HUNTER VALLEY 99'ERS NEWS



TI 99/4A

HOME COMPUTER NEWSLETTER



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.

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

INSTRUMENTS.

HU99 is a non profit group of TI99/4A Computer users, not affiliated in any way with TEXAS INSTRUMENTS.

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 Al Lawrence.

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 HV99 NEWS 9 THIRLEMERE PDE., TARRO 2322

General address for ALL other club

THE SECRETARY HV99 USER GROUP 6 ARCOT CLOSE, TARRO 2322

YOUR CONNITTEE 1986

PRESIDENT Allen Wright 77 Andrew Rd., VALENTINE 2280 Ph. 468120

VICE PRESIDENT Jim Grimmond 31 Jarrett St., TORONTO 2283 Ph. 595751

SECRETARY Albert Anderson 6 Arcot Close, TARRO 2322 Ph. 662602

TREASURER Drian Rutherford 9 Bombala St., REDHEAD 2301 Ph. 498184

SOFTWARE LIBRARIAN Alan Lawrence 35 Bayview St., WARNERS BAY 2282 Ph. 486509

PUBLICATIONS LIBRARIAN Paul Mulvaney 26 Marmong St., MARMONG POINT 2284 Ph. 583623 W

OK 9ts

R

C

n

me

m

B

ti

ti

Go

Gr

TH

me

Sm

be

ma

c1

SU

be

ma

re

If

Ai

By

17

SU

U.

EDITOR Brian Woods 9 Thirlmere Pde., TARRO 2322 Ph. 662307

COMMITTEE Bob MacClure 75 Deborah St., KOTARA SOUTH 2288 Ph. 437431

Garry Jones 53 Janet St., JESMOND 2299 Ph. 573744

Tony McGovern 215 Grinsell St., KOTARA 2285 Ph. 523162

Peter Coxon 25 Reserve Rd., WANGI WANGI 2267 Ph. 751930

EDITOR

Alexandrian Comment Comment of the C

It is with great pleasure that is up and running, and will be available for sale at the Melbourne TI-FAIR at the weekend. Copies of this update (the final?) will be of the TI will be assured. available from the Librarian in the Tony and Will McGovern for their dedication to the TI.

******************************** With time running short to produce own version of the Ron Kleinschafer P.E. Box, Ron graciously agreed to lend us his to take to Melbourne, and for this we sincerly thank him. As you know, Ron does not live close enough to come to our meetings, but this has not prevented him from inputting more to our group than many who live much closer.

By the way Ron, we all hope you find that Black Hole and recover your truck soon.

Good news to all the fans of Millers Graphics' "The Smart Programmer". That production and Bytemasters general consensus was to run another "Super 99 Monthly" have recently merged to renew publication of "The Smart Programmer", the first issue being dated June. A copy of this magazine has been received by the and is available from the club Publications Librarian. subscription in the clubs name has been sent, so future issues will be made available to you as they are received.

If you wish to subscribe yourself send #US2Ø for Surface mail or #US32 Airmail to: Bytemaster Computer Services, 171 Mustang Street, SULPHUR, LA 70663 U.S.A.

Like the Editor magazines around the world, I want your input to the clubs newsletter. It doesn't have to be pages long, anything will do. It doesn't have to be an in-depth technical treatise; short. interesting articles on topics that appeal to you are ideal. If something is of interest to you, you can be sure that there are others equally interested.

Please make the effort to write something, for without your support the ideal of complete local content we announce that V3.3 FUNNELWRITER will be lost, to the detriment of the whole User Group fraternity. If all magazines were content to copy other peoples work, the final death

SPECIAL INTEREST GROUPS

SATURDAY WORKSHOP

A successful afternoon workshop was held at Warners Bay High School on Saturday May 31st when a group of met to excharge rate interesting H.V.99'ers ideas,demonstrate programmes and generally discuss problems.

Programmes demonstrated included :-Word Processing-tape version(Brian Rutherford), Bar Graphs Mulvaney) GraphX (Peter Smith) T.I.Artist (Peter Coxon) and an Adventure game corner (Rodney G.)

Many thanks to these and all the others who contributed so willingly to a very rewarding afternoon. The workshop in about two afternoon months time.How about the 2nd.August? Please let me know if you can participate and what your problems are-T.I. of course!!

It is an afternoon for all members to share and develop their uses for the T.I.99

WANTED: A copy of FINDEX or similar journal indexing programme.

Tim Watkins, Co-ordinator.

FLOWCHART SYMBOLS

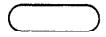
Last months Introduction to Flowcharts showed a good practical use of the flowchart but failed to explain the symbols. The symbols used in this article are based on the Australian Standard AS 2645-1983.



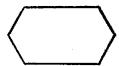
Think Flowchart

Flowcharts are used to break programming problem into logical elements and subdivisions, they divide a large problem into smaller more manageable segments. The time spent on designing a flowchart can be worthwhile as it forces you to think about the program and allows you to write it in a logical instead of haphazard manner. Even though the 99 has an excellent RESequenceing routine it is good practice to structure programs.

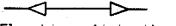
Flowcharts are made up of symbols which describe various processes in the program. The symbols have a short description of their function written within their boundaries.



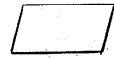
The first symbol we come across is the Terminal. This represents a terminal point in a program, eg start, stop, delay or interrupt.



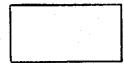
The Initialisation Hexagon defines the setup process at the start of a program, eg the dimensioning of arrays and redefining of characters.



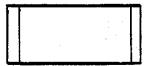
Flow Lines link the symbols with arrows showing the direction of the flow of information.



The Input/Output Parallelogram allows the operator to input information from the keyboard or the program can access information from storage media. If the symbol is used for Output information can be sent to the screen, a printer or to storage media eg disk or tape. The function is usually written in the symbol.



The Process Rectangle represents any kind of processing function, eg the process of executing a defined operation resulting in a change in value, form or location of information.



The Predefined Process Rectangle represents a named process consisting of one or more operations or program steps that are specified elsewhere, eg Sub-Routines.



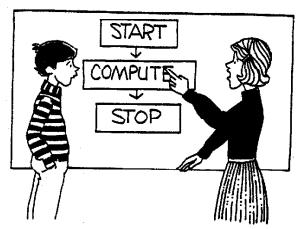
The Decision Diamond indicates a decision or switching type operation that determines which alternative path is to be followed.



The Connector shows an exit to, or an entry from another part of the flowchart. They are pused to eliminate a maze of flow lines that would otherwise overlap. A common identifier, usually an alphabetic character is placed within the connectors to allow the matching of entry and exit points.

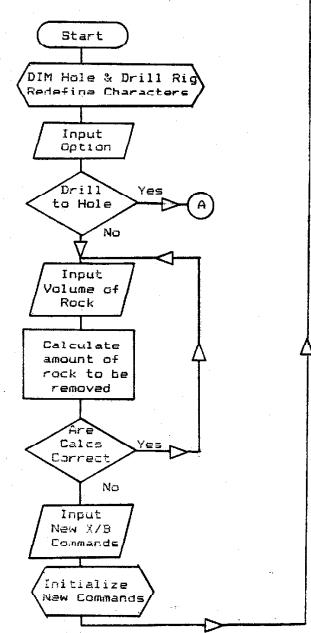


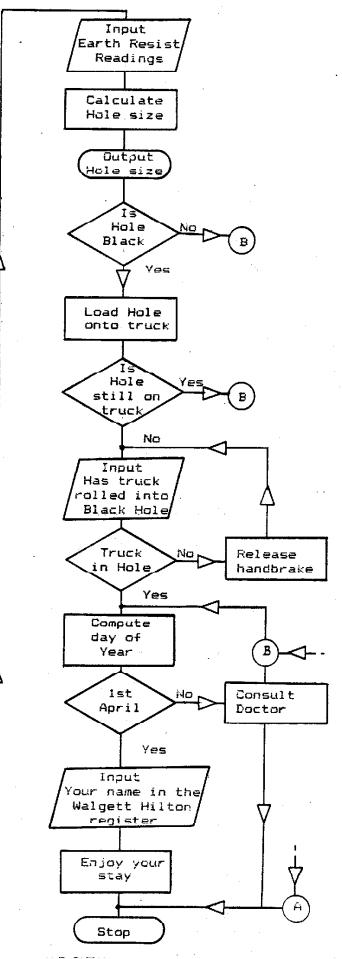
The Annotation symbol allows the addition of descriptive comments to be added for clarification.



"I was hoping for a little more detail"

I have used Ron Kleinschafer's Digression article from last months newsletter as the basis for this sample of a flowchart





ENTOMOLOGY CORNER NUMBER 6

and the same of th

BY TONY PEGOVERN, H. V. 99ERS

The time has come once more to put fingers to keyboard and do something for the HV99 Newsletter. I am of the opinion that monthly issue is just too frequent for a small group to support and will inevitably lead to too much dilution of local content. That can be fixed by having more local contributions too. Anyway for this edition, apart from the general gossip and progress reports we will have a review of Orphan Chronicles by Ron Albright and first impressions of the Myarc disk controller card.

. On to the general gossip first. Funlwriter Vn 3.3 is progressing very satisfactorily and I am even writing some new document files to go with it. The group members going to Melbourne for TI-Fair will after all have something to take with them, unless of course Will uncovers some previously unsuspected fatal bug a day before they leave. Un 3.1 of DM1000 was received from Bob Boone in Ottawa and the updated MGR1/2 have been interfaced to F'Wr Vn 3.2 to make an interim update until Vn 3.3 is released. Clint Pulley has sent the latest release of his small-c compiler with additional graphics, floating point ... libraries contributed from other Canadian sources. The Province of Ontario is a real fount of too grade 'fairware' material. Edgar Dohmann from JSC/UG near Houston tells me that a new version of Superbug II is on the way soon aiso.

Clint has made REL2 of c99 compatible with XB loader programs, and we have taken it a step further and made it re-entrant to F'Wr with filename transfer back to the Editor. Vn 3.3 will also allow return to F'Wr from DM1000 without going through the title screen. have also closed the circle and made a version that will load from the utility option of TI-Writer. Really had to dig to find the module to test that one out. more improvements go to make up a major update, but time enough to see those when it is released. Enough maybe to mention that a disk directory is available from a press of AID.

