has a PE box then mate!,its never too late to pawn your wife/husbands jewelery or if you are not married flog your girl/boyfriends car and get one!.

ABOUT MODULES.

A bug in a program can remain hidden, sometimes for a very long time, and will only manifest itself occasionally, sometimes these bugs can be serious and sometimes of only nuisance value this can also apply to bugs in hardware.

One such problem came to light only recently in the DataBiotics Superspace II 32K cartridge, this brand of Cartridge has been around now for some time, the particular unit that "turned up" in Black Hole county had the copyright date 1986, three years old. Now the owner of that cartridge had problems with it just after purchasing it and after returns to the manufacturer for repairs, each time it was returned the report was all ok!, each time after being returned it still malfunctioned, what made matters worse for the owner was the fact that other owners of the same brand of cartridge all said that theirs gave no trouble at all, convincing the owner that he had a "dud". Not so! it was how the cartridge was being used, and using it that way produced the "bug".
The problem initially was that the battery was very quickly run down untill the Ram would no longer hold its data, one attempt made during the earlier warranty returns at preventing this was to insert a resistor in the battery line, to try to reduce the current drawn, very shoddy!, it only made matters worse pulling the Ram chip down earlier as the available voltage was also reduced, testing of the Module showed that it was drawing approx 10 micro amps, about 10 times more than it should have been, after a few checks a faulty monolith capacitor was found that had shorted internally and became a resistor across the battery, after replacement the meter showed a current of near or below 1 micro amp. HA!, fixed!.

NO WAY MATE!, after about three weeks the battery was flat again, a bit better but still no good, the meter was put on the Module again but it was only drawing a very low current, what the hell?, after some more testing the real "bug" came to light.
The Superspace II can be used with a variety of memory IC's, including Eproms, and to provide for this is some little pins and things that push on them to configure the Module to what type of main memory IC is going to be used, also there are some resistors and other bits to keep the data in a static Ram if fitted, one of the resistors goes to the pin known as "Chip select", this keeps any stored data "alive" when the Module is removed from the console power, there is also another resistor connected from the battery through those configure pins and things to another pin called "write enable", this is to pull it "high", this is where the problem arises, while the Module is NOT in the console all is ok, when the Module is in the console with the power ON, all is ok, BUT when the Module is LEFT inserted in the console and the power is turned OFF the IC's in the console allow the resistor to become a load across the battery and the load draws approx 100 micro amps, over 100 times more than it should.

Simple HUH!, some owners use the Module then remove it and stick it in a drawer or on a shelf, all OK!, but one (maybe more) owner/s loads it with programs then leaves it in the console for weeks on end and the battery goes flat, with a consumption of 100 plus micro amps on the small lithium battery it lasts only about three weeks.

I havn't bothered to inspect the console circuits to see what IC is involved with the write enable line that would do this but with a 32K memory IC installed this "pullup" resistor is unnessary. I have included a drawing of where to disconnect this resistor if anyone is having similiar problems, only

one end of the resistor needs disconnecting and leave it free so that if the owner ever wishes to try some other memory device in the module all that is required is to solder the free end back in.

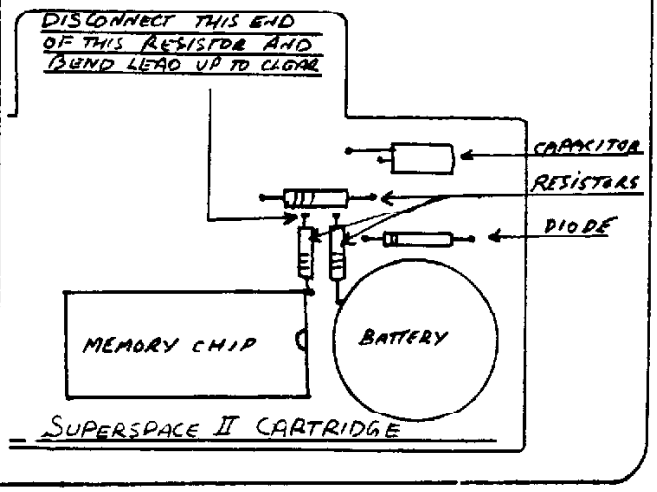
OTHER THINGS.

One of the articles published in this newsletter, by Neil Quigg, that has seemingly gone un-noticed is the modification to the PAD Ram that gives an additional 256 Bytes at >8200, it is a pity that not all consoles were so endowed, the extra ram is extremly usefull for many applications, some of the utilities and programs that I use have been modified to use that address for saving valuable data, it is used as a safe "rollout" area, other programs and bits and pieces can be splattered all over any of the rest of memory, then if the data is wanted it is there, also it provides a much larger Fast Ram area without upsetting FAC, GPL workspace etc, one example of its use is that it is a very handy place to stash Funnelwebs mailbox if it is going to be trashed then before exit of whatever is being done then the mailbox can be restored.

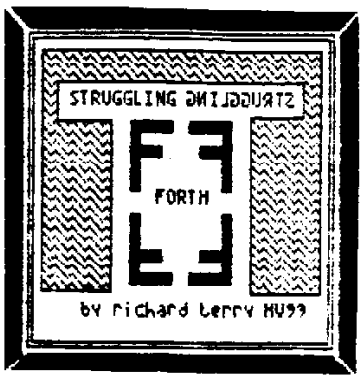
Curiosity about the question thrown out by Tony about the keycode returned by FCTN V prompted a few checks to see what turned up in the consoles here, three consoles from an early black model up to a late 1983 version would probably be a good indication covering the full production time of the 4A, all consoles returned >4F as the byte, also as he stated it is blank on the screen, regardless of the character set used, also it would perform no other functions such as delete in the TI system, but the Epson printer uses character 127 as a control to delete the previous character in the buffer unless it has already been printed and it performs that operation with FCTN V, has anyone else found out any more?

I read with interest in Joes BITS and PIECES column (May) about the reported near miss between Earth and an Asteroid. Actually the report he must have read was very misleading, the real details of the object, to date, has been kept a secret. There has been a lot of activity in the upper reaches of NSW concerning this celestial object, large teams of scientists have been very busy in the region, one reason because of the clarity of the atmosphere, but the other reason is that there is the required technical equipement here and they can carry out their work without too much publicity, these teams of scientists are headed by a relatively little known scientist by the name of Halley, NOT to be confused with THE famous Halley, this gentleman not only has degrees in Astronomy but is also an expert in the field of Astrophysics, tests so far indicate that the object is indeed an unusual Comet, and it seems that it has (this is why the whole operation is being kept secret) a very high concentration of organic matter, the organic matter is not unlike a black flexible substance, which is why the object does not display a tail, like other comets, nor glow in the heavens, the other unusual thing about the object is that it is spinning very rapidly, so much so that it is barely holding itself together, this high spin rate is deforming the object into a shape that can be only described as something like a dohnut.
Observations are almost completed now as the object distance is getting increasingly greater, very shortly details about the findings on the object will be published in all the scientific journals and possibly the newspapers, the name given to the object has been subject to much discussion but the choice is fairly unanimous and will almost certainly be, Halley's Grommet.

Ron Kleinschafer.



```
SCR #128
  0  ----------------------------------------------------------
  1  -                                                        -
  2  -                                                        -
  3  -                                                        -
  4  -   STRUGGLING FORTH                                     -
  5  -                                                        -
  6  -   HV99ERS  JULY/89  ARTICLE                           -
  7  -                                                        -
  8  -   WILL THE REAL FORTH PROGRAMMERS                      -
  9  -   PLEASE STAND UP                                     -
 10  -                                                        -
 11  -                                                        -
 12  -                                                        -
 13  -                                                        -
 14  -                                                        -
 15  ----------------------------------------------------------
```



It's amazing what one forgets in a few months. The last time I fired up the trusty TI to program in Forth was last september! I'd forgotten just how brilliant the language was, and embarrassingly the stack orders of seemingly simple forth words! My excuse for the latter is that memory fades the older one gets, my nagging worry is of early onset Alhzeimers!

WILL THE REAL FORTH PROGRAMMERS PLEASE STAND UP.

