

# NEWSLETTER

of

# TIBUG

TI - 99/4A - BRISBANE USER GROUP INC  
P.O. BOX 3051  
CLONTARF MDC, QLD AUST 4019

MAY 1992













# LETTER TO THE EDITOR

PO BOX 1038  
MILTON QLD 4064

30.4.92

Gary Christensen  
TIBUG  
18 Zammit Street  
DECEPTION BAY QLD 4508

## LETTER TO THE EDITOR

My name is Kerrie Wilson and I have been a member of TIBUG for a couple of years.

After reading your Editorial in the April Newsletter I decided to drop you a line.

I purchased my TI 99/4A in 1983 in Sydney after a friend of mine had bought one and got me hooked on Tunnels of Doom. I used it constantly, trying to buy computer magazines with TI 99/4A programs in them, typing them in and then changing some lines to see how the programming worked. It was easier for me to find errors (my typing) and correcting them than trying to learn how to program from the manuals.

When I purchased Extended Basic I thought I had advanced to higher levels! (although I still used cassette tapes).


I didn't know there were user groups to help people like me until I moved to Brisbane. I had a problem with my 99/4A and was referred to Gary & Col who fixed my problem with ease. Through them I purchased some more modules and joined the User Group. Then a new career started for me and my 99/4A was put in mothballs. I renewed my membership every year just to receive my newsletter which I really enjoy reading.

One day my son asked me what was in the box at the top of the wardrobe and I took down my 99/4A and between the two of us we assembled it in his room and I showed him the games I had. Now I have trouble getting him off it. I am really glad I have kept all of those newsletters because even though he is only 6 years old he will eventually want more out of the machine as he grows, and the programs and tips in those newsletters will help him understand the 99/4A better.

I still only have modules and cassette tapes but maybe one day we will advance to a disk drive!

Thanks for your informative newsletters and keep up the good work.

Regards,

  
KERRIE WILSON

P.S. I enjoy the Pirate Adventure Games but Tunnels of Doom is still my favourite!



# YOU DON'T HAVE TO HAVE IT ALL

by Jim Peterson

Do the conversations at your user group meeting sound like a coffee break in Silicon Valley? Are you confused by talk of GROMs and GRAMs, puzzled by references to HFDCs, intimidated by discussions of megabytes and frightened by talk of burning EPROMs? Well, join the crowd, buddy - so am I!

There are basically three types of people interested in computers. First, there are those who use a computer to run programs, to accomplish something useful or just to have fun. I believe that those people are still in the great majority, although we don't hear much from them.

Then, there are those who get their kicks out of writing programs, of creating software for others to use. There aren't too many of those left in the TI world.

And finally, there are those who like to tinker with the computer, soup it up, plug in doohinkies and thingamajigs, and talk in that strange language I mentioned above. I don't know how many of those folks there are, but they are certainly the most knowledgeable, active, and interested, and they tend to dominate the conversations and the printed material in the TI world nowadays.

I presume that those fellows also do actually run programs on their souped up systems. And, some of them must be skilled programmers, because many of their hybrid hardware creations would be useless without specialized software.

I'm very glad that those people are around. Once in a while they invent something that I actually find useful, and they are a lifesaver when my equipment breaks down.

But, don't be intimidated by all that high-tech talk, and don't think that the computer world is passing you by. There are so many things to do with a computer that no one could possibly find time to do them all. Do your own thing and don't worry about the rest.

I have operated a TI software company for seven years, and I also spend a lot of time writing programs, using the computer as a word processor, etc. I probably spend more time on my TI than 90% of the users. So, what does my equipment consist of?

I have a console with the Extended Basic module plugged in, attached to a P-box which contains a TI disk controller, two double-sided drives, the 32k card, RS232 card, and a Horizon Ramdisk. Also plugged into the RS232 card is an old Gemini 10X printer and an Avatex 1200 baud modem.

