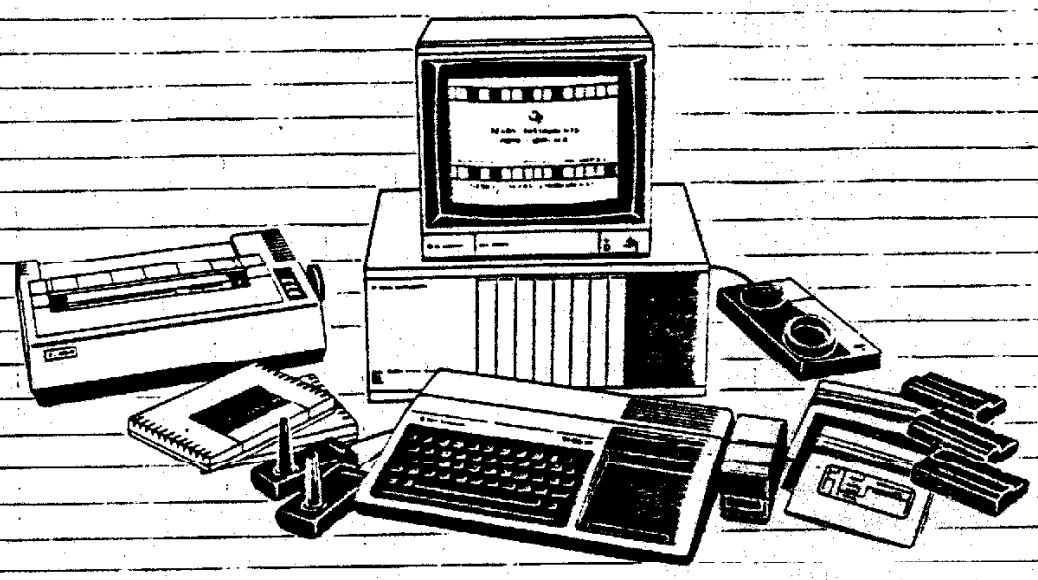
HUNTER VALLEY 99'ERS NEWS

TI 99/4A

VOLUME.1 NO.1 JUNE 85.



YOUR COMMITTEE 1985

A.WRIGHT	PRES.	PH.468120
P.COXON	SECT.	PH.751930
B.RUTHERFORD	TRES.	PH.498184
A.LAURENCE	LIBR.	PH.486509
S.TAYLOR	EDIT.	PH.487076
A.BYRNE	TECH.	PH.498520
T.MCGOVERN	TECH.	PH.523162
B.MAC.CLURE		PH.437431
B.WOOD		PH.662307
D.WINTON		PH.591882

MEETING DATES

BEGINNERS BASIC EACH TUESDAY
ASSEMBLER LAST TUE EACH MONTH
MONTHLY MEETINGS 11/6/85
9/7/85
13/8/85
10/9/85
ALL MEETINGS COMMENCE AT 7PM.
FINISH WHEN THE LAST MEN FALL
PHONE COMMITTEE FOR MORE DATA

NEWSLETTER OF HUNTER VALLEY TI99/4A USERS GROUP

TEXAS INSTRUMENTS

EDITORIAL

Welcome to the HUNTER VALLEY 99'ERS USER GROUP'S first news letter.

The Hunter Valley 99'ers user group was formed to replace the Newcastle Regional Group of TISHUG. It was felt that the needs of local users could be better served by an autonomous regional group.

We are currently in the process of establishing formal ties with other TEXAS INSTRUMENTS home computer user groups in Australia and overseas and we would welcome approaches from groups whom we either may not be aware of or have not been contacted.

The basic function of a user group is for people with similar interests to exchange knowledge, information, ideas and experiences with the aim of increasing each persons ability to use their computer. Within the HV.99'ers people range from beginners to advanced programmers and users. All have information, ideas etc which can be input to the group. The input can be through the regular club meetings or this newsletter, which is an ideal forum in which the information can be presented. The group will be actively encouraging members to contribute to the newsletter in the form of articles, information and/or ideas for articles.

The task of contacting all TI 99/4A users in the region is a high priority for the group to make them aware of our existence and offer the support of the group.

ACTIVITIES

Each of the special interest groups formerly within the Newcastle Regional Group will continue as previously but within the HV 99'ers.

The Basic programming group will continue to meet each week on Tuesday nights under the guidance of Terry Jones.

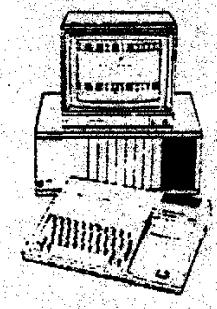
Advanced Extended Basic with Tom Matthews will continue at the [illegible] monthly meeting.

[illegible] Language group will meet on the last Tuesday of each month.

A suggestion has been put forward that a six month beginners course be run. The aim being that at the end of the course those people going through it could join the Basic Programmers Group if they wanted to. One of our members has offered to run this course on two nights a month for the six month period. If you are interested contact one of the committee members at the next monthly meeting.

MAITLAND.

A matter of great concern to the group is the cessation of the TISHUG Regional Group in our neighbour city Maitland. Mr Jim Threadgate one of our stalwart members from Maitland has begun the task of finding the former members of that group. The aim is to either assist to get it operating again as previously or perhaps as a sub group in the HV 99'ers.



PEERING INTO THE MURKY FUTURE.

It could be said by some less well informed people, notably TI 99/4A bashers, that now is somewhat less than an ideal time to be forming a user group for the TI 99/4A. The machine has vanished from major retail outlets and only a handful of retailers carry TI 99/4A peripherals, softwares and literature. In essence a gloomy picture.

The Hunter Valley 99'ers User Group is very confident of the immediate foreseeable future of the 99/4A. A lot of Third Party manufacturers are still supporting the machine in the U.S.A. Several Australian user groups or their members are currently marketing or are about to market add ons for the 99/4A.

The 99/4A is an excellent home computer and its demise was not brought about by the poor quality of the machine but rather by poor

marketing skills and some not so very good ideas about Third Party Software.

The TI. offers features still not found in the "newer" generation of home computers. The Basic is extremely user friendly; in particular the screen control commands in comparison to some home computers which claim a victory over the TI. One recently released home computer (the marketing strategy for which looks a bit like some hamburger marketing) heralded the "new" commands of trace and untrace! The user written subprogram feature of Extended Basic is not offered in far more expensive machines than the TI.

In the Hunter Region the TI 99/4A is a long, long way from dead and the HV 99'ers aim is to ensure that an unjustified and untimely end does not befall it in the Hunter Region.

So if your TI. has taken up residence in the cupboard then perhaps now is the time to get it out and brush the dust off and join us in our adventure into T.I. land.

A. WRIGHT. PRESIDENT HV 99'ERS

****
* SECRETARY'S NOTES *
****

Welcome to the first edition of the HUNTER VALLEY 99'ERS magazine. If you have any questions, enquiries, handy hints etc., you will find names and phone numbers to get in contact with on the front page.
We have contacted all user groups throughout Australia and overseas, to bring as much info as possible to our readers from Gosford to the @ueensland border.
Our main meeting headquarters may have to be changed within the next few months as the building has reportedly been sold, we dont at this stage know who has made the purchase, or for what the building will be used, but we will making enquiries. In the meantime we will be looking for a new home for the group. Any suggestions would be appreciated. Areas we will be looking at are Warners Bay, Cardiff, Glendale, Teralba.