Joe and I often debate whether there are any real Forth programmers out there. Sure, its a language that attracts its devotees, many of whom tend to be byte scrooges, and love producing code tighter than the proverbial... In any group, its great to have them, if only for us lesser mortals to be able to filch their good code, and help our slower brains see whats immediately obvious to them. However I'm often dubious if they do much to advance the cause of what learning a language and what programming is all about - ie writing useful programs. Some of the most widely used, most visually pleasing, and useful programs I know, are, according to buffs of the respective languages they are written in - be it Ext Basic or Assembler - terribly badly written - ie the code is sloppy, illogical and non-compact. Yet they perform well and are great to use. Whereas I can appreciate the comments of those who have the knowledge to recognise a badly written program from a well written one, I wonder how much it advances the cause of program writing when these same critics will not impart to us their secrets of good programming. How many people get put off by the assumption that because they cannot write good code, as a corollary they are unable to write a program. Fortunately we are lucky in our group to have Tony who is not only an expert in assembler and extended basic, able to criticize AND willing to write detailed articles imparting all he knows to whoever is willing to listen.

There must be people out there who do write good, error corrected Forth programs. I'll extend an open invitation to any-one, Australian, Japanese, American or Martian, who will write us a series on how to do it! We may even post them a Koala as a reward - plus a few gum leaves for its on-going maintenance!

When I used to look back on my extended basic programming efforts I used to shudder with embarrassment at the poorly organised convoluted illogical code GOing TO anywhere like the serpents on the good old game of snakes and ladders. The point I missed was that it worked. I was able to produce a very useful financial data base program, a great 12 column general ledger, which I still use today, as well as numerous other support programs. All "BAD" code.

Also I don't think I'm a "good" Forth programmer. I don't understand electronics, I still don't know what GROM,GPL,CRU etc really mean, I suspect I still have a basic lack of understanding of fundamental memory architecture, bits, bytes, and words etc. My code is not-tight, often redundant, wordy and would I'm sure be looked on by

scorn by the Forth pundits. I know I don't understand the inner workings of the dictionary, or the return stack etc. I often have to look things up in the book because I'm not sure how the word works, or because of my short term memory loss! I guess this is where the Forth whizzkids could really help us, by writing detailed articles showing exactly how to **practically** use these various difficult to understand features of Forth, which are probably the most powerful. I'm 100% sure I've not even begun to scratch the surface of the power of the language.

However, one things for sure - I do write programs, useful programs and ones which work, albeit sometimes with a few bugs which I agree to live with. Bugs which may be there because I've missed the concept of something and don't quite know whats causing them.

Most of the Forth programs which grace our shores have been of appalling quality, most emanating from the good old US of A. I will quickly add - the program concept of many are excellent, however virtually without exception there has been an almost total lack of good graphic presentation, no error checking (allowing the user to bomb out or lock up), and poor documentation.

So I've decided to write a series of articles on writing Forth programs for the idiot programmer like me, - a threat I gave last year and never carried out due to pressure of work - now I'm just going to have to make the effort.

WRITING PROGRAMS IN FORTH.

For anyone seriously contemplating writing in Forth, I believe there are several steps one needs to take.

THE MOST IMPORTANT PREREQUISITE IS TO ABANDON COMPLETELY ANY ATTEMPTS TO TRANSLATE YOUR NON-FORTH PROGRAMS INTO FORTH, AND TO TRY AND PRETEND YOU REALLY DON'T KNOW ANY OTHER COMPUTER LANGUAGE AT ALL. In fact probably the best Forth programmers have never programmed at all prior to Forth. As an example I decided the other day to re-write my 12 column general ledger in Forth. I quickly designed the screens and got out the extended basic listings to see what file format I'd used etc. I got a bee in my bonnet for some unknown reason, of making the program compatible with access from extended basic. Within a short time I had myself in a total mess, wishing I'd never started, cursing those wretched floating point routines and just about giving up. I then put it away and next day got up and thought "what in the hell do I want to make it compatible for anyway". I wrote the guts of the program in the next few hours. Anyway.....

Firstly, you need to get together a library of Forth books. I've got heaps, and one gets lots of different ideas from different books. Try book sales at the university or the larger book stores - they tend to throw them out at ridiculous prices because no one except us nuts want them. The books don't have to be TI-Forth, they don't even have to be Fig-Forth or Forth79/83 standards, because the beauty of Forth is that if you find something you like in one language version it can be quickly transposed to another. For example I took the TI-Forth coding of a design program I wrote, and wrote it in MASTER-FORTH on a friends IBM with only minor modifications to the code -mainly to do with the way IBM handles its screen addressing etc. I left it on his hard disk, and the next thing I know I was being pestered by an engineer from a company in Sydney wanting me to sell the program to him for incorporation in an industrial application he was setting up. I was terribly flattered, but refused. I would have felt almost fraudulent selling a modified Ti-editor!!

There are two books I would regard as highly desirable by any serious programmer. The first,by Leo Brodie is essential reading: [No - its not "starting Forth" -when I read that, and had reached the end I still asked "but what good is Forth"], but THINKING FORTH -A LANGUAGE AND PHILOSOPHY FOR SOLVING PROBLEMS. It may seem strange calling a computer language a philosophy , but it rubs off on your life in a strange way, and ultimately effects the way you approach lifes problems (no-you don't start doing everything

backwards!!!). The second is the FORTH ENCYCLOPEDIA, a ginormous help to me ; my copy (I can hear Joe saying "Whose copy?") gives Forth 83 and FORTH 79 equivalents. This volume is of tremendous use in understanding how a particular Forth word is constructed and how it works. It gives the words pronunciation, an example of the form in which it is used, the return and parameter stack values at entry and exit from the routine, and a total breakdown of the word components, each with a full description of the stack effects of each component and a comment on what each component achieves.

Secondly, one needs to take to the keyboard and INTERACT with the language and computer - its heaps quicker than following excercises in a book, and as in life, the only way to learn. Sure, if you're a total novice, get out your STARTING FORTH and try the exercises as a means of familiarising yourself. Try to do things, experiment. (For a longer discussion on this refer to some of my previous articles in HW99ers). Get familiar with postfix notation, duping and swapping etc. Once these are mastered you need to develop routines for keyboard input, screen display, file access, and number handling. Most books can help with this, and I wouldn't neglect the original TI-FORTH manual -some of its sections are actually quite ok, and the Index describing each word is pretty good as well. Once your familiar with all these your as ready as you will ever be to write a program. Don't worry at this stage with all the niceties such as error handling, autobooting etc, they will struggle into place later.

Thirdly have a goal - decide what program you would like to write, even if it seems too ambitious. As you will discover, unlike most languages, you will be able to write your program in easily debuggable segments, each of which will be stand-alone modules. Once your goal is decided, write it - even if your early code seems inefficient and rambling -FORTH will force logic on you if you are illogical yourself

Fourthly peruse you clubs stock of international user group magazines and any Forth programs you stumble onto - even for the IBM. Snaffle the good code (making a note of the author for future recognition), and start a library of Forth subroutines on a seperate disk, even if they don't seem useful at the time you find them.

Lastly, BE PROUD OF YOUR EFFORT, don't let anyone knock it for being BAD code!

All this will get you off to a good start. If you're ultra logical like some of the Forth buffs you will also sit down and design your program according to the accepted wisdom, which is:design from the top down, and write from the bottom up. However I can recall reading, that not even the originator of Forth himself - Charles Moore sticks to this format!

AN ASIDE ON NAMING FORTH WORDS

I Think one of the things that puts most people of Forth is its so called lack of readability. I'm not surprised that they are put off. Whenever I pick up most TI user magazines and look at the Forth code, I'm constantly amazed at the lack of ledgability. With the exception of segments of code using a lot of low level words such as DUP,SWAP etc, much of what we write can be written virtually in English. This is a point Leo Brodie goes to such pains to point out. If my failing memory serves me correctly I can recall him writing somewhere "Be proud to type EMPTY-BUFFERS"!!!!! In what other language can you type exactly what you want the program to do, interactively at the keyboard, and get immediate feedback. This is where I get so annoyed at the BYTE SCROOGES of our fraternity who may write : MT .... ; instead of EMPTY etc, so that they end up with a screen of code illedgible to both themselves and their readers. As an example of GOOD LEDGIBLE CODE here is some out of my new LEDGER, in the section to do with displaying days of the ledger:

```
: SAME-DAY  0 DISPLAY-DAY ;
: LAST-DAY -1 DISPLAY-DAY ;
: NEXT-DAY  1 DISPLAY-DAY ;
```

Immensely powerful stuff!!!

A PROGRAMMING ENVIRONMENT.

I decided a couple of years ago I needed some development tools to take the tedium out of writing. After all, as you will find once you get into Forth, more so than in many other languages, once you've sorted out the basic routines to handle screen display, screen input, array handling, error checking etc, they remain constant in virtually all programs. The ACTUAL amount of code needed to generate an entire program is often not more than a dozen or so screens.