I also have a Speech Synthesizer, a pair of TI joysticks, a TEII module and an Editor Assembler module, all of which I plug in occasionally when I need them; also, a cassette recorder and cable which hasn't been used in a long time.

I use Triton's Super Extended Basic module because it has some editing features which are useful when programming. It also has some limited plotting capability which I have never used - and have never heard of anyone who has. If you don't program, it would hardly pay to switch from the old TI Extended Basic. I also have the Mechatronics module but never got around to trying it.

I had a Gram Kracker but soon sold it and bought a Ramdisk instead. The Gram Kracker has fantastic capabilities if you have the skill and knowledge to take advantage of them, but most users don't seem to have done much beyond personalizing the title screen.

I had a widget, and I guess it is still collecting dust around here some place. It was a nuisance, and since I use XBasic 99% of the time I didn't need it. There are now widgets or "module expanders" that allow you to access more than one module from within a program. That is, if you have the skill to write such a program. I don't know that anyone has released such programs to the public domain, and I can't think of any practical use except to access TEII speech from XBasic - but you can do that with the Text-To-Speech disk.

The ram disk is the one tool that I



would not be without. In order to assemble my TI-PD catalog, I screened over 4000 programs, debugged and modified, merged in help files, conversions to XBasic and loaders, and assembled over 400 disks of programs. It took me hundreds of hours of work - without a ram disk it would have taken thousands of hours and I would not even have attempted it.

The ram disk enables me to switch from one program to another almost instantly, and with John Johnson's Boot program I can just as quickly catalog a disk or view a file. Mine has 256k of memory. I could get one with much more memory but I see no reason to do so; I have every program on it that I am apt to use even once a month, and it is only half full. That leaves plenty of room for temporary storage and downloading.

However, if you only use your computer to play games, do a little word processing and a bit of record keeping, a ram disk would be an expensive convenience rather than a necessity.

Since my ram disk is only half full, I would consider a hard drive to be about as useful as the mammalian appendages on a swine of the masculine persuasion. If I was running a BBS, sure - or if I was doing a lot of work with those memory-gobbling graphics and needed everything quickly accessible.

My old Gemini printer has been a faithful workhorse, although the hood over one sprocket wheel has lost its spring and is being held down by a loop of elastic cord. I will have to give it up soon, because the Gemini printer codes are becoming obsolete and I need to be able to write and test Epson codes. But, I hate to give up these 79-cent typewriter ribbons and start getting ripped off on \$2.50 cartridges! As for a color ribbon, the temperature will have to go way down, down under, before I pay for one of those.

Once in a while, when someone sends me a double-density diskfull of stuff, I wish I had a CorComp disk controller. Otherwise, with diskettes selling for a quarter or less, it wouldn't pay to change.

If I ever get around to subscribing to GENie or Delphi, it will pay me to get a 2400 baud modem.

I can't think of anything else I need, and I don't want what I don't need. If I really wanted to play joystick games, I would certainly get something better than the TI joystick. And if that MIDI interface cable becomes a reality, I will be sorely tempted.

I can't see any advantage in putting the 32k under the hood, or anyplace other than where it is now. If I used speech a great deal, it would be nice to get rid of the synthesizer - but I know only one user who uses speech that much. I don't need a clock built in because I have a watch on my wrist. If I really did a lot of serious writing, an 80-column card would be wonderful. But then I would have to buy a monitor capable of displaying 80 columns. I certainly don't want to give up color, and high-resolution color monitors cost more. I would still want to use my old monitor for programming, because I like to write programs for folks who have basic equipment. I don't have room on my computer desk for two monitors, so I think I'll pass.

I'm a three-finger typist, so a RAVE keyboard wouldn't speed up my typing very much. If I really wanted an IBM keyboard and 80-column capability, I would throw in a few bucks more and get a Geneve.