BOOKS ON COMPUTING.
We have 3 main suppliers in our area:
1. CO-OP BOOKSHOP
2. CLUB LIBRARY
3. LAKE MACQUARIE CITY LIBRARY who have a good range of books from Basic to Forth and Pascal, with existing titles such as "Computer programming for the complete idiot"!!
STEVE TAYLOR, JOE WRIGHT and myself checked out the local Computer Show in April. It was more business related this year, very few home computers. There was one outstanding demo thanks to TELECOM'S new Computer Phone with a word processor, spread sheet, phone directory, archival storage together with Sinclair Basic through Microdrives, and direct entry to VIATEL for the cost of a local call plus a small charge on top of this covers you for such things as what plane is arriving in Darwin, the latest banking charges or what's new in computer games. You will also be able to get into VIATEL from your TI 99/4A , but the Modem you will need for this is what they call a 12/75, this is 1200 baud rate send, 75

Telecom VIATEL

England with a stop at Singapore.
Certainly.

receive, unfortunately this is expensive at present.
If you would like to know more on VIATEL and Computerphone, you'll find a very good article in Australian Personal Computer Jan. 85. Running out of memory but before I go if you require any info in ret. the meetings., Please remember, phone at any reasonable hour before 9PM.
PETER COYNE.

A LOOK INSIDE YOUR TI-99/4A

ARTICLE BY GARRY JONES HV99'ERS

TMS 9900-16 BIT MICROPROCESSOR.

The C.P.U. or Central Processing Unit is the Master Control section of the Computer System. It ties all the other devices together, keeps a reference as to where its up to with the Memory, ensures that the correct external device is selected also. This section is where the calculation and processing of data is in accordance with instructions.

48 MHz CLOCK.

The "clock" is a crystal which ticks at 48 million times a second or 48 megahertz and is the source of pulses for the overall timing whichdetermines the speed of operation.

CLOCK DRIVER.

Sets the different pulse rates and keeps them synchronised. Its like the big brother of the clock, it looks after all its functions.

TIMING AND CONTROL LOGIC.

Ensures the correct mode of operation, monitors all pulses for correct timing and accepts and controls all incoming signals. Its job is much the same as a "Foreman" in a large Factory.

SYSTEM MONITOR & GPL INTERPRETER

This section has the Main part of the ROM for the C.P.U., it also performs most of the system "house keeping duties". These duties like a house keeper make sure every thing is in the right place and the Computer is operating correctly.

The GPL interpreter or "Graphic Programming Language" is the interpreter between the Basic and Machine Languages.

GRAPHICS ROM & BASIC ROUTINES.

The rest of the ROM is located in this section and is used for the Graphics generation and display on the screen. Also the Basic Language programme interpreter is accessible from here.

VIDEO DISPLAY PROCESSOR.

Generates the Video and audio signal for your colour Monitor or T.V. Without it you would not know what the Computer was doing. The VDP isalso the controller of the working RAM, it addresses, stores or pidgeon holes the data being sent from the CPU.

BASIC PROGRAMMES & DISPLAY MEMORY.

This is the RAM or working memory section of the system. All stored programme instructions are carried out from here. This also includes all character displays on the screen are printer, but not Graphics display.

SCRATCH PAD RAM.

As the name implies the scratch pad is where the working out is performed before the data is stored in the memory or displayed on the screen etc. This is also where the C.P.U. keeps track of functions such as where the cursor is on the screen, the stack pointer for GOSUBs and interrupts. Its like the Computers scribble pad, just for noting items of data not needed to be stored in the memory.

16 BIT TO 8 BIT DATA BUS CONVERTER.

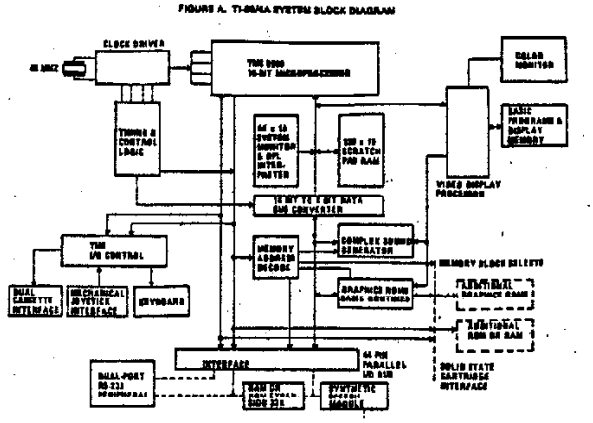
The main function of the converter is to divide or combine a 16 bit word from or into 2*8 bit bytes for using in the external devices of the Computer.

MEMORY ADDRESS DECODER.

Because of the complex addressing of memory locations in the processor, this section decodes and controls the addressing of memory data being handled by the processor. The decoder is similar to a signal box in a large railway goods yard.

CONT

A LOOK INSIDE YOUR TI-99/4A



SOUND GENERATOR.

This generates the frequencies to be converted into sound via the VDP to the monitor or T.V.

TMS I/O CONTROL.

This device controls input data from the external device interfaces.

DUAL CASSETTE INTERFACE.

Controls which cassette is used, also allows the saving and loading of data with a cassette player.

JOYSTICK INTERFACE.

Interprets which joystick is being used and in which direction the joystick is positioned.

KEYBOARD.

A data input device.

INTERFACE.

I/O port for the connection of external hardware, eg speech synthesiser, peripheral expasion box.

COLOUR MONITOR.

The displaying device for the Computer.

END

TI-FORTH

TI/FORTH was placed in the PUBLIC DOMAIN by TEXAS INSTRUMENTS on December 21st. 1983. Unfortunately however TI-forth did not undergo the complete testing and evaluation normally performed by TEXAS INSTRUMENTS prior to distribution. therefore there is the possibility that both the disk and the manual released by T.I. "as is" may contain errors or omissions. Hopefully all the bugs will be detected eventually.It is therefore in your interest to pass onto us any bugs that you find and we will publish them so that all TI-FORTH Users can benefit. In a sense all TI-FORTH users will have to become T.I's quality control. An example of the debugging can be found in the March '84 edition of MILLERS GRAPHICS "SMART PROGRAMMER" which shows how to modify the printer section of FORTH to fix the syntax error.

Anyone interested in learning to programme in FORTH would be well advised to read the series of articles published in "THE SMART PROGRAMMER" as they contain a wealth of information which will help you avoid some of the pitfalls encountered when starting out. HUNTER VALLEY 99'ERS subscribes to this publication and it is available from the club LIBRARIAN Al Lawrence. Please note only one copy is available and it is extremely popular so please dont "HOG" it

The really great news about TI-FORTH is that it is a transportable language, which means that if you learn TI-FORTH on your trusty old 99/4A it is relatively easy to transfer what you have learnt on your 99/4A to any other computer capable of running FORTH eg. APPLE or COMMODORE 64 (Thats if you could bring yourself to using them!). The reason for this is because FORTH sticks to a well defined standard (or two!). most versions of FORTH are written to either the FORTH 79 or the fig-FORTH standards and fortunately the difference between the two is minimal. Therefore a programme written on your 99/4A can be easily

CONT NEXT PAGE

TI - FORTH

adapted to another machine.

TI-FORTH is an extension of the fig-FORTH dialect of the language and the minimum system requirements needed to run it on the 99/4A are at least one disk drive, 32K and E/A. If you dont have E/A dont despair as there are programmes available which enable you to run TI-FORTH using EXTENDED BASIC. (Contact EDITOR for further info.)

If you intend making use of the 64 column display available in TI-FORTH it may be wise to consider investing in a monitor so as to avoid eye strain !!!.

TI-FORTH can be described as a Threaded Interpretive Language, and a basic analogy which may help you to understand what this means is to imagine a string of beads that can be threaded in various ways to create larger pieces of jewellery from the original set in order to suit different needs and occasions. A person programming in TI-FORTH uses the basic precompiled words of the language to thread together other words that will be of use over and over again. The programmer then orchestrates them all together into still higher level words that can be used to create programmes.

The FORTH language itself is not a new language, it was developed in the Sixties by Charles H Moore. And according to the creator FORTH is many things at the same time. It is a high level language, a set of development tools, an assembly language and a "software design philosophy". However one of the great attributes of FORTH is the small size of itself and the programmes produced using it. This is brought about because FORTH crams an amazing number of powerful words into a small space. Naturally this results in a very fast language. Texas Instruments states that TI-FORTH runs an execution speed within a factor of two of Assembly Language for most applications.