0

i

T

T

P

ы

wi

ti

а

51

ec

CC

\$C

wi

yσ

Αp

th

ve

Ċσ

th

th

mi

ma

90

di:

im

da

dr

ha

đi

We

who

de.

ag(

The

the

any

Mya

wit

the

are

of

hel

One

tha

fas

new

swi

bus

tim

FSØ

tri

ori

car

lit

exp

Now to the Orphan Chronicles, published by Miller's Graphics. The copy reviewed was on a round robin to gather an inscribed record Thanks Dick, of its travels. thanks Rex. Before anything else let me say that it is a thoroughly absorbing read for anyone who has spent a lot of time wrestling with the orphan. The book contains a measure of prehistory, and a description of the rise and fall, but most of it is oriented to the ongoing support in programming and hardware that will keep the machine viable in the near future even There is much without TI. information on the great growth in bulletin board and areas on commercial database systems. Unfortunately in Australia we are isolated by the tyranny of telecom costs and distance from the US I would have liked to networks. see more discussion of the engineering and corporate decisions and battles that went into making the machine the flawed gem that it But then again a writeup like that might not stir the same interest in readers without an engineering background. A question implicitly posed that is not answered and probably never can be, is whether the right microprocessor. was chosen. I think it was, and that it is the TMS9900 processor that gives the machine its continuing life after death. machine would have been cheaper for TI to build, though this is confused by the transfer pricing games that went on within TI over the 9900, but I ran't help feeling that a Z80 based machine would have

no more continuing fascination than a Dick Smith VZ2ØØ. More like MSX which hasn't really turned people on either. As Ron Albright indicates, the real tragedy is that TI lost their nerve after the 99/4a and didn't continue with the 99/8. There must have been a lot of people like us who held back from buying a 99/4a expansion system while waiting for the upgrade of the whole system. The author is altogether too kind to some of the characters who floated to the surface - the pompous and vacuous editor of a magazine of very little content and the most amazing disappearing acts, a shonky software dealer masquerading as a user group, and so on. I do note with approval, as do the rest of your committee, that in the Appendix listing User Groups around the world the HV99 group is the very first mentioned.

We now have a Myarc disk controller operating in our PE box thanks to appreciation of F'Wr in the US of A. If Ron K's 3 holer mini PE box project comes good we may be able to get a second system going as well. The following discussion will be of first impressions only. The card gives double density operation of disk drives and whatever faults it may have, seeing 1438 sectors free on a disk more than makes up for them. We thought we were in hog heaven when we installed the Chinon DS drives in the PE box a few months ago, but DD is even nicer still. The comparison of interest is with the Corcomp card. I have not heard any bad news about reliability of Myarc cards, but local experience with Corcomp has not been good. the other hand reports from the US are that Corcomp have solved many of their problems, but that may not help too much if you have a lemon. One thing they do have in common that they can take advantage of faster track stepping rates of newer disk drives with a permanent switch setting. We have been so busy with F'Wr that we have'nt had time to check it out on the Chinon F502 drives yet. Will says he tried 6ms in software on the TI original Shugarts from the TI disk card and they didn't like it one little bit, but that was to be. expected.

The Myarc comes with a manager program on disk which works very well and is easy to use. Some other utilities are provided but we have not had occasion to use them One could say that the yet. original TI controller card does the job that it was designed to do to perfection even if with limited performance . The Myarc card by handling double density does more, but there are some catches apparent that make it fall short of the TI standard of engineering.

The first thing you notice is that you don't notice anything. like this much better than the way the Corcomp captures the machine on power up. As a matter of a fact it has no power up routine at all, and unlike with the TI card the drives don't spin briefly on return to the title screen. That's fine if it never needs one, but I have a suspicion that on occasion it does not in normal operation but when something has gone very, very wrong, a return to the title screen is not always enough to clear things up and the whole machine, PE box included must be powered down. That's more likely to happen around our place where Will has spent a lot of effort mastering the art of talking directly to disk controller chips, than it is in a more normal household. Another problem, which I have also not yet investigated fully, is a propensity to blow holes in the directory. This might occur if an assembly is aborted with closing the object file. the TI card this results a nuisance Dis/Fix Ø file entry which must be deleted before re-assembly to the same file name. This happened with the Myarc and I thought at first that I had lost half the source files of F'Wr. None of SD, DM2 module or DM1000 could find them, but the Myarc DM showed them after a hole in the directory list and would copy them to another disk. Presumably the Myarc directory builder doesn't stop at an imbedded null entry on sector #1 unlike most programs which work fine with the TI card. If the Myarc card really does have this masty habit, then it is essential to use the Myarc DM with it. A definite bug in the Myarc card is that it doesn't support all Status calls corectly. F'Wr Vn 3.0-2 will not assemble to

ms. her hith but We rom bts а disk es, cord se h1y as ith ā he and hine re com 0 ions ng it ike stion be, ssor br Z8Ø for hg.

ver

ling

have

a pre-existing object file if the Myarc is used, and the hangup is in the Status calls made by F'Wr before it actually opens the file. A real "Wellington lady" of a program, it checks the status of files before it will associate with them.

Whereas the Corcomo card seems to use VDP RAM in much the same way as the TI original, the Myarc after setting aside VDP buffer space for compatibility with other software then proceeds to ignore it and uses its own RAM in the card. I have not been able yet to do any comparative timing tests between Myarc and Corcomp, so I don't know whether this speeds things up any. It still has to get things into VDP where normal software expects them, but it doesn't have to do it doubly. The Myarc's use of its own RAM does lead to some incompatibility with some important software, as Myarc have not bothered to adhere to that part of the TI disk system specification that calls for the last disk name to be stored near the top of VDP The latest version (3.1) of RAM. DM1000 bombs out completely with Myarc for this reason, and the boot disk tracking feature of F'Wr Vn 3.0-2 doesn't work. The soon to be released Vn 3.3 of F'Wr takes account of of the quirks of Myarc for its own account, and loads DM1000 as well. This problem afflicts some RAMdisk DSRs too. From Bernie Elsner's last call from Perth I know that they have run into the same problem with the Horizon RAMdisk, a device which has not come to Newcastle yet, and are going to do something about it The Myarc has another quirk there. that renders DMIØØØ less than completely useful. This would appear to be a hangover from when the design was frozen, when only 16 sectors per track were written in double density (I seem to recall that the first IBM PCs came out with 320K disks compared with the later and now stadard 360K). Unfortunately 16 sectors per track is built into the Myarc DSR ROM in sub-program >11, the formatting routine. Programs such as DM1000, TI-Forth, the DM2 module which call this routine end up with with only 16 sectors per track for a total of 1280 sectors as against the now normal 1440. That's OK by disk

DSRs until you run into software that assumes 1440 sectors for DSDD such as disk sector copy programs -DM1000 for instance. Myarc get around this in a fashion that is only partially satisfactory in use by having 18 sector initializing done by the manager program from disk. I consider it reprehensible that Myarc has not by this late stage updated the card firmware or offered a replacement EPROM. is looking into the possibility of patching the Myarc DSR code so that 18 sector initialization is obtained, and maybe if that is possible we may go to the trouble of blowing a replacement EPROM. any of our interstate or overseas readers know of this having being done already, we would appreciate hearing about it, or even better a sample of a revised EPROM.

The Myarc card also offers some new CALLs available from command mode from XB or also as statements from Basic even with no module present. They were probably a good idea at the time but the coming of Funlwriter has made most of them largely obsolete, and they would be better replaced by more powerful disk related functions. The one that is useful is CALL DIR(x) to get an instant disk directory with no module insertion or program loading needed. The others are a set of CALLS, ILR, LR, LLR that replace the INIT, LOAD, LINK from E/A Basic. There are limitations, some of them surprising, for instance the E/A DEBUG which is a program which is as transparent as possible returns to a locked up computer, either from a Basic program or from command mode. Because they are invoked from Basic, even in the command mode, programs are faced with VDP register settings and table locations appropriate for Basic, so only programs designed for this will give proper screen DEBUG has this display. flexibility but most utilities for E/A, as distinct from E/A Basic. don't.

The utilities loaded by ILR are similar in function to those loaded by E/A but occupy a little more of low-mem and the REF/DEF table entries point to different locations. This is no problem for programs which REF the utilities

c d ŧ C ŧ t Þ G 9 W Ç W p, a М de G (GF 11 Ma >6 a t a 0 X٢ \mathbf{c} wh Or fr (t it CC ۵V re

un

\$C

tr

()

re

VE

s i

th

ha

but means that the ILR utilities. are not usable by programs that rely on the E/A utilities being preloaded for cold starting of memory image files. One pleasant surprize is that GPLLNK works even in a bare console. This is probably worth discussing for its tutorial value while I am at it. As you know the bare console contains no mechanism for escaping from the clutches of GPL out into the 32K RAM expansion. Programs coming in from the 32K have more of a chance to fiddle conditions so that if they call a GPL routine, they can regain control from GPL. The point of the GPLLNK routine is to trick GPL into handing back control. As discussed in the E/A manual the GPL routines that can be used are those which end in a GPL return instruction (and otherwise do something useful for you) so that GPL goes back to the stack in console scratchpad RAM to retrieve the next address it should branch to. The GPLLNK tricks GPL by first pushing the address of a suitable GPL return instruction on to the stack before handing over, so that when GPL has done the routine called it hands back to GPLLNK which then returns to the calling program after resetting the GROM address to where it was to start with. Modules such as E/A and XB do have suitable exits built which GPL uses in CALL LINK or CALL LOAD / (not XB) or from the E/A screen. GPL hands over to machine code when it encounters an XML (eXecute Machine Language) instruction (byte >ØF) followed by a byte pointing to a standard table entry. These tables are partially listed in E/A or Minimem manuals. Most of the XML instructions in GPL point to console or cartridge ROM locations which is no good for returning to a program in RAM. So how to do it from the bare console? It turns out (though TI never saw fit to mention it explicitly) that there is one console GPL routine which does hand over to RAM. This is the GPL bit reversal routine in GROM Ø which unloads some machine code into the scratchpad RAM at >8300 and then transfers control to it at >8300 ()FØ table entry) to execute before returning to GPL. Since at least 8 versions of GROM Ø have been sighted, a program cannot rely on this being at a fixed location but has to search for it. There is

ממ

se

le

or

11

of

hat

Ιf

15

9

e

a

no

ыргу

st

lеу

on

LR.

5

'ns

for

le

another possible exit point in GROM Ø, but this is accidental in the sense that data bytes happen to form a XML instruction into the scratchpad RAM at an inconvenient location. Myarc uses this console GROM return point, doing the search every time.

We should mention in passing that E/A's GPLLNK is partially realized in GPL in the module and doesn't look like our description above. Trouble is it has a bug and does not always work. When F'Wr is simulating E/A the GPLLNK always works, thanks to a mixture of brute force and low cunning.

All in all I don't think the Myarc extra CALLs (other than DIR) are particularly useful or well chosen. After all if you have the system and resources to afford a Myarc card you probably have E/A or XB anyway. Now F'Wr has rendered it completely obsolete in this regard. I think the Corcomp designers' decision to include a program file loader shows more design sense, but I'm not instantly familiar with the details. extra CALLs in both of them mean that programs which use them lose a lot of portability because neither disk card is universally used.