So, what about the Geneve? If I had an irresistible urge to run the few great programs that have been written for it, or if I wanted to explore its great programming capabilities, I would get one. But, I like to write programs for other people to use. When so few are interested in programs that I write for a computer that sold in the millions, why would I write programs for a computer purchased by a couple of thousand people?

I am sure that many folks will disagree with what I have written. That's why I wrote it. I hope they will disagree so strongly that they will immediately boot up Funlweb and compose a blistering reply. But don't send it to me - send it to your newsletter editor. The newsletters are badly in need of more articles by more writers!





# WORD PROCESSING #2

by Col Christensen

In part 1 in the April issue I covered such topics as command and edit modes, setting tabs, screen windows, word wrap, screen colours, deleting and inserting, reformatting, the Oops function and the upper/lower case conversion keys. I hope you have practised all these and are fairly conversant with the procedures for each so that their use comes automatically as you encounter a need for them. Part 2 here will deal with some of the command mode functions of the WP.

## LINES

The L command will display the four lines options: Move, Copy, Delete and Show lines. To execute any one of them you don't have to go to Lines first, just go directly to it in the command mode by typing M, C, D or S then pressing <ENTER>.

## COPY and MOVE

This is the cut and paste function used by our WP. The main difference between the two is that the COPY function leaves the original text intact and merely makes a copy somewhere else while the MOVE deletes the original section. It is, unfortunately, restricted in its use to only whole lines of text. This is a slight drawback but the limitation can be overcome with a little extra effort.

To effect one of these two, you need to know the first line number and the last line number of the section to be copied or moved, and the line number after which the text is to be placed. Escape to the command mode and type C to copy or M to move and press <ENTER>. The prompt, "start line, stop line, after line" appears. Suppose you want to copy lines 19 to 24 inclusive and place them after line 16. You now type your line numbers in either of two ways, whichever you are more comfortable with. You can type 19,24,16 with commas to separate each or 19 24 16 using spaces instead, then press <ENTER>. After a brief delay depending on how much text has to be manipulated it will be done and your cursor will be ready waiting for you. Try out both the Copy and the Move and check the text to convince yourself that what you wanted to happen actually did.

Note that the WP won't allow you to copy or move text to a line number that is not currently existent. Suppose your text

so far ends at line number 41. You can move text after line 41 but not after line 42 or higher. If you did want to tuck some text after your last line number but separated from it by a few blank lines then firstly put some blank lines at the end of your text. Note also, and this applies to most places where line numbers are involved, that line number 0 is valid and indicates the first line and line number E (for End) is also valid to indicate the last line. So, in a MOVE, an entry such as 68,E,0 will move lines 68 to the end of your text and put them before line 1. Note thirdly that when you COPY some lines of text, you will end up with more total line of text than before, but if you move lines, the final line count will be the same as before.

Now suppose you want to move, say, a long sentence and place it in a different position. The sentence will be sure to start and end somewhere in the middle of a line. Murphy's Law makes sure of that. Without a block move function in our WP we have a few more steps to do. The idea is to use the insert keystroke to split both the start and ending lines so that the complete sentence and nothing else occupies a unique set of line numbers ready for moving. You also need to split the line into which the insertion must go.

I won't go into detail on the steps to follow to effect a MOVE but leave it for you to nut out. But just one little pointer though. When you use the insert keystroke to split lines of text and wish to move the cursor down the screen, do so with the arrow keys and not the <ENTER> which will leave a CR symbol that you most likely do not want. By all means use the <ENTER> key after the insert if you want that point eventually to be the end of a paragraph.

## MORE ON REFORMATTING

Now that you are making insertions and moves within a paragraph and leaving a mess in the text buffer and on the screen you'll probably like to tidy things up by reformatting. What I want to impress is that it is not necessary to go way up to the beginning of a paragraph to do this. Just move the cursor to a point anywhere before the mess, press insert (FCTN/2) and press reformat (CTRL/2).