TI-FORTH gives you access to the entire 99/4A with the added advantage that it is far easier to learn to programme than the dreaded "Assembly Language". Anything that can be done in Assembly Language can be also done in TI-FORTH. Naturally with all its good points Forth should also have its darker side, the more frequently heard complaints are that it is unreadable, the reason being that forth is a word orientated language similar to LOGO in that an individual defines his or her own words as a procedure and then uses that word to help define the next word and so on. the defined words are then added to Forth's vocabulary. Another complaint is that you cannot produce stand-alone executable files as you can with compiled languages. Most FORTH programmers readily admit that it is a difficult language to master but this is more than compensated for by the greater control over the size and speed of the programmes brought about by the friendly interactive environment provided by FORTH, therefore it is relatively easy to get started writing FORTH programmes.

If you are interested in learning TI-FORTH and have the minimum system requirements a copy of the TI-FORTH disk and manual can be obtained through the club for the cost of the medium. Hopefully if enough people show interest in the language regular classes or discussions may be organised. in the meantime suggested reading material and contacts are listed below.

"STARTING FORTH" by Leo Brodie
"INVITATION TO FORTH" by Katzan

Starting Forth is available from Lake Macquarie Libraries and I'm sure both books are available through the UNI Bookshop.

Also a catalog of all available fig-FORTH books can be obtained by writing to "COMPUTER REALM" P.O. BOX 103 Camberwell. 3124

Further information and assistance can be obtained by contacting the Sydney Forth Group C/O Peter Tregeagle. 10 Einda Ave. Yowie Bay. 2228 ph.(02) 524 7490) also The Forth Interest Group. POB 1105 San Carlos. CA. 94070 U.S.A. Ph.9628653.

STEVE TAYLOR.

END

EXTENDED TUTORIAL

BY TONY MCGOVERN HV99'ERS

WELCOME to the first Extended Basic Tutorial from Funnelweb Farm to appear in the new HUNTER VALLEY 99 NEWS. This will be a continuation of the series of articles which appeared earlier in the TISHUG Newsdigest, but the HV99 NEWS will now be the primary source. The series will go on in the same vein as before, intended neither as an elementary course for raw beginners nor as a reference treatise. It is meant for the interested user who is willing to put some effort into understanding how Extended Basic works in order to make best use of it, and wishes to develop a feel for how the machine actually goes about its business. Plus assorted ravings, news items, and ramblings on.

Some copies, and disk-files, of previous Tutorials will be available at HV99 meetings for newcomers or ex-TISHUGers who don't have all the previous ones in their Sydney file. The Tutorials may even show up in User Group News letters around the world. I have seen one in the TI*MES from the UK. Now HV99 members will be the first to see them, for whatever that is worth. If other User Groups wish to reprint these Tutorials, please get in touch with me, either directly or via the HV99 group, so that you can get a corrected and updated version on disk in the user group spirit of exchange. I usually do an edit on the file after it has appeared in print to correct the little goofs which hide so cunningly before printing, and sometimes to clarify, correct, or extend what had appeared in print. The printed version has not always been precisely what was on disk either.

Well, where do we go in the future ? So far the series has had a detailed look at user SUBprograms and the ACCEPT AT statement, the two most powerful features of TI's Extended Basic, and also at the prescan switch commands lurking in the U11Ø manual addendum. For the next few sessions we will continue with topics which are of immediate relevance to console-only users, namely squeezing programs to fit in memory, and extracting maximum speed from XB. Please let me know of areas you would like covered. My policy so far has been to concentrate on those parts of XB which are especially powerful, not already included in console Basic, not well documented, and not as widely appreciated as they should be. The next few tutorials will be on getting the most into and out of the machine while using XB. On the other hand I can see no point, and have even less interest in writing about, say, SOUND or SPRITEs from the very beginning, as these are fairly well documented in the manuals and the subject of many books and articles. That isn't to say that subtleties in using them won't come up from time to time.

There has been a gap of a few months in appearance of Tutorials, mainly due to pressure of work. The time spent on the TI-99 has been almost entirely devoted to Assembly language programming, much of it in association with XB, and this will provide some real substance for future HV99 News articles, either in this series or separately. One of these projects has been to get TI-Writer running from XB. Why do that ? Well... TI have always been good guys in that most serious disk software can be backed up on disk as often as needed, but they were of course relying on the infamous cartridge GROMs for protection and exclusion of others. Now cartridges are a lot less fragile than disks, but they can die too. So I don't want ever to have to suffer Imagic Australia's less than impressive service and/or rapacious pricing policy if our TI-Writer module ever claps out. May save some module swapping on occasion too. Yes, we do have a spare XB module!

Of late I have been working with Microsoft Basic and Turbo Pascal on CP/M machines with Z-8Ø processor in science laboratory applications. I

must say that the more I see of Microsoft Basic the more I regard it as a cancer that should have been eradicated years ago when computers grew up to have more than 8K of memory. It is only now with their Apple Macintosh version, judging that from reading magazine hype, that they have at last surpassed the level of expressiveness that XB had years ago. TI did a lot of things to screw up this machine, some of historical origin, some quite intentional, but it takes coming from the engineered TI-99/4a file and device handling system to CP/M to make you realize how weak and primitive CP/M is in this area. On the other hand Turbo Pascal almost makes that Swiss straight-jacket feel comfortable, and even CP/M doesn't seem so bad with Turbo. An excellent product at a realistic price that makes one realize that pirates are only the minor league of brigands in the software business. We can only dream that someone will bring out a native code Pascal anywhere near as good as Turbo for our machine. The TI P-code version is most unattractive at Imagic's past and present exorbitant price (eight times that of Turbo). Now that more compact consumer type products are replacing the massive PE box, P-code cards will fade into oblivion. Maybe I'll be proved wrong but I get the feeling that Imagic Aust. is the sort of outfit that would rather use things like that as hard-fill in a swamp than let committed users have them at a realistic price.

It is also another example of how TI had this death-wish to hobble the most powerful micro-processor in any home micro available here, with interpreted languages. Shed a tear for TI-99 Basics with their two layers of interpretation (Basic and GPL), on top of working indirectly from the byte oriented VDP memory and GROMs.

Enough raving on for now and on to the real business. Let's now look at how to face up to that 'MEMORY FULL' message. This even comes up when you have memory expansion with a total of 48K of RAM in various guises. Programs always seem to end up needing more memory than is available! I do feel some unease in discussing this topic as many of the things that are done in compacting programs can only be regarded as poor program practice otherwise, as they make the code obscure and difficult to modify or develop further. The other great trade off that must be considered when scrunching programs is speed of execution. Given an equal level of skill in program writing, coding for speed usually results in a longer program than would otherwise be written. Perhaps the easiest example to see is unrolling of a short loop which is repeated a fixed number of times. A FOR-NEXT loop gives compact code but carries a penalty of the loop overhead which could be avoided by writing out the contents of the loop the appropriate number of times. The subject of coding for speed will be taken up in detail in later Tutorials, and speed sacrifices with compacted code will only be noted in passing. The richer the language the more opportunities there are to optimize code one way or the other. Console Basic offers many fewer ways to do this than does XB and is much less fun.

At what stage should you bother trying to make your code compact ? Remember that XB can only OLD or RUN one program at a time, so apart from loading time from cassette, or disk space, there is no reason at all to scrunch a program that runs in the smallest memory it is intended to run on. Most users with disks now have the 32K memory expansion, so this means the bare console. Minimem Basic programs to store in the module's RAM are the only ones you have real incentive to make smaller still. Unless you know from the start that you are going to run short of space because of large arrays of numbers, or a need for maximum string storage room, be expansive -- document your program thoroughly with REMs, use lots of SUBprograms, use obvious explanatory names for variables, avoid reusing variables for unrelated uses ..... and then you run out of room.