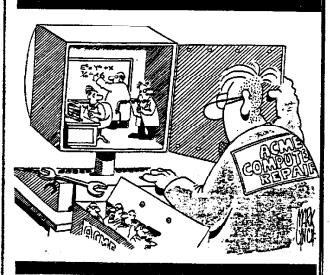
The Myarc controller uses a Western Digital 1770 controller chip and has 2K of RAM and 8K of ROM. How does that fit in the 8K DSR space allocation ? The 2K RAM is located at >5000 and the ROM is paged in two 4K halves by CRU bit addressing, each located at >4000 ->5000. Will found that paging in the second half of the ROM is done by a cute piece of code which loads workspace registers with the transfer code, then does a direct branch to the first register of the block just loaded so that it is now executing out of RAM in its own workspace, and this code pages the ROM, sets up the new branch address from a branch table in the newly paged ROM, and finally does an indirect branch to it. That illustrates why the 99/4a survives. Not only is 9900 code elegantly simple but also it allows you do the most amazing things.

After all that you might get the impression that we're less than entranced with the Myarc. Well,

that just ain't so. It's just fantastic to have all that disk capacity open up, and it just sits there and does its job. problems with not recognizing drives that you know are there. The only real difficulty is the formatting problem, and who knows maybe that's fixable too. Vn 3.3 of F'Wr has been partially re-written to accommodate the Myarc controller with no loss of function. I am debating with myself whether to alter the DMiØØØ files included on the F'Wr disk so that they bypass or refuse to format disks in double density on a Myarc controller, requiring pre-formatted disks for disk copies. Of course if you have to load the Myarc manager program to format a disk, you might as well continue to use it to do the disk cloning.

A request that I would like to pass on is from Bonnie Snyder, President of the Front Range 99er Computer Club in Colorado Springs, who is looking for programs, particularly those using speech that would be suitable for use at a school for the blind. If you have such programs I can forward them or put you in touch for direct exchanges.

Tony McGovern Funnelweb Farm May/86



ADVENTURERS CORNER

Hi!!

Here I have another fine batch Adventure hints. I would like to thank everybody who helped make it possible. A bit of news just off the line... is the word INFOCOM are considering HITCH-HIKERS TWO. Now on to other things though. have only received a response from fellow Club Members, so I have had to turn to other sources for some of the following clues so PLEASE HELP. If you can't decide which Adventure to try just give me a buzz and I will recommend one, and after using you could maybe write a review for inclusion in YOUR Magazine.

ADVENTURELAND

Does the dragon have an egg.
Before choping down a tree try to climb it.
When in a bog say "BUNYON" to relieve yourself of two burdens.
Up from the lower levels is possible.

GHOST TOWN

Don't get into a bother about a locked door just wave your hat.
All musicians like praise.
Burn the grass to get to the mine.
If it gets dark go to a hotel for the night and SLEEP.

GOLDEN VOYAGE

Be nice and pay for something.
Once at sea find land *push the statue *dig twice.
Everbody PRAY for a passageway.

H.H.G.T.T.G. *********

On top of the satchel.

If you find yourself enjoying poetry
you could plot your space walk.

The memorial is hard to miss use
stones for words can harm you.

One more thing to keep up the flow of hints I need some DEADLINE hints

RODNEY G.

Thank you ADVENTURE HOTLINE.

>>

I

50

va

my

me

HM!

Fi

ot

ou.

Ca

CO

in

ca sh Wi

di

the pu: up

la: Se

(1)

P. E. UNIT

FURTHER RAMBLINGS

BH RON KLEINSCHAFER, H.V. SBERS

different sections on the HMPEU some masking tape. after publication of the article. Measure the diameter of the assembly most were only minor details to and get some spaghetti tubing the clear up sections nat understood but some expressed over the now shielded wires, smear difficulty in locating the shielded it lightly with petroleum jelly then cable to connect the I/O port board feed through the pull through wire asthetic appearance (Look at TI's end and apply heat. out.

>>> ROLL YOUR OWN <<<

some multi way shielded cable for slightly open over the wires HMPEU for testing and the results together. were most gratifying.

outer braid that is well plaited, used on the TI PEU. Carefully pull out the inner conductor(s) then the large in diameter.

Select of light RK. 35 lengths (10*0.02mm) hookup wire, cut

length required, and solder one end of each to a single length of heavy hookup wire, also get a length of single core light gauge shielded cable (such as used in VCR remotes etc), remove the outer insulation and solder to the heavy wire with the others, this wire is used for the Ø3 (PHASE 3) signal line as this line is constantly running at a high frequency and is a major source of RFI. After this is done lay the wires out in a 'nice round bunch, Slide the braid on the dowel onto the single heavy wire and feed it over the 36 wires, By running your hand down the braid it can be pulled tight onto the loom, solder this also to the heavy pull through wire. It is suggested that you use 16 wires of one color for the address lines, 8 different for the data lines and then other colors for the rest, this makes it easier to identify (with a means for checking I have had several enquiries on continuity) and label each end with

fully required length that will fit neatly to the main board and felt that IDC and pull on the spaghetti tubing. cable is less than ideal being very To finish off the ends fit a short messy to try and shield with no length of heat shrink tubing either All that is bulky cable), well there is a way required now is to bare back the wires and when filling earth the outer braid to the case and solder the light gauge shielded cable's outer braid to ground on each board. I have on occasions had need for Although the shielding braid may be it various applications and had to make makes quite a satisfactory cable my own, so I made one up using the with no RFI experienced whilst method as below and fitted it to my accessing disk drives and printer With this method it is possable to make shielded cables of First obtain a length of RG/8AU or any number of lines (even one to other large Coaxial cable with an replace the flat cable currently

inner In the article on the Homade PEU insulation (dielectric) then V-E-R-Y (issue No 8 of the HV/99 newsletter) carefully pull out the braided outer there were a few lines of text shield from the outside insulation. Improve and it is not very clear on With a length of dowel about 10mm the photostat copy of the main PCB. diameter, round it on the end, start There should be a .1uF monolith the shield braid over the end then capacitor soldered near pins 19 and push it onto the dowel bunching it $2\emptyset$ on IC's U1-U2-U3, and near pin $2\emptyset$ up as you go, it will expand fairly and ground (marked .1) on U4. These are to help decouple the supply.

to to

٥f

it t off

ke to

that

IKERS

ough. small

bers.

other owing can't

just

mmend

maybe

YOUR

15

bout a

ine. for

sh the

poetry

ss use

flow hints

LIBRARIANS REPORT

HI 99'ers. ****Ultimate Disk*** This would have to it! A Number 1 on anyone's list. Imagine a disk that makes 4 Modules obsolete, but then can BOOT from anyone of them, in case one goes Phutt!Or you have not got a Widgett(D.I.Y or otherwise.)then you may leave whichever one you were using E/A, MINI-MEM, in; TI-Writer(UTIL1), XBload Autoruns. Runs c99 V2.0, Show Directory, Edit, Formatter, Assembler, DM1000 v3.1, Utilities, D-Patch, Assembly Programs - all from one DISK, return from all programs to the sub menu to Main Menu using FCTN 9, change the screen colours to suit, manage your Disk and much more. Thanks to FREE exchange of information from Hunter Valley 99'ers ,Ottawa TI-99/4U.U.G. Canada Clint Pulley.

All this has been put together ONE package by Tony Will McGovern just when we thought it had been all done. The ammount of Effort, Time and Coffee that has gone into this has Any clubs or individuals interested been incredable and I am sure that all who use it will agree that it is software in volume have 2 choices commercial effort's costing several hundred Dollars.

****This is V 3.3 FUNNELWRITER and it is now available on 2 Disks. Yes 2 Disks (that is if you have only got a SS/SD Drive, as it was impossible to document and fit it all onto i Disk. but it is still great value.

But if you have DS/SD it will all be on 1 Disk.***

FAIRWARE and as such only nominal handling costs should be charged when it is being distributed by USER GROUPS and NONE by individuals.For Hunter Valley 99'ers and MOST true user groups this is about \$1.00.

As FAIRWARE no matter how you got We have just received from the Melb. your copy you are Morally Obliged to User Group. The Program you put first acknowledge the Authors by sending on your casssette to LOAD MEMORY

use it all the time and have it on to someone else.) It is common for some user groups in U.S.A. to collect and foward suitable rewards tο **FAIRWARE** Authors.

Tony, Will and the Hunter Valley 99'ers have had great exchange of programs and other more SOLID and USEFUL GOODIES to make it all worth while. From all over the Globe the Mail keeps coming in THANK YOU to all Groups and Individuals who have made it all possible. I know Tony and Will really appreciate donations and we here are happy for Tony. THANK YOU from all HV99'ers.

****** V 3.3 ****** It is available through the Hunter Valley 99'ers Library for only \$5 SS/SD inc.PP.anywhere in Australia. Overseas extra for Surface Airmail as requested. WHY PAY MORE

D i = k **Mewe**

We have in the past month received the latest DM1000 v3.1 and small c99 (2 disks)and 2 disks ⊔f ⊑99 programs. A budget and cheque book system to keep track of your \$\$\$'s.

Software Library

in obtaining any PUBLIC DOMAIN better than most of, NO ALL of the (a) Send blank initialised disks to us with return postage or send us disks with programs on it and we will send at our cost an equal number of disks filled with programs requested or volume disks. (b) We can supply programs, or volume disks on our disks for the cost of

t

F

Ha

ΑI

¥¥

*×

Υc

€×

ma

th

We are not a SHOP and do not sell for PROFIT, any excess is used to buy more material for club usage. Cassette tapes only to be available ALL V 3.0 and up FUNNELWRITERS are at club monthly meetings owing to the problems associated with them. Your library has about 1500 programs on 110 disks.

Cassette are still a Good buy at the HV99'er monthly meet for \$ 3.20.

STOP PRESS.

disk and PP(\$4.00)

them something in return you do not IMAGE FILE Assembley programs from

s in bward WARE

ssed

alley e of and worth the

bu to have Tony the y for

i 5 alley **3**3/3D

Θŕ MORE

eived 1 599 c99 book \$'s.

ested OMAIN es ks to d u.s nd we equal

olume st of

grams

sell ed to Ēρ. lable ing to lem.

t the

grams

Melb. first REMORY from

cassette for the people who now have the 32K in the console and cassette's only. This has been placed in the Public Domain .THANKS to Peter Kennedy Wayne Southwick. Now all we need are some good Public Domain Assembly programs to go with the loader.

User Pays System

Well up here in the Hunter Valley we are all familiar with this system as well as the tear drop or drip!!!!. But only to the system adopted by our local Utility that distributes our WATER to consumers in the area. NOT as a User Group. Here all the Other User Group News Letters, Books, Ideas, Information, as well as TIME from Committee and Indivuals are given FREE in true user group spirit Maybe we are Lucky we have not got Hundreds in the group and Thousands in Swiss Bank accounts!!!. Maybe that is why we are stiil

U.G. and not a company.