I like to get immediate feedback from my writing. The buzz I get out of writing is not just seeing the program do something, nor the intellectual satisfaction of programming, but actually seeing it on the screen - ie what it looks like. One of the slow processes in designing a program is the tedium of the screen design. I found that if I could actually see in front of me what I wanted the end product to be, it became easier to then write the unseen code to link the whole thing together. I hit upon the idea of writing a text-mode text and graphics editor, to take the tedium out of screen design. One would then immediately get visual feedback as to if something "looked right". The screen of design could then be saved back to disk, and when needed in the program, be loaded in. I wrote the original editor as an adaptation of the original in Ti-Forth and it worked fine. I decided that rather than load the screen displays as I needed them, I'd load and store them in VDP at the start of the program and recall as needed.

I then decided to write a series of programs I grouped under the loose title of Forth Programming Utilities the editor; a print utility; a program to construct custom-built pattern descripter tables; a Forth-style disk and screen manager with simple non-Forth utilites such as initialisation, catalog etc; programs to transfer Forth to non-Forth files ie DF128/DV80, and archive Forth in file form.

Having done that, SuperForth came along, giving me access to much more memory. I decided to combine the lot - an ended up with one program I called Forth Utilities which is a combination of all the above and more - a configuration program, a facility to prepare documentation for programs, to transfer Forth to Ti-writer and back etc.

The next thing I regard as being essential to program development is a RAM-DISK. The feature that gives you total control of your machine, namely being able to write to anywhere in memory, also is capable of causing the most catastrophe in the form of lock up. In the early days of programming I was continually locking up and switching off and rebooting. Now-adays thats rare but it still happens. I've allocated my first Horizon as DSK3, filled it with a sizable dummy DF128 sys-screens file, leaving enough room for Forth6 and a couple of assembler files, then from Forth copied over the boot screen and binary codes of the programs I use as my developing environment. The only thing needed in Drive 1 is a disk with a one-liner on screen 3 telling the system to jump to the appropriate screen on the ram-disk to continue booting. Using a simple menu which comes up about 1/4 sec after booting I get a 4 option menu allowing me to run my utilities program, program in Forth, boot a Forth disk of quit. I can then switch back and forth between environments quickly. You can do the same thing with Ti-Forth by changing the byte on screen 8 displaying which drive to jump to after loading Forth.

PROGRAM STAGES

When I sit down to program now, I have a loosely fixed series of procedures.

1.Program concept
2.Window/Screen design and display
3.Writing program peculiar code
4.Testing and debugging
5.Constructing a new program disk
6.Writing program documentation
7 Beta testing (give to Al Laurence)
8.Constructing a distribution disk
9.Reaping the fairware rewards -they never come!

The finished program has the following structure, irregardless of the nature of the application. Your next program disk will differ only in the small amount of application specific code. Everything else is in its logical pre-developed place:

Reference scrns	-Your program overview comment
Windows screens	-program specific easy to write
Documentation screens	-program specific easy to write
Non program Dependant core code screens	-easy to compile store and load as needed
Relatively non program depend ant code scrns	-easy to keep and modify for each new program
Application specific code	-the difficult bit! original Thought!

In the ensuing months, I will present the features of my Forth utilities program, and write articles covering the above listed features of program development, as well as other more basic things for the uninitiated, such as setting up a binary work disk, and perhaps basic simple data accept type routines. As a project we will write the design editor program in TI-FORTH, the most useful utility, to allow genuine programmers to get on with being REAL FORTH PROGRAMMERS.

ADDRESS FOR CORRESPONDENCE

R.TERRY
141 DUDLEY RD,
WHITEBRIDGE
NSW 2290

AUSTRALIA.

You grow up the day you have the first real laugh at yourself

Ethel Barrymore

BITS & PIECES

from

JOE WRIGHT

What amazing times we are living in. Events that we are witnessing through out the world will be looked back on by our future generations. They will study them as history and try to relive through historical records what we are part of. Judgements will be made about us based on these events, just as we judge those before us. Will the legacy we leave these future generations be freedom and liberty. Through these turbulent times will freedom and liberty flourish; I hope so.

Liberty, when it begins to take root, is a plant of rapid growth.

Washington.

INTO THE UNKNOWN.
****
Alan Franks and I had a bit of a chin wagg recently about consoles that had crashed. The number of failed consoles within the group has steadily increased. Something had to be done. One of our members was completely out of action and several others were down to one console.

Prior to this, we both, had been expecting "some one else" to do something. We made a fateful decision. We would try to repair consoles ourselves.

Selecting a console at random from the pile the Geof Trott console tester revealed that ROM A was faulty. This created it's own problem, spare ROM's ???.
This was solved by Alan pulling the ROM's out of a good console, the good ROM was plugged into the freshly soldered socket in the dead console and it fired up beautifully. Now remove the ROM again off to a place where I can lay my hands on an EPROM burner. Pick up 15 2532's from Neil Quigg and blow 8 ROM A and 7 ROM B. Place the 2532 into the socket again with pin 21 free and then tie pin 21 to pin 24. Again the console fired up and passed all tests. One down, this job looks dead easy!!!

Gaining in confidence I pass the word that I was looking for a dead console to fix and possibly use for spares. Don Dorrington found one near where he lives, $10:00 and it was mine. Same proceedure, try the console tester, dead as a door post, blast! Power supply is ok oh no!. What now, buy a logic probe and a pulser (didn't need the pulser yet but I had a gut feeling). Sure enough the CPU is stone dead. Pass the console onto Alan Franks who is without peer at removing chips from circuit boards. Al installed the 64 pin socket and slipped a new CPU in for me, more bad news, the console still did not run mmmmh. Using the logic probe we quickly found that pin 62 on the CPU was held. This is system ready.

This created a new problem in itself, holding the CPU for a long period will overheat and damage the CPU and I was going to need a long time to test this one! No quick fix this time, that gut feeling was becoming a pain. Get logical! why not remove the CPU and power the console up? Then test all the gates in that part of the circuit. That is what was done, all the gates tested ok and finally the problem was isolated to the 74ls74 flip-flop. Again Al's talent at desoldering was put to good use, and a socket fitted. The new chip was fitted and the console now runs perfectly.

That is the shortened version of that story. The moral is that there are some failed consoles which will just simply be not worth taking the time and effort to fix.

Alan has repaired another 4 or 5 consoles with faulty ROM's or faulty cpu's. We did get another curly one out of the way a couple of days ago, this console had ROM A crashed, one of the system RAM's failed and the 9901 crashed.

From what we have been able determine several of the chips used in the T.I. 99/4A will be dear to purchase if available at all. Notable is the TMS9904, in small quantities it would cost around A$30.00. Then the system RAM MC6810 after a quick ring around local shops is not "off the shelf" either.

Luckily our mate from Melbourne Peter Gleed has been able to help out with TMS 9900 and TMS 9901'S. One local supplier is asking A$55.00 for a TMS 9900, I'm sure he thinks it is like one of our endangered species and he has the last one on earth! Stuffed up his profit motive a bit when he was told that a second hand going console sells for $40-$50 and you get a transformer, modulator and a heap of books with it. Oh well! Hope he gets a flat battery on his next trip to the snow fields in his BMW.

Finally it gets down to how much time one is prepared to put into repairing any one console. We all only have a certain amount of free time available for our hobby.

MORE LETTERS.

Must say that some people do take me at my word. In the May newsletter I asked for people to write to me. Well I have since received a letter from Bob Carmany and one from Charles Bagley. Both more than welcome and both answered promptly. The invitation still holds, write! your letters are most welcome. Charles in his letter suggested that we include a glossary of computer terms on a lift out in the newsletter. Well that seemed to me to be a really good idea. The lift out in this month's newsletter is the first of these. The source of the glossary is a book many T.I'ers would have in their possesion.

JOB'S NeXT

A recent news release in a Sydney newspaper tells that the charismatic and since deposed founder of Apple, Steven Jobs has sold 16% of his NeXT company to Canon of Japan, in return for a cool US$100 million cash injection. Canon will have the rights to sell the NeXT machines throughout Asia, but this does not include Australia, according to trade reports. Canon also makes the erasable optical disc drive that is a vital part of the NeXT machine. Jobs originally intended the black, cube-shaped NeXT as a device for university students and researchers, but has since done a deal with US distributor Businessland to sell the machine to corporate users.