Although this section relates to tab settings, it has a bearing on reformatting too, for, as I explained before, reformatting takes place between the left and right tab settings. Tony McGovern has incorporated dual TAB sets into the later



versions of our WP. You can now have one set of tabs for part of your document and different tab settings for another part. Both sets of tabs will be saved to disk with your document and retrieved again the next time you load it into memory again. To change the tabs, escape to the command mode and type ST (swap tabs). If the alternate set has not been set up, do so, and press <ENTER> to accept the new set. The screen format and reformatting will follow the new tab settings. To revert to the other tab setting escape to the command mode, type ST and press <ENTER> twice, one to accept the "ST" input and one to accept the tab settings. It is so simple to make the change over and is a very useful addition to the WP.

#### MARGIN RELEASE (CTRL/Y)

The cursor movement is limited by the tab margin settings so that it can only move within the left and right margins. If you find a need, however, to move the cursor outside these settings, it can be done on the next keypress after pressing CTRL/Y. In other words you need to move the cursor to the margin you wish to cross, press CTRL/Y and then the appropriate arrow key.

#### DELETE LINE and SHOW LINE

Normally you would delete a line or two of text by using FCTN/3 but there are times when a large number of lines have to be deleted. This is done in the command mode after typing D and pressing <ENTER>. The prompt, "start line and stop line" tells what to do. Separate the relevant line numbers with either a comma or a space and press <ENTER> when you are sure you have typed the numbers correctly. OOPS won't help you recover from an error here. Tony has greatly improved the speed of the delete function in later versions.

To find a particular known line number quickly the Show function has been included. In the command mode type S and press <ENTER>, then type the line number of interest. That line number will be displayed as the first on the screen.

In both of the above functions line number 0 and E for zero and End are valid.

#### SAVING, LOADING, PRINTING FILES

This will be just a brief description for the time being. Saving is done (after escaping to the command mode) with the command SF followed by <ENTER> and then typing the filename that you want it to

saved as, such as DSK1.RECIPE3. Loading a file from disk needs the command, LF instead. These procedures are the ones you use 99% of the time but there are powerful extensions to these commands that will be presented in Part 3. To print the text in the memory buffer, use the command PF. When asked for a devicename your response will, for most people, be PIO, otherwise RS232. The printout should match exactly the width that text appears on your screen. More details on this also next time.

#### SHOW

The S command allows you to control which line numbers will appear at the top of the screen. Suppose the assignment you are writing is nearing 500 lines in length and you want to refer back to the first paragraph. Show will speed up the process of displaying it for you. Escape to command mode, type S and press <ENTER> and type a suitable line number. That part of the text will appear the instant you press the <ENTER> key. The line number, E for End, is valid and quickly shows the very last line of your work. Roll down (FCTN/4) and Roll up (FCTN/6), remember, also move the text up or down 24 lines at a time.

That is all for now. In Part 3 the search function will be discussed and there will be more detail on file handling. Also we can make a start in using the text formatter to make our printouts look more professional.

---

## WHAT'S NEWS

I was in the back yard the other day when Tracey (my wife) came around with an enormous package in her arms. It was the software that we ordered from Tex-Comp. We now have 2 Microsoft Multiplan and 2 Logo II packages for sale. The price is \$20.00 each.

Multiplan is a spreadsheet and is very useful for maintaining records, particularly those that include finances. Logo is a learning tool for children aged 4 to 99. It uses simple line graphics to teach how the computer responds to commands and it teaches programming in a way that is easy to learn. It also builds good programming habits. Speak to me (Garry) if you want to know more about either of these pieces of software.

Tex-Comp also advise that TI-Writer is no longer available in module format but



they do have TI-Writer II on disk. It includes a manual but I don't know if it is the same as the original.

I phoned OPA the other day. The hold up in the supply of the TIMs seems to have been a supply of video chips. The gentleman that I was speaking to could not confirm that the chips had arrived but he indicated that Gary Bowser was going on holidays soon and he wanted to have our order finished before then. I hope that means it won't be long. Thank you for your patience.