I am surry to have to say this, so far I can count on one FINGER the ammount of SOLID Thanks to Tony Will that **the**y have received from Australian User Groups yet we know it is being SOLD and used here.I have not seen many TI users(or user groups) with SHORT ARMS and LONG POCKETS.So if you have forgotten and this applies to you DO SOMETHING soon.

Fairware for now, Happy programing, Al Lawrence.

UPDATE UPDATE

Your module library has yet expanded we now have atotal of 10 modules i set ofjoysticks and i tape the modules are:

> 1.EXTENDED BASIC 2.THE ATTACK

> 3. TOMBSTONE CITY

4.MIND CHALLENGERS

5.STAR TREK 6.DRAGON MIX

7. CAR WARS

S.ALLIGATOR MIX

9.ZERO ZAP

10. TOMBSTONE CITY

FASTER CASSETTE FILES

BY BRIAN BUTHERFORD HUNTER VALLEY BBERS

The first thing I notice with most programmes that use files, is that they usually give the user the choice of cassette or disk. This works reasonable well with files saved on disk as any inefficencies do not really show up, because disks are fast compared with cassetts. But with most programmes with a disk or cassette option, you can pack and eat a cut lunch whilst the files are being read or saved by a cassette.

To try and speed things up it is important to understand the options you are allowed with cassettes, Firstly the files can be DISPLAY type or INTERNAL. DISPLAY is what it means, the data is saved to the file exactly as it would be printed to the screen. That means that comma's and semi-colon's are print seperators not item seperators. We come now to INTERNAL type records. the prefered type for cassette files as they are much easier to format using print seperators and pending prints. Next the OPEN mode, with cassette you can only specify INPUT or OUTPUT. Lastly the type and length of the files. With FIXED lengths of 64, 128 or 192 bytes. With 64 being the default if you do not specify a length. Now to put this all together.

Firstly we always use INTERNAL type records of 192 bytes long where possible, this means you must be carefull in designing your programme. You may have to limit the length of the item(s) to enable you to fit a certain number of items in a 192 byte record. Now the number of bytes a string uses, is the length of the string plus i

byte for the length of the string. A number always uses 8 bytes plus i byte for its length. So from this we can see it is better to convert numbers to strings with the str# function, for unless they are longer than eight digits you save space that way. Then using all those ideas, you then store you records in the PRINT BUFFER using PENDING PRINT statements until the record is 192 bytes long, or as close as possible to that. It does not have to be exact as the computer will pad out the record to the length that was specified. Let us say you keep the records for some sort of sporting club, and you keep on file the members names and a score. You could nominate to allowe 15 letters for the name, and knowing that the scores will not exceed three digits. That gives you 16 bytes needed for the name, and if you convert the score to a string, 4 bytes for the score. All together 20 bytes which means you could load 9 sets of names and scores into the PRINT BUFFER using a pending print type statement. The computer would print 9 sets of names and scores on to the tape at a time. If there were 60 members, the computer would only have to print 7 lots of information on the tape not 60 little items. The programme to do this could look like this.

100 CALL CLEAR

1000 END

Notice the semi-colon after SCORE#(I+J) in line 550. That is the PENDING PRINT, when the computer reads that it stores the data in the buffer until the next print statement, and it keeps doin that until we get to line 570 which has no comma or semi-colon at the end of it. So the computer takes that item and all the items it has stored in the buffer and prints them out to the tape in the one 90.

The last hint I have, is to have a set number of files so you do not have to save a value for the loop counter. As nothing slows cassette files down more than having a lot of starts and stops, because the start tone uses up as much time as the data loading. Just to give you some idea of how fast cassette files can be, I have a programme that I wrote which saves or reads 300 seperate data items in approximatly two and a quarter minuts. So don't write off using cassettes for keeping your files i that is the only system you have, little bit of thought and planning can work wonders.

SUBSCRIPTIONS

Don't forget: SUBSCRIPTIONS ARE NOW DUE for the next year. If you are paying by mail, please send the subscription (rates are detailed (the back cover) to:The Secretary,
Hunter Valley 99'ers Users Group 4 Arcot Close,
TAPRO N.S.W. 2322

a

S

d:

V

CQ

Payments may be made at the meeting to the Treasurer, Brian Rutherford,

Please pay promptly to ensure you don't miss any copies of the Clul Newsletter.

t is

s the

next
s doing
Ø which
t the
akes
it has
nts

one

have a
o not
loop
ssette
a lot
the
ime as
ve you
te
amme
reads

er sing iles if have, a anning

18

ARE NOW you are end the ailed o

roup

meeting erford.

sure yo: e Club

COMMUNICATIONS

How much is your computer really worth?...

My old TI is worth millions because it 'aint jest any old TI.'

By the addition of a simple and relatively inexpensive card my TI can control my friend's IBM or it can be as powerful as a mainframe computer.

I can use my TI to access information (eg My eldest daughter, who is studying clinical psychology at university, simply calls up the Uni computer from home and takes control of the big VAX with my little 'ol TI.(I get a kick out of that)).

I can use it as a sort of answering machine, to friends similarly equipped; I can use it to send programs I have written, to friends and I can down-load programs from them too.

I can share files with other types of computers (eg one of my friends has a Microbee and another an IBM.. We can exchange text files etc even rude pikkies such as 'diskart').

I can have others help me with program bugs by actually letting them run the program under development without me having to physically meet them to give them a copy.

I can even send this story to the editor without leaving my study (That is if the editor had a modem too.)

I am sure you have all thought at some time about buying one of these smart things called a **MODEM**.

Well , if you can..DO...

I have a simple 300 baud card which is located in the PE box.It connects up to the RS232 card plug at the back and the telephone lines. (I even managed to put it in the box so it must be easy)

Since I have bought mine, many newer ones which will allow you to use that hallowed institution called VIATEL, have come onto the market.

These modems opperate at different speeds and are more versatile than mine, however they do cost more.

Some have 'auto answer'and 'autodial' facilities, others have jazzy sensors which detect the baud rate of the calling modem and adjust to it. They are smart (and dear)

A number of club members have modems and we get in touch with one another fairly often. It is good to use our computer in this way as we feel it is a very interesting way of communicating.

Another thing we do is to contact various BULLETIN BOARDS.

These are simply computers, set up by dedicated hobbiest, running a program which allows you to look at information stored on its disk, by calling it on the phone.

Messages may be sent and stored, as a form of electronic mail; articles and magazines may be read; programs may be 'uploaded' and 'downloaded'; Any form of notice may be publicised; forums may be set up and even jokes may be exchanged.

Hams have a similar idea unly they use radio.(It's cheaper too.)

The idea of this club running a bulletin board is very appealing however it would require a dedicated telephone line and a dedicated, fully expanded computer with 'lottes memory'.

This is rather out of our range at the moment, especially when there are other ways around the problem.

If enough of us had modems and suitable software to drive them, maybe one of the existing local computer club bulletin board services might be able to reserve some space on one of its disks specifically for our TIs.

This would, of course benefit both clubs as we would become active members using the facility, and we would have a cheap BBS.

I think the idea is worth considering seriously.

If you get half a chance, make a point of visiting either David Winten, John Payton or myself on the Long Weekend so that you can see what I'm talking about. (The gang going to Melbourne is linking to us by modem, and we will be demonstrating our goodies to you.)

I would dearly love to convince you of the need you have for a modem

I think the best thing you can do is to talk to someone who has one.

Please do.

Pete Smith

GRAND PRIX

BY JOHN SMART, H. V. 99ERS

```
1 RANDOMIZE
 2 DIM J$(1Ø), J(1Ø)
 3 FOR S=1 TO 10 :: J$(S)="..
 .." :: J(S)=999 :: NEXT S
 10 CALL TITLE
 20 CALL CLEAR
 100 CALL CLEAR :: CALL MAGNI
FY(3)
 110 CALL CHAR(128, RPT$("01"
16)&RPT$("8Ø",16),136,RPT$("
F",16))
115 CALL CHAR(132, "Ø1Ø7Ø1@5Ø
70503070E0E031B1F1B071F80E08
ØAØEØAØCØEØ7Ø7ØCØD8F8C8EØF8"
116 CALL CHAR(100, "0000000020
3072C1D0F070D0F1101000000000
Ø1@EØFØFØDCDØBØEØAØ1@ØØØØØØ"
117 CALL CHAR(1Ø4, "ØØØØØ2ØF4
92B1FØF1F3FØE1321Ø1ØØØØØØØØ
ØEØFØF1FED8DØD164DØ88ØØØØ")
118 CALL CHAR (108, "4022251F9
1574C3D1F1E3F1F3E334181Ø2Ø49
CB8F8F8FCBFAEECCCFE7388Ø4Ø2"
119 CALL CHAR(112, "1121Ø1112
34103071E4E1313274107E180078
ØAØEØA2C1ØØ3Ø11C1C2C4C8EØF8"
12Ø FOR C=Ø TO 8 :: CALL COL
OR(C,16,3):: NEXT C :: CALL
COLOR (14, 15, 15)
121 CALL CHAR(116, RPT$("ØØ",
7)%"Ø3Ø3"&RPT$("ØØ",48))
122 CALL CHAR(120, "000202030
3030307050910080703010100102
ØCØCØCØCØEØAØ9ØØ81ØEØCØ8Ø8Ø"
123 CALL CHAR(124, "ØØØ8Ø4Ø3Ø
3030307050910080703010100404
ØCØCØCØCØEØAØ7ØØ81ØEØCØ8Ø8Ø"
125 CALL CHAR (96, "0003070706
ØCØFØFØFØFØCØFØFØFØFØØØØEØFØ
FØ3Ø18F8F87878187878F8F8")
```

```
127 CALL SCREEN(11):: CALL V
 CHAR(1,11,136,288)
 13Ø CALL SPRITE(#1Ø,128,16,1
 ,120,#11,128,16,56,120,#12,1
 28,16,112,120,#13,128,16,168
 ,120,#1,132,9,160,120)
 14Ø CT=Ø :: CSP1=8Ø+INT(RND*
 32)-16 :: CSP2=8Ø+INT(RND*32
 )-16 :: CP,SP=Ø :: AC=16
200 CALL SPRITE(#2,132,INT(R
ND*11)+3, INT(RND*64)+1, INT(R
ND#8Ø1+92)
21Ø ! LET THE RACE BEGIN !
22Ø CALL POSITION(#1,X1,Y1,#
2,X2,Y2)
23Ø IF Y1<78 OR Y1>166 THEN
CALL CRASH(SP,CP,CSP1,CSP2)
24Ø IF X2-1Ø>16Ø THEN CALL R
ELOCATE(1,2,CP,CSP1,SP,Y1)
26Ø MF=Ø :: CALL KEY(Ø,K,S):
: IF S=Ø THEN SP=SP-1 :: GOT
0 35Ø
27Ø IF K=13 THEN SP=SP+AC-(1
6/127) *SP :: GOTO 35Ø ELSE S
P=SP-1
28Ø IF K=83 THEN M=-12 :: MF
=-1 :: GOTO 35Ø ELSE IF K=68
 THEN M=12 :: MF=-1 :: GOTO
35Ø
29Ø IF K=32 THEN SP=SP-16 ::
 IF SP(Ø THEN SP=Ø
295 IF K=87 THEN 1500
35Ø CALL COINC(#1,#2,10,T)::
 IF T=-1 THEN CALL CRASH(SP,
CP, CSP1, CSP2)
351 IF MF()-1 THEN M=Ø
352 CALL MOTION(#1,0,M,#2,SP
-CSP1,Ø,#1Ø,SP,Ø,#11,SP,Ø,#1
2,5P,Ø,#13,5F,Ø)
353 CSP1=CSP1+.2 :: IF CSP1>
12Ø THEN CSP1=12Ø
355 IF SP(Ø THEN SP=Ø
356 IF CP>31 THEN CP=31
```