DISTURBING NEWS.
****

I can well imagine being soundly criticised for this next BIT. But! I feel it must be told. Also in a recent newspaper was a report on dwindling whale numbers based on estimates for the Southern Hemisphere.

TYPE OF WHALE	PRE-WHALING POPULATION	CURRENT POPULAT.	No. SEEN IN 6 Yrs
BLUE	250000	200-1100	22
FIN	100000	2000	27
SPERM	1000000	10000	179
HUMPBACK	100000	4000	87
	250000		

Last year 49,000 dolphins and porpoises were caught and slaughtered of Japan for consumption in Japan. It is considered a delicacy by those who can afford it.
Of the 49,000 (highest yet), 2000 were accidently caught in nets the other 47,000 were capture and kill specifically for human consumption.
In 1986 13,000 Dall's Porpoises were caught from the cold waters of the northern Pacific off Japan. By 1988 that slaughter had increased to 41,500.

The point of the article was that as depressing as those porpoise kill numbers are, the reason Japanese authorities released the figures is even more depressing.

The porpoise kill numbers were released in an attempt to force the International Whaling Commission to increase the kill quoters for whales. They claim doing this will decrease the kill rate for porpoises. Whale meat sells for $150.00 a kilogram in Tokyo and is even more sort after that porpoise meat.

The market is so lucrative that Japan, Iceland and Norway, the only countries still killing whale, are applying intense pressure for a return to the gruesome days of commercial whaling. Last year Japan took 241 Minke whales from Antartic waters, for "scientific purposes". Nearly all of these ended up as sashimi in Tokyo's most exclusive restraunts. Japan wanted to take 300 but Greenpeace protesters restricted the kill to 241. Norway took 29 Minke and Iceland took 68. All these ended up on Japanese tables.

An Australian delegate to the IWC made the following comment; "It's an operation driven by high prices for a luxury item in a wealthy country," he said. "If it stopped tomorrow only a few hundred people would be directly effected."

At a time when most developed countries are at least starting to contemplate their enviromental responsibilities, sadly Japan continues its delinquent antics.

Of course the drift nets are another story again.

Well! sorry if you don't agree with me writing this in a computer newsletter. The next thing to come off this word processor are letters to each of the Embassies of the countries mentioned. Why don't you write them also?

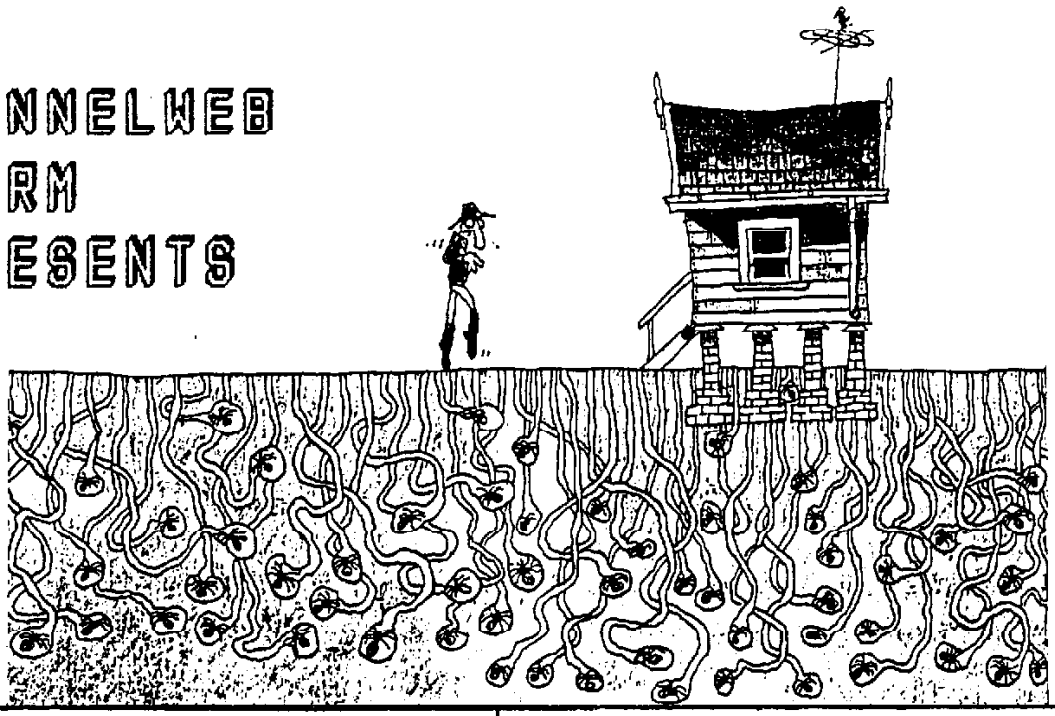
THATS ALL!

Best wishes
Joe Wright.

CALENDAR 89

As mentioned in the president's report, a number of events are planned for members this year.
Your support would certainly benefit not only your club but I am sure, yourself.
The 'Black Hole' will receive a visit from our club members over the weekend of 9th and 10th of September.(Ron has promised to take us on an extensive tour of the metropolis of GRAWIN)
Tim Watkins has indicated that he would like to conduct another of his famous SPECIAL INTEREST days in October so stay tuned for the date.
Melbourne TIUG is holding it's TI FAIR on 2nd October, so some of us may be keen to attend and provide support for that activity.
25th November looks like being a whoopee Christmas dinner, possible at the IMPERIAL PALACE Chinese restaurant at Beresfield where we had a great time a couple of years ago.
A Christmas party/day for the kids and not so young for that matter is planned for Sunday 2nd December.
If you have any ideas about these activities, or others which may be of interest to our group, please let us know.

FUNNELWEB FARM PRESENTS



EXTENDED BASIC TUTORIALS REVISITED

by

TONY McGOVERN

Part 2

III. SUBPROGRAM PARAMETER LISTS

In the last chapter we saw how subprograms fitted into the overall workings of Extended Basic. In this chapter we are going to go into the details of writing subprograms. Most of the fiddly detail here concerns the construction of the parameter lists attached to CALL and SUB statements, and some of the little traps you can fall into.

Any information can be transmitted from the CALLing program to the CALLed subprogram via the parameter list, and anything not transmitted this way remains private for each program, with the exception of the DATA pool which is equally accessible to all. If something is mentioned in the parameter list then it is a two-way channel unless special precautions, provided for in XB, are taken. In this case the CALLing program can inform the subprogram of the value of a variable or entry in the parameter list, but not allow the CALLed program to change the value of the variable as it exists in the CALLing program. Arrays however, numeric or string, can't be protected from the follies of subprograms once their existence has been made known to the subprogram through the parameter list.

Let's for starters take a very simple but useful example, where a program needs to invoke a delay at various points. Now some BASICs (and TI LOGO) have a built-in function called WAIT. XB doesn't have this command (though a cynic might suggest that GPL gives it to you all the time whether you want it or not) so you have to program it. It can be done by a couple of CALL SOUNDs or with a FOR-NEXT loop. Let's use an empty loop to generate the delay, about 4 milliseconds each time around the loop, and place the loop in a subprogram.

```
230 CALL DELAY(200)
670 CALL DELAY(200/D)
990 CALL DELAY(T)
3000 SUB DELAY(A):: FOR I=1
TO A :: NEXT I ::SUBEND
```

This is easier to follow when editing your program then using a

GOSUB, and you would need to enter the subroutine in every subprogram since GOSUBbing or GOTOing out of a subprogram is verboten. Also it's less messy than writing the delay loop every time. The example shows several different CALLs to DELAY. The first supplies a number, and when DELAY is CALLed, the corresponding variable in the SUB list, A, is set to 200. This is a particular example of the kind of CALL from line 670 where the expression 200/D is first evaluated before being passed to DELAY to be assigned to A. Variable D might for instance represent the level of difficulty in a game. The CALL from line 990 invokes a numeric variable T, and A in the subprogram is set to the value of T in the CALLing program at the time when the CALL is executed.