OPA have also released an EPROM for the Geneve. It has TIMODE files on the EPROM instead of booting MDOS. The GPL files load almost immediately. The features are the ability to boot MDOS from any valid device - floppy/hard/RAM disk, improved keyboard driver, built-in mouse driver, GPL files are the equivalent of Son of a Board, additional 4K of GPL code to load cartridges, 4K of code for a micro-manager for floppy/hard/RAM disks, support for 9938/9958 devices, true lower case and 100% TI compatible. Cost is \$45 US.

Other products listed in the OPA catalogue are TASS 2001 (TIA slide show), Diskodex 2001 (disk cataloger), Recallit 2001 (database utilising RAMBO modification for RAMdisks), 9T9 Game Package, Horizon ROS (EPROM ROS for the horizon RAMdisk), RAMBO modification for Horizon RAMdisk, Morningstar RAMBO (for the 128K memory card), Ramblo Developers Package, Son of a Board (operating system upgrade for the TI), GPL Programming Package, TI Image Maker (80 column upgrade), and POP-Cart (puts several cartridges onto one board). OPA, 432 Jarvis St #501-502, Toronto Ontario, Canada M4Y-2H3.

Electronic Systems Design recently showed a prototype of their hard disk controller. It was not operational but the company gave a release date of April 15. Barry Boone is writing the DSR. Prices are expected to start at \$160 US.

Andy Frueh from LIMA Users Group has written a program called DV Manager. He says that it will view, copy, delete files and will search for text, count words, and print text files. Some editing features are also provided. Contact Andy on 638 Maplewood Dr, Lima OH 45805-3418, USA.

There are moves afoot to raise funds to buy the rights for MDOS from the authors, Paul Charlton in particular. Beery Miller

is collecting donations and if enough money is raised they will be able to purchase the copyright. Once they have the commented source code, they can get on with completing it and providing adequate hard disk support.

It seems that the greatest minds think alike. You may remember an editorial some months ago where I suggested a replacement motherboard for the TI. Don O'Neil is reported to be discussing the possibility of just that with Bud Mills. The new motherboard is expected to use a 9995 processor (like the 9640) and include up to 1 Meg of memory, a 9938 video chip with 192K Ram, Mouse and cassette port and use a standard keyboard. There will be provision for the PE Box to plug in and standard GROM ports. The TI GROMS will need to come out of your old motherboard. There may be provision for a second 'slot' but no decision on what will be plugged in there has yet been decided.

Harrison software have released a product that is claimed to combine advanced assembly routines with a skeleton X BASIC program. The user can call the routines to operate on arrays in their program. Features provided include a fast sort routine for data in your array and a user configurable menu system. The cost is \$6 and it comes from Harrison Software, 5705 40th Place, Hyattsville MD 20781, USA.

---

## BITS & PIECES

by Col Christensen

### CHEAP FLOPPY DISK DRIVES

Some hardware items to start. I had been told that there were inexpensive disk drives currently available. They are 360k (double sided double density) slimline types with the Mitsubishi brand name. They are made in Japan and are guaranteed by the retailer for 1 year. The price is \$50 each and you couldn't better the price even in a fire sale. I bought two and installed them and they are working perfectly although I did have some trouble to start until I found that the disk I was trying to read was incompatible with the TI disk controller in my PEB. Get in early if you want some as there were only about 700 left in stock three weeks ago. The company is Energy Control (otherwise known as Accord Computer Engineering Pty Ltd) and their



address is 26 Boron Street, Sumner Park,  
(Ph 376 2955).

#### SPEECH ADAPTOR CARD

A few speech cards are expected to be available by the time of the next meeting. I would like to reserve one to post to an interstate member in exchange for a synthesizer I got from him about 8 weeks ago. There are only 9 being made so if you want to move your speech into the PEB, contact Chas (Ph 273 6254) who is arranging their construction or you can also contact me (Ph 284 7783).