and the state of the

360 CT=CT+1 :: DISPLAY AT(2, 1) SIZE (8): "KM/H: "&STR\$(INT (S P*2.5)):: DISPLAY AT(3,1)SIZ E(8): "RANK: "&STR#(32-CP);; D ISPLAY AT(4,1)SIZE(8):"TIME: "&STR\$(CT) 37Ø CALL POSITION(#2,X2,Y2): : CALL COINC(#1,#2,10,T):: I F T=-1 THEN 400 ELSE 450 400 CALL CRASH(SP,CP,CSP1,CS 45Ø IF CT>299 THEN 5ØØ 46Ø CALL SOUND(-999,-3,Ø,12Ø +SP/6,Ø) 47Ø CALL COINC(#1,#2,1Ø,T):: IF T=-1 THEN CALL CRASH(SP, CP, CSP1, CSP2) 48Ø GOTO 22Ø 500 CALL DELSPRITE(ALL):: CA LL CLEAR :: CALL SCREEN(3) 510 DISPLAY AT(2,2): "TIME UP 520 DISPLAY AT(4,2): "FINAL R ANK: "%STR#(32-CP):: RANK=32-CP 530 FOR D=1 TO 250 :: NEXT D :: CALL NEON 540 IF RANK<J(10)THEN 1000 35Ø CALL NEGN 560 CALL CLEAR :: DISPLAY AT (2,2): "TOP SCORE TABLE" 57Ø DISPLAY AT(4,1): "PLACE RANK NAME" 580 FOR S=1 TO 10 590 DISPLAY AT(5+S,1):STR\$(S >&"...."&STR≠(J(S))&"..., "&J\$(S) 600 NEXT S 610 CALL NEON 62Ø GOTO 1Ø 1000 CALL CLEAR :: DISPLAY A T(2,1): " CONGRADULATIONS!YO U HAVE ONE OF TODAY'S TOP T ΕN SCORES!PLEASE ENTER YOUR NAME." 1005 ACCEPT AT(6,2)SIZE(10): **NAMES** 1010 FOR X=1 TO 10 1020 IF RANK(J(X)THEN 1050 E LSE 1030 1030 NEXT X 1040 GOTO 10 1050 FOR Z=10 TO X+1 STEP -1 1000 J(Z)=J(Z-1):: J*(Z)=J*(Z-1)1070 NEXT Z 1090 J\$(X)=NAME\$ 1100 J(X)=RANK

1110 GOTO 550 1121 CALL CHAR(116, RPT\$("ØØ" ,12)&"Ø8Ø8"&RPT\$("Ø",32)) 1500 CALL DELSPRITE(ALL):: C ALL CLEAR :: DISPLAY AT(2,2) :" YOU WITHDREW FROM THE RAC E!" :: CALL NEON :: GOTO 10 10000 SUB TITLE iøøiø CALL CLEAR 10020 FOR C=0 TO 8 :: CALL C OLOR(C,16,1):: NEXT C 10030 CALL SCREEN(3) 10040 DISPLAY AT(11,10): "GRA ND PRIX" 10050 DISPLAY AT(13,10): "BY JOHN SMART" 10055 DISPLAY AT(15,4): "DONA TED TO HV99'ERS 1986" 10060 CALL NEON 10070 CALL CLEAR 10080 DISPLAY AT(2,2): "INSTR UCTIONS?(Y/N)" 10090 ACCEPT AT(2,20)SIZE(1) BEEP: I + :: IF I = "Y" OR I = " y" THEN CALL INSTR 10100 SUBEXIT 10110 SUBEND 11000 SUB NEON 11Ø1Ø M\$="HIT ENTER TO CONTI NUE...." 11020 M==SEG\$(M\$, LEN(M\$)-1,1) & SEG \$ (M\$, 1, LEN (M\$) - 1):: DIS PLAY AT (23,1):M\$ 11030 CALL KEY(0,K,S):: 1F K <>13 THEN 11020 11Ø4Ø SUBEXIT 11050 SUBEND 12000 SUB INSTR 12Ø1Ø CALL CLEAR 12020 DISPLAY AT(2,10): "INST RUCTIONS" 12030 DISPLAY AT(4,1):" WEL COME TO GRAND PRIX! THE OBJECT OF THIS GAME IS TO WIN THE RACE BE OVER TAK ING ALL OF THE OTHER " 12040 DISPLAY AT(8,1):" CAR S.THERE ARE OVER 100 CAR S IN THE RACE. YOU BEGIN IN THE SØTH POSITION, AND YOU

MUST WORK FROM HERE."

Стати в пример по прим

12050 DISPLAY AT(12,1):" YO U CRASH WHEN YOU RUN OFF TH E ROAD OR WHEN YOU CO LLIDE WITH ANOTHER CAR. ΕN TER IS THE ACCELERATOR, " 12060 DISPLAY AT(16,1):" TH ' S E SPACE BAR IS THE BRAKE , D MOVES YOU LEFT AND ' MOVES YOU RIGHT." 12070 DISPLAY AT(19,1):" YO AB UR MAXIMUM SPEED IS OUT 32Ø KM/H.AS YOU NEAR TH E LEADING CARS THE CARS WI LL INCREASE THEIR SPEED." 12080 CALL NEON 12100 DISPLAY AT(2,10):"INST RUCTIONS" 1211Ø DISPLAY AT(4,1):" OTH ER OBSTACLES STAND IN YOU WR R WAY, HOWEVER. EAKAGE: OCCASIONALLY YOU WIL L ENCOUNTER THE 12115 DISPLAY AT(8,1):" WRE AKAGE OF OTHER CRASHED CAR S.THE SCREEN FLASHES RED WHEN WREAKAGE NEARS. A H IT WILL CAUSE YOU TO" 1212Ø DISPLAY AT(12,1):" CR P TH HOTOGRAPHERS: RUN ONTO E ROAD AND DODGE BACK AN D FORTH TO GET A GOOD" 1213Ø DISPLAY AT(16,1):" SH IN OT. THERE MAY BE SEVERAL A ROW. oc MBULANCES: AMBULANCES CASIONALLY CRUISE PAST" YO 1215Ø DISPLAY AT(20,1):" GO UR CAR.A SIREN WILL CO BEFORE IT APPEARS.A LLISION CAUSES A CRASH." 1216Ø CALL NEON 1219Ø SUBEXIT 12200 SUBEND 2000 SUB RELOCATE (F,C,CP,CS P1,SP,Y1) 20005 CALL MOTION(#C,0,0):: CALL COLOR(#C,1):: IF RND*5< 1 THEN CALL OBST(SP,CP) 20010 CALL LOCATE(#C,246,Y1) :: CL=INT(RND#11)+3 :: IF CL =12 OR CL=11 THEN CL=1Ø 20011 CALL COLOR(#2,CL) 20015 IF SP>CSP1 THEN CP=CP+

1 ELSE CP=CP-1

20020 SUBEND

21000 SUB CRASH(SP,CP,CSP1,C SP2) 21005 CALL DELSPRITE(#5):: C ALL COLOR(#1,10):: CALL SOUN D(-4000,-7,0):: CALL PGSITIO N(#1, X, Y)21006 CALL SPRITE(#20,116,10 ,X,Y,-8,8,#21,116,1Ø,X,Y,8,8 ,#22,116,10,X,Y,8,-8,#23,115 , 1Ø, X, Y, -8, -8) 21010 FOR C=SP TO 0 STEP -12 :: CALL MOTION(#1,Ø,M,#2,C-CSP1,Ø,#3,C-CSP2,Ø,#1Ø,C,Ø,# 11,C,Ø,#12,C,Ø,#13,C,Ø,#2Ø,C /16,Ø,#21,C/16,Ø) 21Ø11 CALL MOTION(#22,C/16,Ø ,#23,C/16,Ø) 21020 DISPLAY AT(2,1)SIZE(8) : "KM/H: "&STR\$(INT(C*2.5)):: NEXT C :: SP=Ø 21025 CALL SOUND(-4000,-7,0) :: CALL DELSPRITE(#20,#21,#2 2,#23,#5):: CALL PATTERN(#1, 100):: CALL COLOR(#1,7):: FD R D=1 TO ZØ :: NEXT D 21Ø26 CALL PATTERN(#1,1Ø4):: CALL COLOR(#1,1Ø):: FOR D=1 TO 25 :: NEXT D 21027 CALL PATTERN(#1,108):: CALL COLOR(#1,11):: FOR D=1 TO 3Ø :: NEXT D 21028 CALL PATTERN(#1,104):: CALL COLOR(#1,10):: FOR D=1 TO 25 :: NEXT D 21029 CALL PATTERN(#1,100):: CALL COLOR(#1,7):: FOR D=1 TO 25 :: NEXT D 21030 CALL PATTERN(#1,112):: CALL COLOR(#1,2):: CALL MOT ION(#1Ø,Ø,Ø,#11,Ø,Ø,#12,Ø,Ø, #13,Ø,Ø):: FOR D=1 TO 1000 : ; NEXT D 21Ø35 CALL COLOR(#1,9):: CAL L PATTERN(#1,132) 21Ø4Ø CP=CP-3 219ØØ SUBEND 25000 SUB OBST(SP,CP) 25010 ON INT(RND*3)+1 GOSUB 25050,25060,25070 25020 SUBEXIT 25030 ! 25040 ! 25Ø5Ø CALL ACCIDENT(SP,CP):: RETURN