Nothing untoward happens to T in this example, as the DELAY subprogram does nothing to change A. Now it may not matter in this instance if T did not retain its value after the subprogram CALL. Suppose instead the delay was to be called out in seconds. Then a subprogram on the same lines DELAYSEC might go

```
230 CALL DELAYSEC(2)
990 CALL DELAYSEC(T)
4000 SUB DELAYSEC(A):: A=A*2
50
4010 FOR I= 1 TO A :: NEXT I
:: SUBEND
```

Now after DELAYSEC has been executed with the CALL from 990, T will have value 250 times its value before the CALL. This won't be a bother if you don't use T again for its previous value. If the CALLing program specifies a numeric constant as in line 230, or a numeric expression, the change in A in the subprogram has no effect on the main program. Suppose you can't tolerate T being changed in line 990 (and this kind of thing can be a source of program bugs). You will find that XB allows for forcing T to be treated as though it were an expression, thus isolating T from alteration by the subprogram, if T is enclosed in brackets in the CALL (not SUB) list. Suppose DELAYSEC is also called from line

```
970 CALL DELAYSEC((T))
```

If this CALL in line 970 is followed by the CALL from line 990, T not having been altered in the meanwhile, the same delay will be obtained, but if the order of CALLs were reversed the second delay would be ~250 times the first. In the language of XB this is known as "passing by value" as distinct from "passing by reference". This can only be done for single variables or particular array elements, which behave like simple variables in CALL lists. Whole arrays cannot be passed by value, but only by reference. Expressions and constants can only be passed by value, and its hard to see what else could be done with them. In the example as written, a different variable name was used in the SUB, but if you remember the little experiment in the last chapter you'll see that it wouldn't make any difference if T had been used in the SUB list instead of A.

Now let's complicate things a little by flashing up a message on the bottom line of the screen during the delay interval.

```
200 CALL MESSAGE(300," YOUR
TURN NOW")
270 CALL MESSAGE(T,A$)
3000 SUB MESSAGE(A,A$):: DIS
PLAY AT(24,1):A$
3010 FOR I=1 TO A :: NEXT I
:: DISPLAY AT(24,1):""
3020 SUBEND
```

The SUB parameter list now contains a numeric variable and a string variable in that order. Any CALL to this subprogram must supply a numeric value or numeric variable reference, and a string value or string variable reference, in precisely the same order as they occur in the SUB list. In the little program segment above, line 200 passes constants by value and line 270 passes variable references. There is no reason why one cannot be by value and one by reference if so desired.

This process can be extended to any number of entries in the parameter list, provided the corresponding entries in the SUB and CALL lists match up entry by entry, numeric for numeric, string for string. The XB manual does not say so explicitly, but it appears that there is no

limit apart from the usual line length problems, on the number of entries in the list. This is the only apparent difference between the parameter list in XB subprograms and the argument lists for CALL LINK("xxxxxx", , ...) to machine code routines in XB, and Minimemory and E/A Basics.

One little freedom associated with built-in subprograms is not available with user defined subprograms. Some built-ins, such as CALL SPRITE permit a variable number of items in the CALLing list. Parameter lists in user defined subprograms must match exactly the list established by the SUB list or
error "INCORRECT ARGUMENT LIST in
allow whole arrays, numeric or
g, to be passed to a
subprogram. Complete arrays may be passed by reference only. Individual array elements may be used as if they were simple variables and may be protected from alteration by bracketing in the CALL list. An array is indicated in the parameter list by the presence of brackets around the array index positions. Only the presence of each index need be indicated as in A(). MATCH(,,) indicates a three-dimensional array MATCH previously dimensioned as such, explicitly or implicitly. Don't leave spaces in
list. If the subprogram needs
know the dimensions of the array these must be passed separately (or as predetermined elements of the array). TI Basics are weaker than some others in that they do not permit implicit operations on an array as a whole, a very annoying deficiency.

Arrays may be DIMensioned within subprograms. This will introduce a new array name to the program, and an array or variable name from the SUB parameter list can't be used or an error message will result. In the following code the main program passes, among other things, an array SC to subprogram BOARD (perhaps a scoreboard writing routine in a game).

```
100 DIM SC(2,5) :: ....
450 CALL BOARD(P,A$(),SC(,))
4000 SUB BOARD(P,A$(),S(,))
:: DIM AY(5):: ......... ::
CALL REF(P,AY(),S(,)))
4080 SUBEND
5000 SUB REF(V,A(),B(,))::
```

BOARD generates internally an array AY() which is passed to another subprogram REF (maybe this resolves ties) along with SC(,), which BOARD knows as S(,), and REF in its turn as B(,) -- the same name could have used in all places. There is however no way that the main program or any subprogram whose chain of CALLs doesn't come from BOARD can know about the array AY(). This would hold equally well for any variable or array, string or numeric, first defined within BOARD and whose value has not been communicated back to the CALLing program via some other variable mentioned in BOARD's parameter list.

By following this line of reasoning you can check out the conclusion that there is no way for a subprogram whose chain of CALLs does not come through BOARD to know about array AY(). The only way around this is for AY() to be DIMensioned in the main program (even if this is its only appearance there) and the message passed down all necessary CALL-SUB chains.

This idea of DIMensioning an array only within a subprogram is particularly useful if the array is to READ its values from DATA statements and to be used in the subprogram. This could be done again from any other subprogram needing the same data, without having to pass its name up and down CALL-SUB chains. Remember that DATA statements act as a common pool from which all subprograms can READ. If the array values are the results of computations then these values must be passed through the CALL parameter lists.

For completeness note that although the XB manual has nothing to say about it, IMAGE statements for formatting PRINT output are accessible from any part of a program in the same way as DATA statements and not confined to the subprograms in which they occur as are DEF entries.

It is not necessary to have any parameters in the list at all. Subprograms used this way can be very helpful in breaking up a long program into more manageable hunks for ease of editing. We shall also see in later chapters that there can be other benefits as well.

One more XB statement for subprograms remains, the SUBEXIT. This is not strictly necessary as it is always possible to write SUBEND on a separate line and to GOTO that line if a condition calling for an abrupt exit is satisfied. Like a lot of the little luxuries of life however, it is very nice to have and makes programs much easier to read and edit. It does not replace SUBEND which is a signal to the XB pre-scan to mark the end of a subprogram. SUBEXIT merely provides a gracious and obvious exit from a subprogram (awkward in some Pascals for instance). The next chapter will demonstrate typical examples of its use.

IV. USEFUL SUBPROGRAM EXAMPLES

In the previous chapter we used as an example a DELAY subprogram which could, with a little refinement, be used to substitute for the WAIT command available in some other languages. You can extend this idea to build up for yourself a library of handy-dandy subprograms which you can use in programs to provide your own extension of the collection of subprograms that XB offers. The MERGE facility with disk based systems makes this particularly easy. See Jim Peterson's Tigercub Tips for many further examples.

For our first example let's take one of the more frustrating things that TI did in choosing the set of built-in subprograms. If you have Minimemory or E/A you know that the system keyscan routine, SCAN, built into the console ROM returns keyboard and joystick information simultaneously, while XB forces you to make separate subprogram CALLs, KEY and JOYST, to dig it out. Since these GPL routines are slow it is difficult to write a fast paced game in XB that treats keyboard and joysticks on an equal footing as is done by many cartridge games. On the other hand in games where planning and not arcade reaction is of the essence there is no reason why the player(s) should be forced to make a once-and-for-all choice and not be able to use either at any stage of the game.

The subprogrammers approach to this problem, once it realised that it can be done (and we have seen commercial XB games where the writers haven't) is to write the game using joysticks, but replacing JOYST by a user defined sub-program JOY which returns the same values as JOYST even when keys are used.

The first step in telling whether keys or joysticks are being used is to check the keys, and if none have been pressed then to check the joysticks. If a key has been pressed then its return, K, has to be processed so that the direction pads embedded in the keyboard split-scan return the corresponding JOYST value. A subprogram along the lines of the one used in TXB does just this.

```
900 SUB JOY(PL,X,Y):: CALL
KEY(PL,K,ST):: IF ST=0 THEN
CALL JOYST(PL,X,Y):: SUBEXIT
910 X=4*((K =4 OR K=2 OR K=1
5)-(K=6 OR K=3 OR K=14))
920 Y=4*((K=15 OR K=14 OR K=
0)-(K=4 OR K=5 OR K=6))
930 SUBEND
```

PL is the player (left or right joystick or side of the split keyboard) number and is unaltered by the procedure. The simple-minded approach for converting K to (X,Y) values by using the XB logic operators (one of the more annoying omissions from console Basic) seems to work as well as any. The subprogram as written checks the keys first but balances this out by putting the processing load on the key return.

This is as good a time as any to sharpen your own skills by working out alternative versions of this procedure, and also by writing one for mocking up a substitute CALL KEY routine to return direction pad values even if a joystick is used.

FUNNEL WEB (Male). Approximately 25mm in body length. Black to very dark brown in colour. Identification points: Spur on second front legs, long slender spinnerets on rear of abdomen, shiny surface on the head and front section of the body. One of the world's most deadly spiders.