Now first of all a program has to be short enough to load. This is purely a function of program length. Next it has to be able to complete prescan when RUN. For prescan to succeed there must be enough room left over after the prescan for variable pointer and subprogram tables to be set up, and room set

aside for numeric values, at 8 bytes per number. String variables are not assigned space until it is actually required, so it is possible for a program to crash later because it can't find enough room for strings. The well known hiccupping of long Basic programs occurs while Basic scratches around to reclaim string space when it has run out of new space. XB does it too, but it is a lot faster at 'garbage collection'. Now let's look at how to squeeze programs in, starting with things that affect the program length only.

The most obvious thing to do is to remove REMs from your program. I would suggest that this be left till later in the development process as you put them there in the first place to help. At the least keep some for now. If you have been following earlier Tutorial advice to use lots of clearly named subprograms then you don't need many REMs. For the same reasons you should not abbreviate subprogram names beyond recognition at this stage. Basic as an interpreted language, where the source code is also the run-time code, has this problem that commentary and explanation are not eliminated by a compiler or assembler and compete for memory space with the executing program. One way round the problem is to restore REMs to a file copy after intensive development is over, even if it does make it too long to RUN. The REMS can always be removed later.

Now it's time to look at what makes an XB program as long as it is. To get started let's look at two very short programs to clear the screen.

```
100 CALL CLEAR
```

Before entering this clean up the machine with NEW and SIZE it. Then enter this program and SIZE it again. The difference will be the length of the program 13928-13914 = 14 . I will mostly quote SIZEs on the basis of a console only machine for simplicity, but there are some interesting differences. With memory expansion XB lists high memory and stack separately, and ignores low memory altogether. XB stores the program and numeric variables in high memory (24K), while the stack - 12K of VDP memory - contains variable descriptions, subprogram details, PABs, and the string storage space. This ALL has to fit in 14K of VDP RAM with XB/console only. Console Basic doesn't use memory expansion for Basic at all. Now try a second program which does almost the same thing

```
100 DISPLAY ERASE ALL
```

and SIZE it. Only 9 bytes now ! Although the LIST of this second program is longer, the computer thinks it is shorter. Consult your XB manual p40 where you will find all three words DISPLAY, ERASE, ALL are listed as reserved words, as is CALL but not CLEAR. Reserved words are treated differently -- when you enter the line they are recognised and "tokenized" as one byte symbols with ASCII values >127. 'CLEAR' takes 7 bytes, one the token for a string without quotes, one for a length byte, and 5 for the string itself. Why use tokens ? For one thing it shortens the program length, and also makes it easier for the interpreter to recognize them when the program is running. XB's range of tokens is very limited and built-in subprograms are the way XB gets around this.

Now you don't have to take my word for this. If you have an expanded system you can write programs using CALL PEEK to explore stored programs, or better still use the E/A DEBUG (reassembled as uncompressed object code so the XB loader can handle it) for a quicker look. With console XB you can at best get an indirect insight by entering <CTRL+various keys> in a REM statement and LISTing that. Be careful, you can crash the computer in ways wondrous to behold that way. Someone forgot to tell the computer not to try to turn token values back into reserved words when LISTing REMs. Ever notice when writing file specifications that keywords that do extra duty elsewhere LIST with the extra space, but the others do not. EASY-BUG in Minimem also allows you to look directly into VDP RAM or cartridge RAM to see Basic programs in their internal state.

In TI Basics, unlike those which store programs as ASCII files, the line number is always stored as a 2 byte integer, and it makes no difference to program length to use line #1 or line #10000. Try various

line numbers in one of the examples above. If you are peeking around in the program, don't expect to find the line number at the start of its program line. It is in a separate table below the program, and each 4 byte entry has the line number followed by the location of the line itself. The line # table is sorted into order, but new or edited lines are always added to the lower address end of the program block. The program lines themselves are preceded by a length byte and terminated by a null (>00) byte. I won't go into it here but you can use this general information to interpret the various time delays when you edit a line or enter a new line.

From this you can see that there is a 6 byte overhead associated with every new line number. Now enter the program lines above as lines #100 and #200 and SIZE. Next combine them as a single line

```
100 CALL CLEAR :: DISPLAY ER
ASE ALL
```

and SIZE again. There is a saving of 5 bytes. The reserved word "::" has cost 1 byte, but 6 bytes have been saved by having one line fewer. Now if you scrunch a 500 line console Basic style of program into 200 XB multi-statement lines you have gained 1500 bytes. Of course you can't do this to every line because line numbers, as well as being line editor markers, are also where GOTOs and GOSUBs go, so you will usually end up with a few short lines you can't condense. FOR-NEXT loops work perfectly well within or across multi-statement lines. The use of prescan switch commands is costly because you end up with !P+ and SUBEND on separate lines at the end of each subprogram so treated. Still, it's usually worth doing even though a long program may have several hundred bytes tied up in prescan switching. In desperation at the end you can always remove prescan switches starting with the shortest subprograms.

How much room does a variable take up ? Take a simple numeric variable. There are 8 bytes for the radix-100 floating point form that both TI Basics use for all numbers (they even do 1+1 to 14 significant figures every time - another reason they are slow). Next the interpreter has to be able find where this value is stored so there's 2 bytes for a pointer to the value, and 2 more to point to the name associated with this value. Further it has to record the nature of the variable, whether it is numeric or string, simple or array, DEFed or normal. Also in a Basic language which allows long variable names a length record is also likely, though not absolutely necessary. All told there is a practical minimum of 14 bytes of overhead for every simple numeric variable.

As I have noted in other connections in this series, TI in its self-defeating secretive way, never openly specified the details. TI Basic is most likely highly consistent in this from model to model, because any console can be called on to work with separate E/A or Minimem Basic support utilities such as NUMREF. On the other hand each XB module contains its own set of support utilities, and only has to be internally self consistent. For instance it is not possible on the basis of TI's published data (XB, E/A, or Technical manuals), for an assembly routine to determine the dimensions of an array passed in by CALL LINK without its being told explicitly, or for that matter even to find where a simple variable is stored. Now it is done from time to time, but TI never made any commitment to ensure that such procedures would work with all XB modules. Maybe they do. Only TI knows for sure. Then again TI lost big while Apple and IBM make lots of money being open about their machines, though Apple seems to be developing more secretive ways as it gets older and more arrogant.
Time for some little program experiments again. Enter the miniscule program

```
100 A=0
```

Before you do anything else work out how many bytes this uses. The answer is 11. In accepting the line the editor has already figured 'A' for a variable (because it starts with an allowable character) and not a reserved word and it is represented exactly as it occurs, no token involved. On the other hand it doesn't yet care that '0' is meant to be a number and treats it as an unquoted string. If it isn't

an honest number, say 2N, it will only find out later when it RUNs and tries to convert it to a floating point number.

SIZE the program, then RUN it and SIZE again. XB does not reset everything until you have made an editing change, as you know from debugging efforts after BREAKing (fctn-4) program execution. At this stage you get more information from an expanded system, which will show 8 bytes of memory used and 9 bytes of stack. Now repeat the process with a longer variable name. The length is reflected both in the original program length and in the stack used. The stack usage is 2 bytes plus the variable's name length more than the minimum we figured out before. Most likely the 2 bytes are for a linked list structure to help table searching, and there is a symbol table entry of the variable name. Now turn off your expansion system and be like everybody else with console only, and repeat the above. Now you will find the increase over the program length is always the 14 bytes we figured earlier no matter how long the variable name is. Now try

```
100 A2345678901234S,A2345678
901234S=0
```

Still 14 bytes RUNtime overhead ! Change the second A to a B to make a distinct variable name, 15 bytes long. Only another 14 bytes overhead ! So how come ? Maybe it's doing without the full word list link and squeezing things up a bit, but what about the symbol table ? The only consistent conclusion is that it doesn't have one as such, but points to the first location of the variable name in the program as located by the prescan. Read the Tutorial on prescans again.