25ø6ø CALL PHOTO(SP,CP):: RE

TURN

RN 25080 SUBEND 26000 SUB ACCIDENT(SP,CP) 26Ø1Ø F=Ø :: FOR L=1 TO 5 :: CALL SCREEN(9):: GOSUB 261Ø Ø :: NEXT L :: CALL SCREEN(1 1):: GOSUB 261ØØ 26020 F=1 :: CALL SPRITE(#5, 112,2,16, INT(RND*8Ø)+92,SP,Ø 26030 GOSUB 26100 :: CALL CO INC(#1,#5,24,T):: IF T=-1 TH EN CALL CRASH(SP,CP,Ø,Ø):: G OTO 26090 ELSE GOSUB 26100 26040 CALL POSITION(#5,X,Y): : IF X>170 THEN 26090 26050 GOSUB 26100 :: CALL CO INC(#5,#1,24,T):: IF T=-1 TH EN CALL CRASH(SP,CP,Ø,Ø)ELSE 26939 26060 GOTO 26030 26080 ! 26090 CALL DELSPRITE(#5):: S UBEXIT 26100 DISPLAY AT(2,1)SIZE(8) ;"KM/H:"&STR\$(INT(SP*2.5));; CALL KEY(Ø,K,S):: IF S=Ø TH EN 2614Ø ELSE IF K<>13 THEN SP=SP-1 26105 IF F=1 THEN CALL COINC (#1,#5,24,T):: IF T=-1 THEN CALL CRASH(SP,CP,Ø,Ø):: GOTO 2611Ø IF K=13 THEN SP=SP+8-(8/127) *SP :: GOTO 2614Ø ELSE SP=SP-1 2612Ø IF K=83 THEN M=-12 :: GOTO 2614Ø ELSE IF K=68 THEN M=12 :: GOTO 2614Ø 24130 IF K=32 THEN SP=SP-14 :: IF SP<Ø THEN SP=Ø :: GOTO 2615Ø ELSE GOTO 2615Ø 2614Ø IF K=68 OR K=83 THEN 2 6150 26145 M±Ø 2615Ø CALL SOUND(-999,-3,Ø,1 20+SP/6,0):: CALL MOTION(#1, Ø,M,#1Ø,SP,Ø,#11,SP,Ø,#12,SP ,Ø,#13,SP,Ø):: IF F=Ø THEN R ETURN 26155 CALL POSITION(#5,X,Y): : IF X>170 THEN 26090 ELSE C ALL MOTION(#5,SP,Ø) 2616Ø RETURN 26999 SUBEND

ŘΕ

antimater in the first the contract of the con

25070 CALL AMB(SP,CP):: RETU

27005 F=1 :: CALL SPRITE(#5, 12Ø,14,16,INT(RND*8Ø)+92,SP, Ø) 27Ø1Ø GOSUB 271ØØ 27030 GOSUB 27100 :: CALL CO INC($\#1,\#5,2\emptyset,T$):: IF T=-1 TH EN CALL CRASH(SP,CP,Ø,Ø):: G OTO 27090 ELSE GOSUB 27100 27Ø4Ø CALL POSITION(#5,X,Y): : IF X>170 THEN 27090 ELSE C ALL PATTERN(#5,120) 27050 GOSUB 27100 :: CALL CO INC(#5, #1, 2Ø, T):: IF T=-1 TH EN CALL CRASH(SP,CP,Ø,Ø)ELSE 27Ø3Ø 27Ø6Ø CALL PATTERN(#5,124):: GOTO 27Ø3Ø 27080 ! 27090 CALL DELSPRITE(#5):: S UBEXIT 27100 DISPLAY AT(2,1)SIZE(8) : "KM/H: "&STR\$(INT(SP*2.5)):: CALL KEY(Ø,K,S):: IF S=Ø TH EN 2714Ø ELSE IF K<>13 THEN SP=SP~1 27105 CALL COINC(#1,#5,24,T) :: IF T=-! THEN CALL CRASH(S P,CP,Ø,Ø):: GOTO 27Ø9Ø 2711Ø IF K=13 THEN SP=SP+8-(8/127) *SP :: GOTO 2714Ø ELSE SP=SP-1 2712Ø IF K=83 THEN M=-12 :: GOTO 2714Ø ELSE IF K=68 THEN M=12 :: GOTO 2714Ø 2713Ø IF K=32 THEN SP=SP-16 :: IF SP<Ø THEN SP=Ø :: GOTO 2715Ø ELSE GOTO 2715Ø 2714Ø IF K=68 OR K=83 THEN 2 715Ø 27145 M≃Ø 27150 CALL SOUND(-999,-3,0,1 2Ø+SP/6,Ø):: CALL MOTION(#1, Ø,M,#1Ø,SP,Ø,#11,SP,Ø,#12,SP ,Ø,#13,6P,Ø):: IF F=Ø THEN R ETURN 27155 CALL POSITION(#5,X,Y): : IF X>170 THEN 27090 ELSE C ALL MOTION(#5,SP,INT(RND*1Ø) -5) 2716Ø IF P=1 THEN CALL PATTE RN(#5.120):: P=0 ELSE CALL P ATTERN(#5,124):: P=1 2717Ø RETURN 27999 SUBEND

27000 SUB PHOTO(SP,CP)

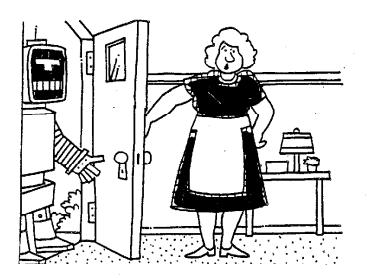
28000 SUB AMB(SP,CP) 28005 F=0 :: CALL MOTION(#1. Ø,Ø) 28010 FOR S=1 TO 3 :: CALL S OUND(375,500,0):: CALL SOUND (375,45Ø,Ø):: NEXT 8 28020 F=1 :: CALL SPRITE(#2. 96, 10, 248, INT (RND*80) +92, -16 ,Ø) 28030 GOSUB 28100 :: CALL CO INC(#1,#2,16,T);: IF T=-1 TH EN CALL CRASH(SP,CP,100,0):: GOTO 28090 ELSE GOSUB 28100 28040 CALL POSITION(#2, X, Y): : IF X<36 THEN 28Ø9Ø 28050 GOSUB 28100 :: CALL CO INC(#2,#1,16,T):: IF T=-1 TH EN CALL CRASH(SP,CP,100,0)EL SE 28Ø3Ø 28040 GOTO 28030

28080 !
28090 CALL LOCATE(#2,240,96)
:: CALL PATTERN(#2,132):: SU
BEXIT
28100 DISPLAY AT(2,1)SIZE(8)
:"KM/H:"&STR\$(INT(\$P*2.5))::
CALL KEY(Ø,K,S):: IF S=0 TH
EN 28140 ELSE IF K<>13 THEN
SP=SP-1 :: IF SP<0 THEN SP=0

28105 IF F=1 THEN CALL COINC (#1,#2,16,T):: IF T=-1 THEN CALL CRASH(SP,CP,1ØØ,Ø):: GO TO 28090 2811Ø IF K=13 THEN SP=SP+8-(8/127)*SP :: GOTO 2814Ø ELSE SP=SP-1 :: IF SP<Ø THEN SP= Ø 2812Ø IF K=83 THEN M=-12 :: GOTO 2814Ø ELSE IF K=68 THEN M=12 :: GOTO 2814Ø 2813Ø IF K=32 THEN SP=SP-16 :: IF SP(Ø THEN SP=Ø :: GOTO 2815Ø ELSE GOTO 2815Ø 2814Ø IF K=68 OR K=83 THEN 2 8150 28145 M=Ø 2815Ø CALL SOUND(-999,-3,Ø,1 2Ø+SP/6,Ø):: CALL MOTION(#1. Ø, M, #1Ø, SP, Ø, #11, SP, Ø, #12, SP ,Ø,#13,SP,Ø):: IF F=Ø THEN R ETURN 28155 CALL POSITION(#2,X,Y): : IF X<36 THEN 28Ø9Ø 2816Ø RETURN 28999 SUBEND

MEMBERSHIP NOW DUE

and the control of th



He's from the computer club dear. He says you're behind in your dues. bit ago: do t the it i

MEMO

Fort

Open memo arti Thes the manu Is i the successions

No, use you autor as i of ti have is. as a and i new i and wi11 nest write comp1 longe dicti memor Compu addin

IE th avail

avail the m



Today I thought I'd talk a bit about something I promised ages ago: VDP reads and writes. Before we do this let's take a brief look at the architecture of the Ti994a as it is relevant to programming in Forth.

MEMORY MAP.

Open your manual to chapter 4 on memory maps and as you read this article refer to pages 3,4,5,6. These are reprinted at the end of the article for those without the manual.

Is it really necessary to understand the bowels of your computer to be a successful Forth Programmer? The answer is really No, but yes.

No, because if you do not elect to use specific memory addresses for you own purpose Forth will automatically allocate them for you as it sees fit, within the confines of the dictionary. You don't even have to know where the dictionary is. You can conceptually think of it as a book, a la concept of Brodie and his monk scribe, entering all new words you define with a quill and ink. Anything you write in Forth will be sequentially allocated the next spot on the page, and you can write programs of ever growing complexity until your scribe can no longer fit them in his book, ie the dictionary will be full. At the memory address level within the computer, we can similarly keep adding words which fill up all available spots, until we receive the message "dictionary full".

IE there is a finite amount of space available to use for Forth words.

Consulting page 6 this stretches from HEX AØØØ-> FF9F which is a total of 24479 bytes. When you boot the system disk all the binary code held on screens 8 thru 19 is loaded into the first part of the dictionary. This is the Forth Kernel -corresponding to all words marked "resident" in the listing in the appendix of the manual. It takes up. about 7296 bytes, and although it is possible to delete parts of this by various means, lets consider it inviolable for the moment is we cannot access this space. We now have about 17000 bytes left. Any options you load on top of this eg graphics, assembler, floating point, also take up space in the dictionary quite a considerable amount. Again, for the purpose of this exercise lets consider this inviolable, although in practice its dead easy to prune massive amounts out by re-editing the source code, a topic I'll discuss at a later date. Hence we may now have only 7 or 8K of dictionary space left for our program. Not much you may say!. Well, Forth code compacts down amazingly, yet even so one can easily run out of space. Say for example one has a program where we want to accept up to twenty sets of data, each set taking up 100 bytes each. We allocate this by:

Ø VARIABLE DATA 1998 ALLOT.

ie allot a 2000 byte buffer. Chop!! Masses of dictionary space gone already. Then within a program one might have alot of Screen displays of lists/screen formats which by dint of their expansive nature take up alot more space. Suddenly, what seemed an adequate space has shrunken dramatically.

What other space exists in the computer which is either not being used, or will not be used when the program is running? Scanning the memory maps will show 2 massive hunks:

1.8000 odd K of VDP ram at HEX1400. 2.5000 odd K at HEX 2010 which the system uses for the five 1k disk buffers.

Lets expand on number 2.

The state of the s

HISTNG DISK BUFFER SPACE FOR PROGRAM.