FINDING OUT ABOUT THE FOUNDLING 4A

by A. Member
and
Foster Parent

FOREWARNING: The following diatribe is a protracted account of a foster-parent's attempt to communicate with his newly-adopted 4A. It has been forwarded to the editor in answer to his desperate plea for fill material for the newsletter. Having made it into print (obviously he must have been desperate), it may encourage a reader or two to contribute articles to the newsletter. It need not be a daunting prospect. As you can see from this example, it does not have to be technically brilliant or show outstanding literary merit. There always seems to be regular input of interesting and informative material in any case.

Some years back I acquired my first 4A. I had no previous experience of computers, so it was a case of reading the Users Reference Guide (URG) and Beginners Basic (BB) in an attempt to assimilate the unfamiliar words of wisdom contained therein.

After setting up as per URG, I found the advice on page II-2 and started off with BB. In seemingly no time at all I was HI THERE!ing and 2 and 2 are 4 - oops - 2+2=4ing away. I worked through BB and then it was back to URG and the pages (and pages) of commands, statements, functions and program examples which, I might add, seem designed to ensure that the reader is not over-excited.

I eventually battled through to the Application Programs where again I was advised (page III-14) to start with BB. Ignoring this advice, I tried out the 9 programs. This was when I realised that there was more to life than HI THEREing, even with a B flat accompaniment bordered by flashing asterisks in two different colours. I decided there and then that I wanted to PROGRAM!

At the time I couldn't locate other books or magazines about the 4A, so it was back to BB and URG again. It was heavy going but I persisted, and now, years later, I continue with my attempts at programming - and enjoy every minute of it.

In retrospect, the grind of trying to learn TI Basic with such limited (and uninspiring) resources had some advantages as I had to work by trial and error and experimentation. It was also detrimental in as much as my approach to programming became modelled on BB and URG examples.

Eventually I got hold of other books on the 4A and magazines with some 4A articles. Then I found out about User Groups and joined one, now two. The resultant additional material on the 4A was invaluable in expanding my primitive programming horizons.

However, the bad habits I'd learned from BB and URG stayed with me until I ventured into more ambitious and complex programming efforts, when it became obvious that the resultant untidy messes were mainly due to the haphazard use of variables and some unnecessary program lines. As I began to clean up the programs, it was obvious that I had to also clean up my (bad) programming habits.

So I decided to try a BB (and URG) revisited exercise, not in nostalgia, but to look through them again from an if-only-I-had-known angle. As I did so I spotted some don'ts and I recalled some of the 'discoveries' I had made. I thought that I'd put some of it down on paper and this is the result.

If, having made it this far, you do decide to stay with it, dig out your copy of BB as I'll be using a few page references - although it's not essential to following the 'story'.

On page 8 you're shown how to type

```
 PRINT "THIS IS A MESSAGE"
```
(space)

You don't have to include the space between PRINT and ".

On page 13 you are ceremoniously introduced to the LET statement to start making life difficult for the novice. Although you are informed on page 16 that it is optional and you don't have to use it, most of

the subsequent programming examples continue to use it. Probably it was included in TI Basic because it was used in some other Basics. To a raw novice it was very confusing, and I continued to use it in case I did something 'wrong'. Just one BB example of how to add unnecessary clutter to a program.

More confusion for the beginner on page 23. At this stage, naturally, I was following (and believing) all that was printed. So it took me, a slow learner, quite a while to work out what I was doing wrongly when I typed CALL VCHAR(1,10,86,50) and ended up with 47 Vs on my screen when the BB diagram clearly showed only 31.

Moving along to Simple Programming - on page 26 there's another program line, this time with a line number, using PRINT.

```
      (space)
   10 PRINT "ARE YOU READY"
```

As previously noted, you don't need a space between PRINT and ". Nor do you need a space between the line number and PRINT, or, for that matter, between a line number and any statement or command. eg

```
10A=2
20B=3
30PRINT"A+B=";A+B
```

goes into memory - and comes out of memory - just fine.

The computer (clever little 4A) can sort it out internally so that when you LIST you'll get

```
10 A=2
20 B=3
30 PRINT "A+B=";A+B
```

which, admittedly, looks neater. My programming efforts, however, take a lot of two-finger tapping effort and I can't see any point in tapping in unnecessary spaces.

If you have a penchant for spaces, include them by all means. Unless they cause an obvious conflict, eg. A B=2, the computer doesn't care and will simply remove them.

```
10CALL HCHAR (12, 3 ,42, 28 )
```

works OK and will LIST back as

```
10 CALL HCHAR(12,3,42,28)
```

In retrospect I wonder why the line number introduction in BB missed the chance to introduce NUM as it isn't too difficult for the novice. And, of course, it automatically provides a space after the line number.

On to page 30 where we come across the END statement, and the reader is informed that it is also optional. However, as it's 'conventional', it is used in the example shown - then it continues to be used in most of the subsequent examples, except some with GOTO loops.

Of course the implication is that, as END is optional, you don't have to use it - do you? Something to remember when you try to omit it from the program on page 101. The cases where END/STOP are necessary, not optional, are not mentioned.

Back to page 34 of BB and the INPUT statement. Once again, you don't need the spaces -

```
10INPUT"KILOGRAMS?";K is fine
```

Where quotation marks are used after a statement (DATA is another one), no space is required.

Next we come to the GO TO statement on page 38. You are informed (page 39) that GO TO can be typed GOTO - another unnecessary space eliminated. True to form, however, apart from a few lapses, most subsequent examples persist with GO TO.

Before leaving Chapter 2, the reader is advised that he/she is well on the way to becoming an experienced computer programmer. I recalled thinking, when I first read this, that it would be nice to have one - an experienced computer that is. I thought at the time that rather than well on the way, I was still in the starting blocks. In hindsight I was wrong even then - I was just about to wander onto the track.

In Chapter 3 the reader is given an introduction to FOR-NEXTs and their associated control variables. Here was the source of one of my worst programming habits - the haphazard use of variables. eg on page 51

```
FOR A=1 TO 1000
```

is used to start a delay loop. For the same (delay) loop on page 52, B is used as the variable and on page 53, A is used as an input variable.

Such indiscriminate use of any old letter for a variable became my way (monkey do) also for quite a long time. I now realise that it makes long programs difficult to decipher. I've now begun to standardise my use of variables.

In my earlier programming attempts I often used variable 'words' or their recognisable abbreviations. I never found them helpful at any stage and quickly dropped the practice. Also they require more typing effort.

I've noticed that I and J are used almost universally for the standard repetition loops. I seem to recall reading somewhere that I and J were specially tagged for loop variables in an earlier language and their use in Basic is a hangover from then. I dispensed with I a long time ago as it was all too easily confused with the number 1 in the programs that I wrote (scribbled). I now use J, K and L for such loops along with M and N for subscripting into arrays.

Also, after trying R and C for a time, I've now settled on X and Y for row and column in the various CHARs - the decider being that they are conjoint.

Likewise for CALL KEY, where the variables required are Return and Status, it's R and S - conjoint.

I use A and B (A$ and B$) for my INPUTs, extending to AA, BB etc. if required. For counters I now use C and also D. Like all improving programmers I don't use D for delay loops any more, just CALL SOUND - thanks CW.

I still haven't finalised preferred letters for all variable usage. T seems to have the nod for temporary (as in sorts) and Z will probably get the nod as the final item for data reads, etc.

E, F, G and H are still unassigned, but I'm sure that I can find a small niche somewhere for G$.

P and Q hold possibilities if some minding is required. U is fairly obvious as another counting variable for late night use. For arranging things the opposite way around we have V and W.

Before leaving variables, I seem to recall reading that they use fewer bytes than numbers although they may be slightly slower in execution. So instead of

```
FOR C=1 TO 5    why not use

FOR C=C TO 4    as C will be Ø if
```

it has not been used otherwise.
Or you can select any permissable variable listed in URG on page II-11 (Roman numerals in a publication on the LXXXXIX/IVA (or is it XCIX/IVA?)

The back slash doesn't seem to get much work, so why not

```
FOR C=\ TO 4    or, if you want
```

to keep C at Ø you can use

```
FOR \=C TO 4
```

Continuing on with BB and MORE POWER PROGRAMMING - the last paragraph on page 55 advises that a program line, in general, can be up to four screen lines (112 characters) in length, the exception being the DATA statement. That 'in general' is nicely unspecific.