If you wanted to make a faster interpreted Basic, you would, in the prescan, replace all variable names by some token plus a storage pointer to eliminate table searches. Which is just what TI claim in their Software Development Handbook to have done with the Basic for their 990 minicomputers. Unfortunately they failed to make an honest machine of the 99/4.

That should be plenty to chew on for this inaugural issue of the HV99 News. The next Tutorial will continue with the principles of program scrunching, getting more into the program writing end of things.

Funnelweb Farm
215 Grinsell St.
Kotara, NSW 2288
Tel (049)52-3162

END

****
* JUNIOR LIBRARY SOFTWARE REVIEW *
****

This month's review by D.McClure.

A Review of SHARSHOOTER.

This Game is far from original but has been well written. The programme is in three parts.

(1) The takeoff.
(2) The dogfight.
(3) The landing.

The Takeoff.

This section has been quite well done with reasonably good graphics and quite a good level of skill is required to accomplish a takeoff.

The Dogfight.

This is not as good as the takeoff or landing but is reasonable.It's main fault is that if the pursuit craft flies off screen you are dead. It often starts off screen so the game is over before it starts. The Tie fighter is quite hard to out manouver which makes the game into quite a challenge and a compelling pastime as you build up you skill to beat D'vader in his fighter.

The Landing.

dramatised but still holds it's own over the other two parts of the game and is not dissimilar to the take off.

In all the game has potential a could be a prized addition to your computer software collection. The progammer has excelled himself with his first real attempt at releasing a programme. Sharpshooter received a small honour when it was judged third in a Sydney Club software competition scoring 30 from a possible 50 points.

Sharpshooter is available from t H.V.99'ers. on cassette.

To run Sharpshooter you need Extend Basic and joystick.

```
**********************************
*          AUSSIE GUESS          *
**********************************
```

AUSSIE GUESS is a simple question and answer game for young people. The topic of course is Australia. The programme has been designed to allow the instructor (Parent) to develop question for the game which suit the child's needs. They can then be used immediately or saved to tape or disk for future use.

The questions are in three parts.

1) The question section.
2) The written (typed) answer which requires a keyboard input from the player.
3) The physical location of the answer for (2) is located by using the joystick and fire button.

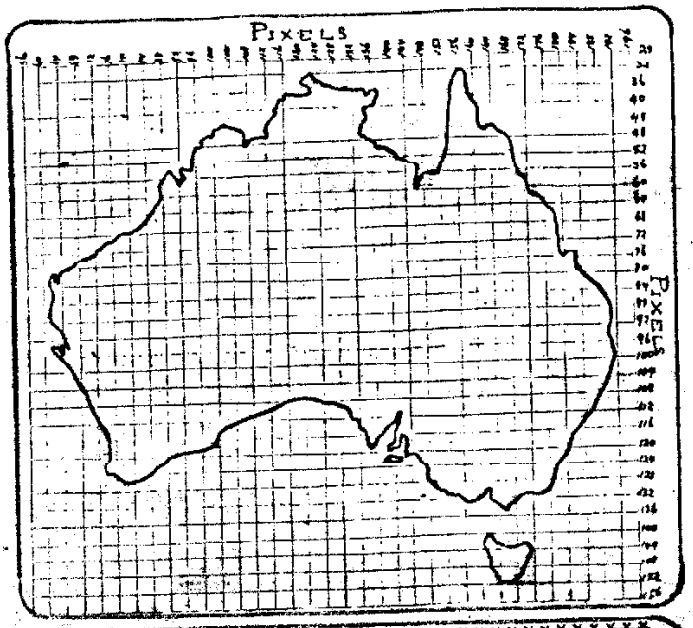
Prior to the map of Australia appearing on the screen the player is asked to input his or her name.

```
100 '********************
110 '*   AUSSIE GUESS    *
120 '*         By A.WRIGHT*
130 '*          H.V.99'ers*
140 '********************
150 OPTION BASE 1
160 DIM I$(10,3),M$(10)
170 CALL SCREEN(16):: CALL C
HARSET :: CALL SCREEN(10)
180 CALL SCREEN(10):: CALL C
OLOR(1,2,10,5,2,10,6,2,10,7,
2,10,8,2,10,2,2,10,3,2,10,4,
2,10)
190 DISPLAY AT(1,7)ERASE ALL
:"  AUSSIE GUESS  ": : :"
THIS IS A USER INTER-ACTIVE"
: :"EDUCATIONAL GAME FOR SCH
OOL": :"     GRADES 3 TO 6"
200 DISPLAY AT(11,2):"YOU HA
VE FOUR OPTIONS": :" ENTER 1
- TO LOAD QUESTIONS": :"
   2- TO DEVELOP NEW":"
    QUESTIONS"
210 DISPLAY AT(18,8):"3- PLA
Y/VIEW GAME": :"      4- EX
IT": : :"FOLLOW DIRECTIONS C
AREFULLY"
220 ACCEPT AT(20,20)VALIDATE
(DIGIT)BEEP SIZE(1):E :: CAL
L CLEAR :: CALL SCREEN(10)::
 ON E GOTO 380,230,450,1580
230 CALL SCREEN(10):: DISPLA
Y AT(1,2):"NEW QUESTION DEVE
LOPMENT": : :"  YOU WILL HAV
E AVAILABLE": :"SPACE FOR 10
 QUESTIONS AND"
240 DISPLAY AT(8,1):"TWO ANS
WERS PER QUESTION": :" REFER
 TO THE ACCOMPANYING": :" NO
TES FOR FORMAT DETAILS"
250 GOSUB 1460 :: GOSUB 990
:: GOSUB 1010
260 DISPLAY AT(3,4)ERASE ALL
:"ALL QUESTIONS AND ANSWERS"