We can do several things with this, which not only leave alot of dictionary space free, but, by

using techniques accessing VDP RAM and the screen image table, can visually enhance our programs, making them quick and professional.

WHERE ARE THE BUFFERS?

As the map indicates they start at HEX 2010 in the low memory expansion. It is easy enough to find the start of each 1K buffer (if needed) by examining the variable USE, a resident Forth word which contains the address of the block buffer to use next. Sitting at the keyboard load any screen then type:

USE ? Forth will reply with any of the following address's depending on where it is in its buffer cycle, but repeating the sequence using sequential loading of screens will uncover the whole cycle:

BUFFER1 from 8208->9235 BUFFER2 from 9236->10263 BUFFER3 from 10264->11291-BUFFER4 from 11292->12319 BUFFER5 from 12320->13347 BUFFER1 from 8208->9235

ie back to the start.
Forgive th decimal format!!!
This process by the way illustrates using Forth commands to discover how your machine works-doing it yourself will enhance you knowledge and exploratory techniques when it comes to debugging things.

It is important to know the limits of the area because over-running the end with data will overwrite what comes next in the system map, in this case the 99/4 support for Forth, contamination of which may spell an ultimely end to you computing session.

HOW TO USE THIS SPACE

Once delimited we can decide what to loading each screen. put in there.

REFER TO SCR# 161.

Firstly we may want to store a display which we will later write to screen using VDP commands when needed. This example illustrates the use of the !" (string-store) word. Unlike our evolving definitions of GET\$, this deposits

the strings to the address's you nominate, anywhere within the computer, but without a preceding count. I have kept my chart unneccessarily neat by allocating one address per line, I could have spaced it apart using only a start address, but it would have looked messy.

As the category listing is not a word within the dictionary, it must be loaded/compiled after booting the binary saved image of our completed program. Loading these routines it sort of like basic housekeeping before we get down to business of using the main program. Using multiple images like the above poses a few problems. Firstly before loading the buffer area has to be blanked with ASCII 32's or whatever is in there before the strings were stored will also be written to the screen when eventually used in the program. Secondly one has to be very accurate about not running them over one another during allocation. Thirdly one must decide into what part of the screen image table (see below) one dumps them. Finally, as they are loaded into the buffer area itself, ie the area Forth uses to sequentially Toad screens, we must tamper with the loading system to force loading through the same buffer address otherwise we can end up storing an image into one buffer area only to find Forth has caught up with us and re-used this space to load the next screen.

H

ď

C

÷

L

As

a

fc

RC

rd

Nc

φ×

ti ca

wh.

pr

an

foi

Da

VD

tal

CA:

the

pla

nur

SCI

Try

LOADING REPEATEDLY IN ONE BUFFER.

Dead easy really, all one has to do it to re-install the same address into USE before every screen load. For example if we decide we want to use 4000 bytes of our buffer area for screen images, we can continually point the value of 12320 into USE (see example below) before loading each screen.

The second way one can use this region is by allocating variable/consants there. In the above example the last 1000 bytes odd of the region is not wasted as we can use the space in the above manner. Eg 12320 CONSTANT NAMELIST would point to perhaps a 1000 byte buffer the start address of which is obtained in our program by writing

NAMELIST whenever the start address is desired.

REFER TO SCR# 162 FOR EXAMPLES.

VDP WRITING TO READING FROM SCREEN.

Refer to page 3 Chapter 4 to the VDP memory map (Videodisplay processor) Note at the top of the chart starting at address HEX ØØØØ is the annotation "GRAPHICS AND MULTI-COLOR SCREEN IMAGE TABLE". This stretches to HEX ØZFF a total of HEX 3ØØ or 96Ø bytes. Each of the 24 lines of your monitor consist of 4Ø columns in text mode. 4Ø * 24 = 96Ø !!!!

Line Ø=bytes Ø->39 Line 1=bytes 4Ø->79 Line 2=bytes 8Ø->119

Ìέ

he

d

es

е

lem

lee

s

Line 23=bytes 92Ø->959

Hence it becomes an easy matter when doing VDP reads or write to calculate where to put it or read from it.

Line start= ROW * 4Ø

As a useful exercise write yourself a routine to print a chart to pin on the wall next to you in the following format:

ROW START END eg:

Now, in the above category list example, as listed on scr# 161, every time we want to display this list we can define a word:

: CATLIST CATLSTADDR (start addr)
400 (col0 rowi0)
560 (no.to wrte)
VMBW; (write it)

where CATLSTADDR was allocated previously as:

10208 CONSTANT CATLSTADDR and VMBW is the command that stands for vdp multiple byte write. (see page 63 of glossary). It expects the VDP address in the screen image table to start the write to, in our case DEC 400, the address containing the data, in our case 10208 which we placed there on loading, and the number of bytes to write into the screen image table, in our case 560. Try it -it is lightening fast.

Similarly we can read things from various parts of the screen , place them into an address as a number/string which may be later used in our program. I will not elaborate on these today.

ASIDE ON PERIPHERAL ACCESS BLOCKS.

Contrary to what you are now thinking about MY disjointed thinking, this IS related to VDP WRITING. The memory map will indicate space left in VDP RAM for our PAB's for accessing files on non-Forth disks. This area starts at HEX 460 with a mammoth amount left of 800 bytes!! sheer waste!!.

REFER TO SCR 143

The second of th

This refers to setting up a file to access with no file name included, just a name lengh indicated by blanks. The user may want to access files, or name them himself to start files. No matter how proficient Forth programmers we become, we are not futuristic mind readers, so we have to have some way of placing the information into the exact soot of VDP RAM which will contain the File Descriptor of the file to be created/accessed.: If you consult Page 2 Chapter 8, you will see what I mean. The File descriptor starts at byte 10 +. By accepting a user input, placing it at a temporary address, and then writing it to the exact address of the start of the file descriptor, we are not locked in to fixed file names.

REFER TO SCR# 164

👫 y III Maria a tangan yang manggang menganggang ang manggang tang manggang ang kanggang manggang manggang ng manggang ang manggang ng ma

This screen illustrates an example of this. A buffer has previously been allocated named F-D, standing for file descriptor(surprise!). Assume the program has accepted what the user wants as a filename and validated its lengh etc. When we set up the PAB on scr# 163 we told the program how far from the start of the PAB region we were starting. The definition expects this offset, in my case 40 byte for one file, 84 for the other, and simply using the VMBW command write the user inputted name into the correct spot of the appropriate PAB, thus enabling proper disk accessing.

Another way we can use VDP RAM to take pressure off our buffer areas and dictionary is to dump images into VDF ram itself, starting from address 1400. Travelling into the never-never here, as intrapid explorers we discover a massive 8664 bytes of unused computer space. A minor problem here is that it is not easy (or it is -if I am wrong here let me know instantly!) to directly move bytes within various parts of VDP RAM. One has to move it byte by byte out to stack/address and back in to where one wants it, eg transferring a screen image up into the screen image table proper. Although many times faster than basic, this is disappointingly slow compared to multiple byte dumps from an ordinary address.

EXAMPLE OF USING VDRWRITES.

REFER TO SCREENS 165,6,7,8.

Those of you using Forth will be familar with the snappy, way one can flick back and forth across the screen using function 5 key. Similarly when using the Ti-Writer one can access all 80 columns of text by flicking back and forth. The next example does just that, simplified, and with no overlap between the screens.

First refer to screens 167 168. You can type in here what you like, for simplicity's sake I have just drawn a box with left and right hand labels. This will be loaded to the addresses indicate as described earlier in the article, within the buffe region.

FORCELOAD: updates USE with the same buffer address- the last in the series.

LOADPAT : blanks the buffer and via the same buffer installs the contents of scr 167,8.

into RAM.

DISP1 : does a vdpwrite to display first screen

DISP2 : as above for the second.

Having installed them we now need a way of flicking back and forth. This (Ø49) 436861(9-5) is handled by:

SWITCH

: which scans the keyboard If function S or left arrow is pressed screen! ie left hand is displayed and function D puts up right hand screen.Both leave a Ø flag for UNTIL. We can only escape by a Function 5 keypress which leaves 1 on the stack for UNTIL.

If no keypress is found a Ø is left by ?KEY and as it is an invalid choice it is DROPped after ELSE and a Ø left for UNTIL so we can try again.

LEFT/RIGHT CLS clears the screen, loads the pattern, starts the initial display with the left hand screen and SWITCH then awaits your instructions.

NOTE: Though I have included LOADPAT in the final definition you would not do this in a program. the strings would be loaded independently at the start of the program ready for permanent use. It is un-necessary to reload every

ANOTHER SCREEN DUMP.

Out of time to do much on VDF reads, so expansion on these will have to wait for another time. As an ultimate expression of VDP reading I include a full-screen dump.

REFER TO SCR# 169.

This can be included for example in the above examples as an option within a CASE statement to dump the current screen by detecting an appropriate key press. It only takes up 56 bytes of memory.

Thats all for this month. Who knows whats coming up next!

ADDRESS FOR CORRESSPONDENCE (keep those letters rolling in!!)

R. TERRY. 141 DUDLEY RD. WHITEBRIDGE. NSW 229Ø.

2245Ø (A.H)

```
SCR #161
 O ( B-CHEQUES - Category listing 14Mar86)
                                             ( CHECKED CLEAN)
             1.Shop Rental
                                15. Home Counc rates"
 3 10248 !"
                   Stationary 16.
                                       Water rates"
 4 10288 !"
             3.
                    Telephone
                                17.
                                       Gas"
                   Petty cash 18.
 5 10328 !"
             4.
                                       Electricity"
 6 10368 !"
                   Electricity 19.
             5.
                                       Insurance"
 7 10408 !"
                    Supplies
                                20.
                                       Loan"
             á.
 8 10448 !"
             7.
                   Equipment
                                21.
                                       Building"
 9 10488 !"
                   Books/Journ 22.Pers Cash"
            8.
            9.
10 10528 !"
                   Organisatn 23. Insurance"
 11 10568 !" 10.
                   After Hours 24.
                                       Med/Dental"
12 10608 !" 11.
insurance 25.
                                       Clothing"
                                       Liquar"
                                       Resturant"
                                       Sundries"
SCR #162
 O (B/CHEQUES - Buffer allotment (14Man94)
                                              ( CHECKED CLEAN)
 2 10208 CONSTANT CATESTADDR
                                     t adds CATLIST
                                                        560 5/te)
 3 10768 CONSTANT HELPSCRADER
                                    ( addr HELPSCREEN
                                                        720 byte)
 4 11528 CONSTANT BCARDBAL
                                    ( addr bcard balarry280 byte)
  5 11908 CONSTANT CHOBAL
                                    ( addr chque balarry260 byte)
 6 12088 CONSTANT ISTCATREC
                                    ( addr ISTCATREC
                                                         54 byte;
 7 12142 CONSTANT LSTCATREC
                                    ( addr LSTCATRES
                                                         54 byte:
 8 12196 CONSTANT NAME
                                     ( addr ACCOUNT
                                                         30 byte)
 9 12226 CONSTANT ACCNT#
                                     { addr ACCOUNT#
                                                         15 byte)
10 12241 CONSTANT BANK
                                     1 addr BANK
                                                         20 byte)
11 12261 CONSTANT SUBURB
                                    t addr SUBURB
                                                         20 b/te)
12 12281 CONSTANT ADDRESS

    addr ADDRESS

                                                         30 byte)
13 12311 CONSTANT PHONE
                                    ( addr PHONE
                                                         15 byte)
14 12326 CONSTANT CARDDATA
                                    ( addr CARDDATA
                                                        210 byte)
15 12536 CONSTANT TOTAL
                                    ( TOTAL ends 13316 780 byte)
SCR #163
 O ( B/CHEQUES - cheque
                          PAB 14Mar86) ( CHECKED CLEAN)
 2
 4 HEX
       PABS @ 4A + BUFR 17A3
       FILE CHEQUEFILE
 8 : CHQ# CHEQUEFILE SET-PAR RLTV
          INTRNL UPDT FXD 4E REC-LEN
10
          F-D" DSK2.
7 7
12 DECIMAL
13
14
15
```