I found out during an editing effort in my early days that it's no great problem filling 6 screen lines (more or less) in Basic. A few times I have come across brief articles on the subject. A detailed expose has probably appeared somewhere in print.

However, I haven't come across the 11½ screen lines per line number - 321 characters - case mentioned in print anywhere.

Anyhow, this seems like as good a time as any to call a BREAK (better late than never). To be CONTINUEd when the fingers (both) recover and the grey cells (both of them also) recharge.

WHY TI?

BY *Bob Carmany*

What is it about the TI-99/4A that sustains the interest in the computer some five years after all production ceased? After all, all of its contemporaries have long since passed into oblivion. The IBM PC Jr has long since gone along with the Commodore VIC-20 and the TIMEX-SINCLAIR. What is so special about the TI? It can't be just "nuts and bolts" because, quite frankly, it isn't THAT good!

One of the very best things that TI did when it introduced the 4A was to create a vast network of Users' Groups throughout the United States and the rest of the world. When TI was "in the game", there was a Users' Group newsletter from TI and there were releases of software and programming tips --- something the other computers lacked. Then, one dark day, TI abruptly "pulled the plug". But rather than give up, the Users' Groups pulled the known TI world together. New software began to appear --- intermittently at first -- but later in a more steady stream. Not the trash that usually appears in the dying throes of a computer's life but "class" commercial grade software. Funwriter, NEATLIST, ARCHIVER 2.0, DM1000, and the RAG software packages started the process. These were followed by FUNNELWEB, QED 4.4, ARCHIVER 3.02, TELCO, and many others too numerous to mention. The one common thread throughout it all is that these software packages and the quantum leaps in TI hardware technology were created by Users' Group members!

The Hunter Valley UG can be justifiably proud that a disproportionate share of the class software entries are the fruits of its own UG members. One of the reasons that I chose to join a couple of years ago is because I felt that Hunter Valley was the best User Group, without qualification, in the entire world! The fact is, I will continue to mail in my annual dues as long as there is at least one other member willing to do so --I am THAT dedicated to trying to make Hunter Valley the best there is!

So it was with a sinking heart that I read through the meager offering that appeared in April. As the editor so aptly put it, it wasn't to save postage! Admittedly, there were articles missing from a couple of the "regulars" but, believe me, it is tough to come up with something every month. That brings me to the crux of the matter -- we need material for the newsletter! A newsletter editor's dream is to have so much material that you have to decide what to put in this month and what to save for the next issue. If you have done anything with your TI except use it for a door stop or paperweight, you have the expertise to write an article for YOUR newsletter. It doesn't have to have technical expertise of a Funnelweb Farm article or the wit of "Black Hole". If everyone in the UG would take an hour or so and write a single article for the newsletter, there would be enough material to last throughout the year. Almost anything TI-related will do. Experiences with software packages you have tried (good or bad), tips on playing games, programming problems, or musings about how you use your TI will do. Let's stuff the newsletter editor's mailbox so full of letters and articles that the postman will get a hernia carrying the lot up there.
If you can't type, write it out and someone will take care of getting it in print. Between now and the end of September give it a real "go".

On a closing note, my 'Random Bytes' column is meant to be exactly that --- a column of random topics. I will research and write about anything that ANY member of the UG would like to see. If you would like to see a column on a particular topic, no matter what it is, write to me I'll see what I can do!

Remember, this is YOUR Users' Group and YOUR newsletter so pull your finger out and write an article today!!!

THE COMMITTEE

BY Peter Smith

Don't let the knocking of your legs distract your next-door neighbour as you read this survey of your new committee's individual details.

John Paton.. Vice president.
Has a full(hic!) system with 2 ram-disks, an 80 col Digit card and an Amiga 1081 monitor + a Maestro supermodem and a Microbee printer.
He is a plant operator with a local firm (Civi-com) and his favourite program is FUNNEL-WEB.
His wife Sue is expecting their first child in February 1990 and John's ambition is simply to see tomorrow.
John has served as software librarian for a short time and has a great interest in communications using the TI.
He is also a keen exploiter of the graphics capability of the TI and is learning ASSEMBLER by writing programs and hacking others.
John can be called at 326014 right in the heart of the HUB.

DON Dorrington is a committee member and this year is his first effort at officially being of service to the club.
He has a Peter Schubert MINI-PE system with a 512k r/d. He is having a little trouble with his d/drive set up at the moment, however he is a well trained electrician and should have no problems sorting things out in the near future.
Don can be reached on 531228

ALLAN (JOE) WRIGHT is a person we should all be familiar with. He was a founding member of this club and has served as president for a number of years and is still a very vital guiding sage in it's current affairs.
As well as being a stirrer of the greatest degree and one heck of a nice bloke, he actually works at BHP as a shift supervisor.
His 2 slimline drives and horizon r/d have learnt all about forth and assembler, and what they don't know about family trees isn't worth knowing.
Joe's favourite program at the moment is BATTLE CHESS.
His lovely wife is Beverly and his phone is 468120.

Bob McClure is a stalwart supporter of the TI and is always able to be counted on to support any function which this club undertakes.
He is renown for his fine soldering techniques which must surely be of assistance to h im as a design draughtsman.
His full TI system is well used to running Bob's favourite program , FUNNELWRITER.
Bob has been interested in maintaining the club's hardware and certainly has done a good job getting various drives repaired and helping maintain systems.
He can be contacted on 437431.

ALAN FRANKS is a boilermaker with the hands of a surgeon.
He has served as soft-ware librarian and as Console repairer extraordinaire with Joe Wright.
He is at present our very active hardware co-ordinator (that means when you want something see Al) and has as his favourite program TI CHESS.
I don't know what Al intends to do if he wins lotto (His secret ambition) however I am sure he would use some of the money to expand his full system even further.
Al can be contacted on 459170 at Mark's Point.
The remainder of the committee for 1989/90 have yet to reveal their secrets, however watch for details in the next newsletter about Brian Woods (Past editor extraordinaire and new secretary), Tim Watkins (Director of most pleasurable car-rallies), Stewart and Jim Bradley (our software librarian devils), Noel Cavanagh (Our new treasurer and father for the third time), Peter Smith (that's me) and Ken Lynch (An Ambulance man).

GLOSSARY OF TERMS.
****

ABSOLUTE ADDRESS.
****
1. An address that is permanently assigned by the machine designer to a storage location.
2. A pattern of characters that identifies a unique location without further modification.
3. Synonymous with machine address, specific address.

ACCESS TIME.
****
The time interval between the request for information and the instant this information is available.

ACCUMLUATOR.
****
A device which stores a number and which, on reciept of another number, adds the two and stores the sum.

ADDRESS.
****
An expression, usually numerical, which designates a specific location in a storage or memory device.

ADDRESS FORMAT.
****
1. The arrangement of the address parts of an instruction. The expression "plus-one" is frequently used to indicate that one of the addresses specifies the location of the next instruction to be executed, such as one-plus-one, two-plus-one, three-plus-one, four-plus-one.
2. The arrangement of parts of a single address, such as those required for identifying channel, mode, track, etc on a disk system.

ADDRESS REGISTER.
****
A register in which an address is stored.

ALGOL.
******
ALGOrithmic Language. A language primarily used to express computer programmes by algorithms.

ALGORITHM.
****
A term used by mathematicians to describe a set of proceedures by which a given result is obtained.

ALPHANUMERIC.
****
Pertaining to a character set that contains letters, digits, and usually other characters such as punctuation marks.

ALU.
****
Arithmetic Logic Unit, a computational sub system which performs the mathematical operations of a digital system.

ANALOG.
****
Electric analog information is information represented by a variable property of electricity, such as voltage, current, amplitude of waves or pulses, or frequentcy of waves or pulses. Analog circuitry, also called "linear" circuitry, is circuitry that varies certain properties of electric continuously and smothly over a certain range, rather than switching sudden between certain level.

AND
***
A logic operator having the property that if P is a statement, Q is a statement, r is a statement..., then the AND of p, P,Q,R, is true if all statements are true, false if any statement is false, P AND Q is often represented by P*Q or PQ, Synonymous with logical multiply.

ARITHMETIC SHIFT.
****
1. A shift that does not affect the sign position.
2. A shift that is equivalent to the multiplication of a number by a positi or negative intergral power of the radix.

ASCII.
******
(AMERICAN NATIONAL STANDARD CODE for INFORMATION INTERCHANGE, 1968). The standard code, using a coded character set consisting of 7-bit coded charact (8 bits including parity check), used for information interchange among data processing systems, communication systems, and associated equipment. The ASC set consists of control characters and graphics characters. Synonymous with USASCII.