: :"ARE NOW ENTERED": :"DO Y
OU WANT TO CHECK THEM": :"OR
 USE THEM ON THE GAME?"
270 DISPLAY AT(13,5):"ENTER
1- TO CHECK": :"          2-
 TO PLAY GAME": :"
3- EXIT TO STORAGE": :"
    4- RETURN TO MENU"
280 ACCEPT AT(22,12)VALIDATE
(DIGIT)BEEP SIZE(1):C :: ON
C GOTO 290,450,300,180 :: GO
TO 180
290 GOSUB 990 :: GOTO 1040
300 GOSUB 1570
310 OPEN #1:DEV$,SEQUENTIAL,
INTERNAL,OUTPUT,FIXED 64
320 FOR A=1 TO 10
330 IF (LEN(I$(A,1))=0)AND(A
<>10)THEN 350
340 PRINT #1:I$(A,1),I$(A,2)
,I$(A,3),A
350 NEXT A
360 PRINT #1:0$
370 CLOSE #1 :: DEV$=" " ::
GOTO 260
380 GOSUB 1570
390 OPEN #1:DEV$,SEQUENTIAL,
INTERNAL,INPUT ,FIXED 64
400 FOR A=1 TO 10
410 INPUT #1:I$(A,1),I$(A,2)
,I$(A,3),A
420 NEXT A
430 INPUT #1:0$
440 CLOSE #1 :: DEV$=" " ::
GOTO 260
450 DISPLAY AT(6,3)ERASE ALL
:"PLEASE ENTER YOUR FIRST NA
ME" :: ACCEPT AT(9,3):NAME$
460 DISPLAY AT(12,5):"QUESTI
ON CHOICE": :"  ENTER YES OR
 NO FOR EACH "
470 DISPLAY AT(20,3):"QUESTI
ON PART 1)": :"  QUESTION PA
RT 2)" :: ACCEPT AT(20,19)VA
LIDATE(UALPHA)SIZE(3):0$
480 ACCEPT AT(22,19)VALIDATE
(UALPHA)SIZE(3):P$
490 CALL CLEAR :: CALL SCREE
N(2)
500 CALL COLOR(8,16,2,1,10,2
,2,7,2,4,7,10,9,10,2,10,9,2,
11,9,2,12,7,2,13,7,2,14,7,2)
510 DISPLAY AT(24,1):"  PLEA
SE WAIT WHILE I DRAW"
520 CALL COLOR(3,16,2,5,16,2
,6,16,2,7,16,2):: RESTORE 32
767 :: FOR A=58 TO 61 :: REA
D C$ :: CALL CHAR(A,C$):: NE
XT A
530 RESTORE 970 :: FOR A=40
TO 43 :: READ C$ :: CALL CHA
R(A,C$):: NEXT A
540 RESTORE 980 :: FOR A=58
TO 60 :: READ R,C :: CALL HC
HAR(R,C,A):: NEXT A
550 FOR A=40 TO 43 :: IF A>3
5 AND A<40 THEN 560 :: READ
R,C :: CALL HCHAR(R,C,A)
560 NEXT A
570 RESTORE 850 :: FOR CR=33
 TO 34 :: READ PAT$ :: CALL
CHAR(CR,PAT$):: NEXT CR
580 FOR CR=96 TO 143 :: READ
 PAT$ :: CALL CHAR(CR,PAT$):
: NEXT CR
590 RESTORE 680 :: FOR CR=33
 TO 34 :: READ R,C :: CALL H
CHAR(R,C,CR):: NEXT CR
600 FOR CR=96 TO 143 :: READ
 R,C :: IF CR=111 THEN 610 :
: CALL HCHAR(R,C,CR)
610 NEXT CR
620 RESTORE 770 :: FOR A=1 T
O 12 :: READ R,C,RPT :: CALL
 VCHAR(R,C,61,RPT):: NEXT A
630 RESTORE 950 :: READ C$ :
: CALL CHAR(37,C$)
640 RESTORE 800 :: FOR A=1 T
O 8 :: READ R,C,RPT :: CALL
VCHAR(R,C,37,RPT):: NEXT A
650 RESTORE 840 :: FOR A=1 T
O 5 :: READ R,C,RPT :: CALL
VCHAR(R,C,111,RPT):: NEXT A
660 GOTO 1120
670 REM N.S.W./N.T.CHAR 94-1
03
680 DATA 13,24,14,24,15,23,1
6,23,5,15,5,16,5,17,5,18,6,1
7,7,18
690 REM S.A.CHAR 104-110
700 DATA 17,19,16,19,15,18,1
5,17,14,17,14,16,14,15,1,1
710 REM QLD./VICCHAR 112-125
720 DATA 7,19,7,20,6,20,5,20
,6,21,7,22,8,22,9,23,10,23,1
1,24,12,24,17,22,17,21,17,20
730 REM WA.CHAR 126-143
740 DATA 14,14,15,13,15,12,1
6,11,16,10,15,10,14,9,13,9,1
2,8,11,8,10,8,10,9,9,10,9,11
,8,11
750 DATA 7,12,6,13,6,14
760 REM N.A.
770 DATA 7,14,7,7,13,8,8,12,
7,10,11,6,10,10,5,11,9,2,8,1
9,4,8,20,5,7,21,6,9,22,4,11,
23,2
780 DATA 16,20,1
790 REM NSW
800 DATA 13,20,2,13,21,3,13,
22,3,13,23,2
810 REM NT
820 DATA 6,15,6,6,16,6,7,17,
6,8,18,4
830 REM SA
840 DATA 12,15,2,12,16,2,12,
17,2,12,18,3,12,19,4
850 DATA "FEFEFFFEFEFCFCF8",
"F8F8F0F0E0C0B0B0","FFFEFCFC
F8F8F0F0","F0E0E0C0C0C0B0B0"
,"00000000F1FFF7F"
860 DATA "40F07C3EFFFFFFFF",
"00000000FFFFFFFF","00000000
C0C0B0B0","FCFEFEFCFCF8FCFE"
,"00B0E0F8FCFFFFFF"
870 DATA "0F07010000000000",
"FF7F7F3F1F1F1F0F","E7EFCF8F
0F0B1B03","1F1F0F0706020000"
,"FFFFFFFFFFFF3F3F"
880 DATA "FFFFFFF0F0300000",
"FFFFFFE080000000","FFFFFFFF
FFFFFFFF","0000000000B0E1EF"
,"3F3F3F3F7FFFFFFF"
890 DATA "3F3F7F7F7F7F7F7F",
"183C3C3C3E3E1E3F","B0B0B0B0
98BCFEFF","B0B0B0B0C0C0C0C0"
,"C0C0E0F0F0F8FCFE"
900 DATA "00B0C0C0C0E0E0E0",
"E0E0E0EAFEFEFEFF","B0B0C0E0
F0F8FCFC","FCFCFBF8F8FCFCFC"
,"FFFEC080000000"
910 DATA "FFFFFFBF3F180C00",
"FFFFFF3F1E000000","FFFFFFFF
FFFCF0C0","FEFEFCF0E0E00000"
,"FFFFFFFFFFFFFC00"
920 DATA "FEFEFCF8E0C00000",
"3F7F7F3F1F070000","FF7F3F3F
3F3F3F3F","0F07030301010101"
,"7F7F3F3F1F1F0F0F"
930 DATA "0303010080B090F03",
"0703030303070703","00000000
01010505","030F1F3FFFFFFFFF"
,"000000000707F7F7"
```