the standing theory is a state of highly of the legal photo-material in the

```
SCR #154
 O ( B/CHEQUES - New Disk Format
                                    94pr86)
                                                     ( STACK CLEAN)
                                      ( expect offset to start F-D)
 2 : PUTINYEAR
                  F-D
                                      ( addr with file descriptor )
  4
                  PABS @
                                      ( @ PAB addr put offset on )
                  ROT +
                                      ( top 60=bank 84=cheque
  5
                  15 VMBW
                                      ( write file descriptor here)
                            ;
  8
 9
 10
 11
 12
 13
 14
 15
SCR #165
 O ( VDP WRITE PRACTICE
                                     4Jne86)
  1
  2 : FORCELGAD 12320 USF 1 ; ( load last buffer addr to use
  4 : LOADPAT
                                 ( expects nothing
                8208 2000 BLANKS ( blank out ist 2000 bytes buffer)
  6
                169 167
                                 ( loop indices
                                 ( start loop
                DO
                                 ( force load to same buffer
  8
                   FORCELOAD
                                 ( load screens 167 168
  9
                  I LOAD
 10
                LOOP
                                 ( end loop
                                 ( leaves nothing on stack
 1.1
               ;
 12 : DISP1
               8208 0
                                 ( addr left hand part of display)
                                 ( on screen write 320 byles
               520 VMBW ;
 13
                                 ( addr right hand part of display)
 14 : DISP2
               8690 0
 15
               520 VMBW :
                                 ( write to screen
SCR #166
  0
  1 : SWITCH
               BEGIN
                    PREY DUP DUP to leave 4 copies of key
  2
  3
                         8 = SWAP
                                     ( was it 6=left arrow:f'n 5 )
  4
                         9 = MAX SWAP ( was it 9=right arrow:f'n d)
                                      ( was it function 5=leave
                                                                 )
  5
                        14 = MAX
       IF
            CASE 9 OF DISP2 O ENDOF ( if 9 swap to right 1/2 scrn)
                  3 OF DISP1 O ENDOF ( if 8 swap to left 1/2 scrn)
                               ENDOF ( user wants to quit, flag=1 )
                 14 OF 1
  8
  9
                    ENDCASE
               ELSE DROP O
                                      ( invalid key, drop copy, 0flg)
 10
 11
               THEN
 12
                                      ( leave if stack top =1
               UNTIL :
 13 : LEFT/RIGHT CLS LOADPAT
 14
                                     ( display left/right halves )
                  DISP1 SWITCH ;
 15
```

```
SCR #167
  O ( PRACTICE)
  2 8208
            [ " -----
  3 9248
            !" - LEFT HAND
  4 8288
            10 -
  5 8328
            1.0
  6 8368
  7 8408
  8 8448
            ; n
  9 8488
            i n
 10 8528
 11 8568
            1 " -
                                                        _ #
 12 8608
            1 11
            1.4 ...
 13 8648
 14 8688
 15
SCR #168
  O ( PRACTICE)
  1
  2 8690
            1 15
  3 8730
                                         RIGHT HAND
  4 8770
           į a
  5 8810
           1.0
  6 8850
           F 1F
  7 8890
           1 12
  8 8930
  9 8970
           į H
 10 9010
 11 9050
           1 11
 12 9090
           4 11
 13 9130
           t # _
 14 9170
 15
SCR #169
 0 ( B/CHEQUES - Screen dump
                                    20May86) ( B: 56 STACK CLEAN)
 1 : SDUMP
                                       ( Expects nothing on stack )
 2
                  SWCH CR
                                       ( PAB will point to printer )
 3
                  959 0
                                       ( 960 bytes on 1 screen
                  \mathbf{p}_{\mathbf{Q}}
                                       ( start loop
 5
                     I VSBR
                                       ( read each byte of screen
                                       ( and send to printer
                     I 1+ 40 /MOD
                                       ( I+1/40=result + remainder )
 8
                     DROP
                                      ( leave only remainder
 Þ
                     0=
                                       ( 1=even div=ie at end line )
10
                     ₹F
                                       ( if are at end of a line
11
                     CR
                                       ( perform a carriage return )
12
                     THEN
                                       ( and go on
13
                  LOCP
                                       ( end loop
14
                  UNSWCH
                                       ( turn off printer
15
                                       ( leaves nothing on stack
```

and the same reading to a mark transmission of the first states of the first of the same o

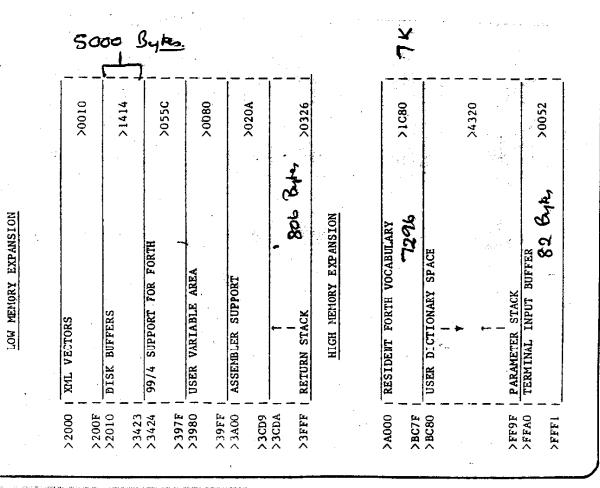
و با الرواق المراق المراق المراق المراقع المر

VDP MEMORY MAP

HEX		•	HEX ADDR
ADDR	CDADUTCE S		>0000
>0000	GRAPHICS &	i	20000
1	• •	· .	!
\00EE	SCREEN IMAGE MODE		
>02FF	TABLE > 300 SCREEN		
	SPRITE ATTR I IMAGE		
>037F		BIT MAP	
>0380	COLOR TABLE >3CO	COLOR	
>039F	>20 42 647	TABLE	
>03A0	UNUSED >20 1		
>03BF	32 bytes	>1800	
>03C0	VDP ROLLOUT AREA >20		
>03DF			
	STACK FOR VSPTR >80		
>045F		[
>0460	PABS ETC >320 800 Butes		
>077F (•		İ '
>0780	SPRITE MOTION TABLE >80		
>07 FF	128 byles		
>0800	PATTERN DESC TABLE		
İ	SPRITE DESC TABLE	j.,	
	0 - 127 >400		
>OBFF		,	
>0C00	128 - 255 >400	· •	
>OFFF			
>1000	FORTH'S DISK BUFFER		
>13FF	(4 SECTORS) >400	<u> </u>	İ
>1400	UNUSED		
	>21D8 8664		>17FF
	90-1	BIT MAP	>1800
	Bytes	SCREEN	\
		IMAGE	!
		TABLE	İ
	• •	>300	>lAFF
		PABS ETC. >CO	>1B00->1BBF
		STACK FOR VSPTR >40	>1BCO->1BFF
		FORTH'S DISK BUFFER	>1C00
	· 	(4 SECTORS) >400	>1FFF
		BIT MAP PATTERN	>2000
		DESCRIPTOR	1 /2000
·	L	TABLE	
>3507	, <i>,</i> 	>1800	>37FF
	DISK BUFFERING REGION		>3800
	FOR 3 SIMULTANEOUS	LIST >80	>387F
i	DISK FILES		>3880
	DISK FILES >A28		/3000
i	•		ı >3909
	2600	· 	
	* .⁺		_
\1 000	[
>3FFF	l	>626	>3FFF

CPU MEMORY

>0000	CONSOLE ROM	
i		1
>1FFF	<u></u>	SK OF DISK SUCTERS
>2000	LOW MEMORY EXPANSION	TO ON BIOS
1	LOADER, YOUR PROGRAM, REF/DEF TABLE	LI BIN OF DISK YOU
>3FFF	<u> </u>	U 5ku
>4000	PERIPHERAL ROMS FOR DSRs	<u>-</u> .
		1
>5FFF		1
>6000	UNAVAILABLE - ROM IN COMMAND MODULES	- [
1		T .
>7FFF		_1
>8000	MEMORY MAPPLO DEVICES FOR VDP, GROM, SOUND,	
1	SPEECH. CPU RAM AT >8300-83FF	1
>9FFF		<u></u>
>A000	HIGH MEMORY EXPANSION	1
!	YOUR PROGRAM	I be seen of
1	YOUR PROGRAM	irzyk.
\EEDD		(ا
>FFFF		



THE INFORMATION PAGE

COMING EVENTS

tan iking dalam dalam iking iking dalam da

Next Committee meeting: Tuesday, 1st July at 6.45pm Next General meeting: Tuesday, 8th July at 7.00pm 1st Australian TI-Fair, Malvern Town Hall (Melbourne): Saturday, 14th June 12 noon to 6.00pm

AGENDA for JULY GENERAL MEETING

PLATO Courseware demonstration Communications demonstration Monthly disk/tape demonstration

GARY JONES' EXTENDED BASIC GROUP

Sprites

AL WRIGHT'S BASIC GROUP

Loops (Branching, For...Next) Handling character Strings Basic mathematic functions

All meetings are held at the Warners Bay High School

Annual subscriptions to the Group cover the period 1st July to $3\emptyset$ th June following year.

j

,

People interested in joining our Group are asked to contact:

The Secretary, Hunter Valley 99'ers Users Group, 4 Arcot Close, TARRO. N.S.W. 2322 AUSTRALIA phone (049)662602

The annual subscription is as follows:
Residents of Australia...\$20
Overseas residents.....\$40 (Airmail)
\$30 (Surface Mail)

Back issues of our Newsletter are available for \$1.00 plus postage