ASSEMBLE.
****
To prepare a machine language programme from a symbolic language programme by substituting absolute operation codes for symbolic operation codes and absolu or relocatable addresses for symbolic addresses.

ASSEMBLER.
****
A computer programme that assembles.

ASYNCHRONOUS DEVICE.
****
A device in which the speed of operation is not related to any frequency in t system to which it is connected.

											C	CON-SOLE
DEL	INS	ERASE	CLEAR	BEGIN	PROC'D	AID	REDO	BACK		QUIT	F	
											C	MICRO-SURGEON
ULTRA-SONIC	ANTI-BIOTICS	ASPIRIN	EKG	HEART	SLOW	FAST					F	
SINGLE EXECUTE	CONT EXECUTE	PRGM/STAT SCREEN	INTRUPS ON/OFF	SOUND OFF				SAVE OPTIONS		EXIT	C	MG EX-PLORER
MEMORY WINDOW	MEMORY SIZE	DISASSEM SIZE	PAGE UP ↑	SEARCH	PAGE DOWN ↓	OPTIONS	REGIST'S	EDIT FIELD/MEM	BASIC BIAS ON/OFF	ASCII/ HEX	F	
											C	GRA PHID
SLOWER	FASTER	DRAW	ERASE	NOHELP	ZOOM	COLORS	LINES	CIRCLES	COPY	MENU	F	
DATA	TEXT		ABORT PRINTER	NEXT WINDOW		PRINTER ON/OFF	OUTPUT ON/OFF	ESCAPE	SCREEN CAPTURE	SCROLL SPEED	D	DISK ASS EM-BLER
DELETE	INSERT	ERASE	ABORT		NEXT PAGE			BACK PAGE			In	
OOPS!	RE-FORMAT	SCREEN COLOR	NEXT PARAGRAPH	DUPE LINE	LAST PARAGRAPH	WORD TAB	NEW PARAGRAPH	NEW PAGE	WORD WRAP		C	TI WRITER
DEL CHAR	INS CHAR	DEL LINE	ROLL DOWN ↓	NEXT WINDOW →	ROLL UP ↑	TAB	INS LINE	COMMAND/ ESCAPE	LINE #'S	QUIT	F	
HOME	TAB	NEXT UNL CELL	FORWARD CHAR	FORWARD WORD	CHANGE WINDOW	REL/ABS REF				CANCEL	C	MULTI-PLAN
LOWER RIGHT			BACK CHAR	BACK WORD		HELP	RECALC	BACK SPACE	DELETE FORWARD		F	
										QUIT	C	GRAN KRACKER
WINDOW C/G/V	MOVE	FILL	PAGE ↑	SEARCH	PAGE ↓	COLOR	DUMP	BACK	BIAS ON/OFF	ASCII/ HEX	F	
											C	TI LOGO II
DELETE		ERASE	CLEAR	BEGIN	PROCEED	PAUSE		BACK		QUIT	F	
A PRINTER COM	B BLOCK	C CENTRE	H TAB	J REQ SPACE	L NEW LINE	N NEW PAGE	M MID LINE	P PARAGRPH	← MOVE → COPY - DELETE	(UNDER-LINE)	C	COMP
DELETE ↓ CANCEL PRT	INSERT	ERASE & HIGHLIGHT	↑ ↓	BEGIN	↑	MENU	END	EDITOR COMMANDS	V COLOR	QUIT	F	
											C	
HEX	ASCII	EXIT	← BACK	RE-START	FWD →		RE-WRITE	MENU		QUIT	F	
A → CLEAR I.	B → CLEAR C.	C → INPUT D.	D → DRAW	E → ALPHA N.	F → FILL	H → H OR V	I → INVERT	K → K-LINE	L → LINE	M → MIRROR	C	TI ARTIST
N → SWAP	O → CIRCLE	P → POINT	Q → DISC	R → RAYS	S → STORE	V → FRAME	X → BOX	Z → ZOOM	FCTN . → PEN	FCTN ; → SPEED	F	
DEL FILE	* TE II *										C	4A - TALK
OPEN PBUFR	CLOSE PBUFR	H/F DUP'X	OPEN FILE	SAVE FLE CLEAR FILE	KEYBRD	^ CAT ^	CONFIG	FILE/TRAN	DIAL	QUIT	F	
Ctrl . = ESC Ctrl C = ETX Ctrl I = TAB											C	P-CODE CARD
DEL	INS		BREAK	SUSPEND	ALPHA LOCK	SCREEN LEFT	SCREEN RIGHT	LINE DELETE			F	

BAUD	TOGGLE SPOOLER	MODEM PARITY	MODEM PORT	PRINTER PARITY	PRINTER PORT	PRINTER BAUD			TOGGLE 40/80 COLUMN	QUIT	C	FAST-TERM
XMIT DEL >7f	XMIT Ctrl-T >14	XMIT Ctrl-N >0e	ABORT TRANSFER	WINDOW →	XMIT Ctrl-E >05	TEXT COLOR	SCREEN COLOR		FREEZE/ WINDOW BACK	QUIT	F	
MODEM BAUD RATE	SPOOLER ON/OFF	MODEM PARITY	MODEM PORT	PRINTER PARITY	PRINTER PORT	PRINTER BAUD RATE			TOGGLE		C	FAST-TERN
N FILE 'SEND	B DUMP TO DISK	Y CLEAR LOG	CANCEL	W/DOW	K-TIMER	FORE GROUND	BACK GROUND	LOG ON/OF	FREEZE ON/OF	QUIT	F	
MODEM BAUD	SPOOL TOGGLE	MODEM PARITY	MODEM PORT	PRINTER PARITY	PRINTER PORT	PRINTER BAUD			40/80 TOGGLE	QUIT	C	FAST-TERN
DELETE	^T	^N	BREAK	WINDOW →	^E	FORE COLOR	BACK COLOR		PAGE TOGGLE	QUIT	F	
SPEAK	OUTPUT	CANCEL	TRANS	WRAP	CASE	PAGE			EXIT		C	TE II
										QUIT	F	
SPEAK	OUTPUT	CANCEL	TRANS-FER	WRAP. TOGGLE	CASE TOGGLE	PAGE			EXIT		C	TE II
^G → BELL	^H → BACKSPACE	^J → LINEFEED	^L → FORM FEED	^. → ESCAPE	^/ → BREAK	^G → X-ON	^S → X-OFF		FCN V → DELETE	QUIT	F	
RESTART		CANCEL	LOG ON	LOG OFF	TRANS-FER						C	TE 1200
										QUIT	F	
											C	ADVANCED DIAGS.
DELETE	INSERT	ERASE	ROLL↑	BEGIN	ROLL↓	AID	REDO	BACK	DUMP	ASCII/HEX	F	
DELETE CHAR	INSERT CHAR	DELETE LINE	ROLL UP ↑	NEXT→ SCREEN	ROLL DOWN ↓	TAB	INSERT LINE	ESCAPE		QUIT	C	ED/ AS
DELETE	INSERT	ERASE	CLEAR	BEGIN	PROCEED	AID	REDO	BACK		QUIT	F	
											C	DM-1000
DELETE	INSERT	CON-FIGURE	HALT I/O	MENU	EXECUTE	PRINT	RE-ENTER SHIFT KR	SUB MENU		QUIT	F	
							INSERT LINE				C	TI FORTH
DELETE CHAR	INSERT CHAR	ERASE LINE	FORWARD SCR	TAB →	BACK SCR	BUFFER IN	BUFFER OUT	ESCAPE		QUIT	F	
							INSERT BLANK LINE				C	FORTH
DELETE	INSERT	ERASE	NEXT SCREEN	NEXT WINDOW	LAST SCREEN	ERASE TO END OF LINE	COPY LINE	EXIT EDITOR			F	
							BLANK LINE		V TAB →		C	FORTH
DEL	INS	MOVE LINES	ROLL↓	WINDOW ←28→	ROLL↑	TO PAD	INS LINES PAD		V TAB ←		F	
											C	
											F	

Compiled from numerous newsletters and other sources from around the world. Any other contributions gratefully received by . . .

GEOFF SHIPTON, 8 TEESDALE CRESCENT, PLYMPTON PARK 5038, SOUTH AUSTRALIA, AUSTRALIA

LAMINATED COPIES ARE AVAILABLE FROM THE ABOVE ADDRESS FOR $1(EACH