This is followed by two questions asking if the player wants to respond to both or one of the two sections. This feature will allow the game to be played by persons who are not proficient at typing.
When questions are being developed simple wording should be used.
The co-ordinates are entered as 6 digits. The first 3 are the row in pixels and the second 3 are the column address in pixels, example 154098 represents row 154 column 98. These co-orinates are obtained from the accompanying map, the test for position of the answer has a tolerance of + or - 10 pixels and is tested against the top left corner of the sprite which is positioned by the joystick.
The typed answer to part 1 of the question when being interrogated will permitt minor spelling error to pass as correct answers.

```
940 DATA "0F1F3F7FFFFFFFFF",
"0103010101030307","00010101
010D47E7F","0000000000E0F3F7F"
,"0000000101B1E7EF"
950 DATA "FFFFFFFFFFFFFFFF"
960 DATA "0000000000F0FDFF",
"00B0F0FEFFFFFFFF","00000000
E0FCFCFE","FFFFFFFFFFFFFFFF"
970 DATA "000000000040E0E0",
"E0C0C0C0C0C0B000","FFFF7F7F
7F3F3E14","000000040C0F8FF"
980 DATA 15,20,16,21,16,22,1
8,22,19,22,19,21,18,21
990 CALL CLEAR :: PRINT "QUE
STION DEVELOPMENT"
1000 FOR A=1 TO 10 :: PRINT
A :: PRINT :: NEXT A :: PRIN
T :: PRINT :: RETURN
1010 FOR A=1 TO 10 :: D=A*2
:: ACCEPT AT(D,4)BEEP SIZE(2
6):I$(A,1)
1020 ACCEPT AT((D+1),4)BEEP
SIZE(10):I$(A,2):: ACCEPT AT
((D+1),15)VALIDATE(DIGIT)BEE
P SIZE(6):I$(A,3):: NEXT A
1030 GOSUB 1460 :: RETURN
1040 FOR A=1 TO 10 :: D=A*2
:: DISPLAY AT(D,4):I$(A,1)::
 DISPLAY AT(D+1,4):I$(A,2)::
 DISPLAY AT(D+1,15):I$(A,3)
1050 IF A<>10 THEN 1110
1060 DISPLAY AT(22,3):"ENTER
 NUMBER TO BE AMMENDED":"OR
PRESS ZERO TO CONTINUE" :: A
CCEPT AT(24,27)VALIDATE(DIGI
T)BEEP SIZE(2):B
1070 IF B=0 THEN 1110 :: GOS
UB 1450 :: D=B*2
1080 ACCEPT AT(D,4)SIZE(-26)
:I$(B,1):: ACCEPT AT(D+1,4)S
IZE(-10):I$(B,2):: ACCEPT AT
(D+1,15)SIZE(-6):I$(B,3)
1090 DISPLAY AT(22,3):"INFOR
MATION ENTERED":"PRESS ZERO
TO CONTINUE OR NUMBER" :: AC
CEPT AT(24,27)VALIDATE(DIGIT
)BEEP SIZE(2):B
1100 GOSUB 1450 :: IF B<>0 T
HEN 1070
1110 NEXT A :: GOTO 260
1120 GOSUB 1380 :: CALL CHAR
(64,"FFF1F1F0B0B0B0E0"):: CA
LL SPRITE(#1,64,16,50,20)::
 GO SUB 1440 :: T=0
1130 FOR A=1 TO 10 :: IF LEN
(I$(A,1))=0 THEN 1200
1140 CALL QNUM(A,NN$):: B2$=
" " :: GOSUB 1450 :: DISPLAY
 AT(1,1):"QUESTION NO.":NN$
:: GO SUB 1530 :: IF B$="NO"
 THEN 1200
1150 DISPLAY AT(2,1):I$(A,1)
 :: DISPLAY AT(3,4)SIZE(11):"
ANSWER HERE" :: ACCEPT AT(3,
16)VALIDATE(UALPHA)BEEP SIZE
(10):B$
1160 IF B$=I$(A,2)THEN GO SU
B 1300 ELSE 1170 :: GOTO 120
0
1170 N=0 :: FOR B=1 TO LEN(I
$(A,2)):: IF SEG$(B$,B,1)=SE
G$((I$(A,2)),B,1)THEN N=N+1
1180 NEXT B
1190 IF N=LEN(I$(A,2))-1 THE
N GO SUB 1300 ELSE GO SUB 12
90 :: IF B$="Y" THEN 1140
1200 IF LEN(I$(A,3))<>6 OR P
$="NO" THEN 1250
1210 GO SUB 1440 :: DISPLAY
AT(1,1):"USE JOY STICK TO PO
SITION":"SHUTTLE NEAR WHERE
YOU THINK"::I$(A,2);"  IS"
1220 GO SUB 1450 :: DISPLAY
AT(22,1):"PUSH FIRE BUTTON W
HEN IN":"POSITION" :: B$=" "
1230 CALL JOYST(1,X,Y):: CAL
L MOTION(#1,(Y*-3),(X*3))::
CALL KEY(1,KEY,STAT)
1240 IF KEY<>18 THEN 1230 ::
 GO SUB 1340 :: IF B$="Y" TH
EN 1210
1250 GOSUB 1440 :: GOSUB 145
0 :: IF A<>10 THEN GOSUB 148
0
1260 NEXT A
1270 GOSUB 1310 :: DISPLAY A
T(22,1):"THAT COMPLETES THIS
 SERIES":"OF QUESTIONS":"PRE
SS ENTER TO RETURN MENU"
1280 ACCEPT AT(23,20):A$ ::
CALL CLEAR :: CALL CHARSET :
: CALL DELSPRITE(ALL):: GOTO
 180
1290 GO SUB 1440 :: GOSUB 14
50 :: B=INT(RND*5)+6 :: GO S
UB 1470 :: GOSUB 1560 :: RET
URN
1300 GO SUB 1440 :: GO SUB 1
450 :: B=INT(RND*5)+1 :: GO
SUB 1470 :: T=T+1 :: RETURN
1310 GO SUB 1440 :: DISPLAY
AT(1,1):NAME$:"YOUR SCORE IS
 ":T:" FROM ":A:" QUESTIONS
" :: RETURN
1330 ! SUB POSITION CHECK
1340 CALL POSITION(#1,Y,X)::
 :: ROW=VAL(SEG$(I$(A,3),1,3
)):: COL=VAL(SEG$(I$(A,3),4,
3))
1350 IF X<(COL+7)AND X>(COL-
7)AND Y<(ROW+7)AND Y>(ROW-7)
THEN GOSUB 1300 ELSE GOSUB 1
290 :: RETURN
1370 ! SUB SET RESPONSE
1380 RESTORE 1390 :: FOR D=1
 TO 10 :: READ C$ :: M$(D)=C
$ :: NEXT D :: RETURN
1390 DATA "YES VERY GOOD","T
HAT IS CORRECT","CORRECT WEL
L DONE","RIGHT GOOD GOING","
THAT IS THE CORRECT ANSWER"
1400 DATA "NO THAT IS NOT CO
RRECT","THAT IS NOT RIGHT","
YOUR ANSWER IS NOT CORRECT"
1410 DATA "YOU ANSWERED INCO
RRECTLY","NOT RIGHT BUT KEEP
 GOING"
1430 ! SUB BLANK LINES
1440 CALL HCHAR(1,1,32,96)::
 RETURN
1450 CALL HCHAR(22,1,32,96):
: RETURN
1460 DISPLAY AT(24,1):"PRESS
 ENTER TO CONTINUE" :: ACCEP
T AT(24,27):A$ :: CALL HCHAR
(22,1,32,96):: RETURN
1470 FOR FL=1 TO 6 :: CALL S
OUND(-99,554,FL,1527,FL):: D
ISPLAY AT(22,1):M$(B):: GO S
UB 1530 :: GO SUB 1450 :: NE
XT FL :: RETURN
1480 DISPLAY AT(22,1):"PRESS
 T TO RETURN TO MENU"
1490 FOR AA=1 TO 80 :: CALL
KEY(0,K,S):: IF K=84 THEN 18
0
1500 NEXT AA :: RETURN
1520 ! SUB DELAY
1530 FOR DEL=1 TO 100 :: NEX
T DEL :: RETURN
1550 ! SUB TRY AGAIN?
1560 GOSUB 1450 :: DISPLAY A
T(22,1):"DO YOU WANT TO TRY
THAT":"QUESTION AGAIN Y OR N
" :: ACCEPT AT(24,20):B$ ::
RETURN
1570 DISPLAY AT(10,1)ERASE A
LL:"Enter name of data stora
ge": :"          device  (Ca
pitals)" :: ACCEPT AT(14,9)B
EEP:DEV$ :: RETURN
1580 DISPLAY AT(10,1)ERASE A
LL:"  ALL DATA IN MEMORY WIL
L BE": :"  LOST WHEN PROGRAM
ME HALTS"
1590 DISPLAY AT(16,1):"   PR
ESS E TO END PROGRAMME": :"
          K TO RETURN TO MENU"
1600 CALL KEY(0,K,S):: IF S<
1 THEN 1600 ELSE IF K=69 THE
N 1610 ELSE 180
1610 END
1620 !SUB QUESTION NUMBER
1630 SUB QNUM(A,NN$)
1640 ON A GOTO 1650,1660,167
0,1680,1690,1700,1710,1720,1
730,1740
1650 NN$="ONE" :: GOTO 1750
1660 NN$="TWO" :: GOTO 1750
1670 NN$="THREE" :: GOTO 175
0
1680 NN$="FOUR" :: GOTO 1750
1690 NN$="FIVE" :: GOTO 1750
1700 NN$="SIX" :: GOTO 1750
1710 NN$="SEVEN" :: GOTO 175
0
1720 NN$="EIGHT" :: GOTO 175
0
1730 NN$="NINE" :: GOTO 1750
1740 NN$="TEN" :: GOTO 1750
1750 SUBEND
```

CONT NEXT PAGE



****
* T.I. TECHNOLOGY *
****

Hello from TI. TECHNOLOGY. In this column I intend to present information on hardware aspects of the TI99/4A computer. This will include such items as: care, maintenance and enhancements to your existing console and peripherals.

There are many exciting things happening in the hardware development field for the TI99/4AA. (For a computer which is supposed to be dead it will not lie down!!) The developments range from modulators which can operate on UHF or VHF TV sets with only a minor modification, to high speed RAM disks for speeding up file manipulation.

The item which at present most intrigues me is the development made by the WEST AUSTRALIAN GROUP T.I.U.P. of putting 32K of RAM inside the console. Yes I know the comment and I've made it myself "but there is no more room inside the console". However above the main computer board, outside the metal shield is almost enough room for another computer and this is where those clever SAND GROPERS put their 4X64K RAM chips. All the address and data lines are available on the back of the module socket and the memory chip decoding is already done on the main computer board.

How much does it cost? Well about $105 in RAM chips and very little else which is not bad when you consider the TI. unit now costs around $250.

The big advantage however is that you can carry your console around with you and run all the software that reqs 32K without having to use the station wagon to carry the PE

I am currently building the WA 32K expansion and it took me about 1 1/2 hours to bring all the necessary wires from inside the console to a 50 pin plug which will connect to the RAM board. I will tell you the outcome of this modification in the next issue. If it works i'll assist people to build their own 32K system.

Also next month I hope to describe a way to

(a) Operate a monochrome monitor from the TI. computer.

(b) Get rid of that ugly interface unit which plugs into the side of the console and goes to the PE BOX.

(c) Reducing the noise from the PE BOX fan. N.B. DO NOT use a series dropping resister external to the box as has been suggested in other magazines as it will almost certainly be unsafe unless done by a qualified person.

ALAN BYRNE.

****
* STOP PRESS: FORTH INFORMATION *
****

THE MILWAUKEE AREA USERS GROUP is serving as co-ordinator for TI-FORTH users. For more information write to FORTH NATIONAL INFORMATION CENTER, 1007 n. 71 St, Wauwatosa, WI 53213. U.S.A.

The group supposedly has information about modifying TI-FORTH for use with double sided disks, as well as other information of interest to FORTH users.

(MICROPENDIUM NEWS)

****
* EDITORS DESK *
****

Anyone wishing to contribute information or articles to HV99'ers NEWS can contact me on 049-487076 before 9PM. Or by writing to 15 Gayton Close. Warners Bay 2282 N.S.W.

As this is the first issue of HV99'ers NEWS please bear with any spelling or formatting mistakes. I would like to thank all those who contributed in any way to the production of the first issue and in particular to Joe Wright whose assistance was invaluable.

If you would like to know more about our club or magazine please dont hesitate to contact our secretary PETER COXON at 25 Reserve Rd. Wangi N.S.W.

STEVE TAYLOR

CASSETTE LOADING TIP

This is an old tip which all the old timers like myself know, but probably not to many of our newer members have heard it. If you have ever waited for an ordinary basic program to load only to have the dreaded "NO DATA FOUND" to appear, you will find this a useful tip.

Before loading a program from cassette complete the following procedure:

```
NEW
100 FOR A=96 TO 159
110 PRINT CHR$(A)
120 NEXT A
RUN
OLD CS1
```

If you follow this procedure, when the tape loads, the redifined characters march across the screen as the tape buffer is filled. If they don't march then you have time to rewind the tape, alter the volume and have another go!
COURTESY OUR NEW FOUND FRIENDS TIUP.

NEW 9900 BASED COMPUTER

The new computer is fitted with a TI.9995 chip and is supposedly everything the 99/4A and 99/8 should have been with the added plus of being compatable with original 99/4A hardware and software.
Prototype design work is completed however the technical specs have not been released for security reasons. The formal release date is set fot the June CES. Just to wet your appetite expected features are.

* Standard memory configuration is 128K. expandable internally to 512K. with option to address up to 1MB. of mem. directly
* The 16K. VDP Ram is expandable to 64K. with a new VDP chip from TI.
* Full size "Selectric" keyboard.
* The existing TI. PE Box will be used as the CARD cage.
* Resolution twice that of 99/4A in both directions.
* Full 80 Column display.
* Runs at 10 MHZ.!!!!!!!!!

For further details and subscription to a newsletter devoted to the new computer write to RYTE DATA Box 210 MOUNTAIN ST. KLIBURTON. ONTARDIO KOM1B0. CANADA

Further Ramblings with ACCEPT.

As you have probably read, using ACCEPT AT will only allow you to input data up to the 28th column. You cannot wrap around to the next row. And ACCEPT without AT will allow this, in fact ACCEPT will allow you to input data up to 9 screen lines and 3 places on the 10th line, a full 255 bytes.

ACCEPT also accepts QUOTES, COMMA'S, PERIODS etc. It does not strip off leading blanks, but trailing blanks are more complicated. If the number of trailing blanks has wrapped into the next row, then the trailing blanks up to column 28 are not stripped off, only those that wrapped around to the next row, ie. if you input half a line of data, then the rest of that row, all the next row and half the 3rd. row as trailing blanks. Only the last half row of trailing blanks will be striped off. I have not found a use for this feature yet, still some body might.

If you DISPLAY AT row 24, the ACCEPT without a SIZE then the input field is not cleared, so what ever you DISPLAY'ed AT row 24 stays there. But to get the ACCEPT to accept what is on that screen line, you must use FCTN D to move the cursor over the part of the line that you want accepted. The cursor must pass over the last point (number, letter etc) that you want, not just as far as it. FCTN ERASE works, but FCTN INS and FCTN DELETE only works if you have FCTN D over the line to read it first.

If you use ACCEPT with a SIZE it works basicly the same as ACCEPT AT. Except that, if it is followed by an ACCEPT without a SIZE, the cursor is left at the column SIZE+1.

```
i.e.  600 ACCEPT  SIZE(-8):A$
ACCEPT B$
```

Will after inputing A$, leave the cursor at column 9 for the input of B$.

If any body gets any results different to this or has any more information on this please let me know.

Brian R.

HV'99ERS LIBRARY NEWS

Hi 99'ers

At present your library has over 400 Programs available on tape or disk for access to by members of the HUNTER VALLEY 99'ers USER GROUP. These are made up and sold for your enjoyment and education at the monthly meetings on tape for $3.00. It is also available on request on disk for $5.00. for those who have an expanded system.

Due to the problems with the various cassette recorder compatability, the tapes can be exchanged but if the second tape has problems then it can be arranged for the tape to be recorded with your own cassette at a time suitable to both parties.

While our group is small and friendly we can by arrangement make up tapes or disks with the programs of your personal choice either on club media or your own. Cost is 30 cents/program plus cost of media, or if you supply your own tape or disk 30 cents/program.

Back copies of previously issued tapes are also available by placing your order on club nights. We will have details of programs in the library tapes issued in the past. We are in the small "back room" off the main hall at monthly "meet's" and at most weekly classes sometime during the evening.

At the monthly "meet's" we also have lots of books and information for you to borrow. Just see us and sign your name address in the borrowers record book. We are NOT a SHOP but if we can help you locate anything we will do our best to please and foster the 99'er Family atmosphere.

Dont forget as support for software for the TI 99/4A dissapears from the commercial retailers (you may have noticed this already) We your "LOCAL" friendly users group will continue to SUPPORT you with all the enthusiasm the new committee can give as long as we have your IDEAS AND SUPPORT as well.

Remember the club is you and we need you. The Library is yours use it. We have games, utilities, educational, strategy, adventure and lots more besides, if you have programs to exchange we will gladly do so. The library is EXPANDING and we have interchange of information with other groups around the world. Here is a preview of some goodies to look forward to.

EXTENDED BASIC

Air Traffic Control.	Land Planes
Cavemaze.	Text Advent
Corner Wars	Space War
Dogfight.	Biplanes
House Finance.	Watch The $
Inter Planet Rescue.	Space Game
Mini Golf	NO Walking

BASIC

Mini Organ	Save $$$$$
3D Car Race	Hang On !!!
Hamurabi	Good Luck !
Lunar Lander	Space Game
Pink Panther	Graphics
Raindrops	Musical
Candyman	Musical

UTILITIES

Keyword	Locater
Medical Records	Tax Help
Personal Banker	Overdraft

thats it for now
Happy Programming
Al Lawrence.

DISCLAIMER