#### TI-WRITER QUIRK

Did you know that with TI-Writer you can have in memory thousands of lines of text 80 characters long without getting a text buffer full message. Try this. With word wrap off for convenience, fill the first line with 80 characters all the same, say all asterisks or all dollars. Go to the command mode, press C for copy and press <ENTER>. Type the sequence 1,1,1 to copy line one and put the copy after line 1 so that you now have 2 lines the same. Now continue copying and each time doubling the number of lines until you get a memory buffer full message. Do the copy with these sequences to double the number of lines: 1,1,1 (you already had done that) 1,2,2 1,4,4 1,8,8 1,16,16 1,32,32 etc. Then go to the last line showing on the screen. Over 4000? Do some maths with that data. A character is represented by a byte and 4000 lines multiplied by 80 characters on each makes 320,000 or 320kbytes all tucked up in part of the 32kbyte memory expansion. You don't believe me! O.K. Delete about half leaving only lines 1 to 2000. Now save this file to a blank formatted double sided disk. This disk format can store 180kbytes if single density and 360k bytes if double density. While you wait for the save to finish, it might be an idea to put the kettle on. When the save is complete, do a directory of the disk. There should be one file that used 702 sectors, almost the whole of a 180k disk. If you still don't believe, you can Purge the TIW buffer and reload the file from disk. Yes, it all loads back in.

I'm amazed at the foresight and planning adopted by the TI people involved in such software development way back in 1982. That's when TIW was released but I guess it could have taken at least a year to develop and beta test. Using the then current state of the art in computer

technology, they certainly got the utmost out of the 99/4A.

What TIW does is to do a form of compression of each line of text before storing it in memory. Only where characters are repeated is this compression used so you can see that the above example uses the ultimate of compression. The 80 characters are stored in a 3 byte memory space plus 2 bytes in the line number table to point to where the line is in memory. Tony McGovern uses the term scrunching for this compression. The three bytes that the line is scrunched into are:- >03 being the number of bytes in the scrunched line. The ASCII value of the repeated character to which >80 is added (a flag to indicate repetitions). An \* is >2A, so this plus >80 makes the byte >AA. The third byte is >50, decimal 80, telling how many repetitions. Simple really. All the interpreter has to do to convert a scrunched line for screen or printer output is, after the line length byte, to output each byte if less than >7F else subtract >80 from those bigger than >7F to get the byte and output that byte the number of times as given in the byte following it.

#### EXPANDED SYSTEMS FOR SALE

We have a few expanded systems on the market just now so if you know of possible buyers put them in touch with me.

#### THE MICRO EXPANSION SYSTEM

This is coming along now, almost up to the final circuit board and software development. A further extension I am planning but haven't tested on the final board is to double the capacity of the ROM so that it can also house a text formatter program. So not only will it be useful as a word processor with the capability of driving a printer, but the output can also be formatted. Another innovation is the provision of a second 32kbyte memory chip which is battery or "super capacitor" backed and acts as a storage device for text files. The upper three quarters of this memory will be able to store a full TI-Writer buffer of text and the lower part can separately be loaded with about one-third of that amount.

#### P.E. BOX AND RAMDISK

A club member recently gave me his PEB as it was locking up his system. Removing in turn the Ramdisk, the 32k Memory and the RS232 cards made no difference. The trouble



disappeared however when the Disk controller card was removed and placed in the adjacent slot. What was also intriguing was his ramdisk was also locking up the works. As the contents of the Ramdisk at the time of the problem were important, I tried to negate the auto power-up routine using Mini Memory as a first step to getting back control. But, alas, no matter which way I tried, as soon as the PEB was switched on, everything locked up. The remedy in such an extreme case is to remove one of the back-up batteries but doing that would lose all data stored in it. I have a gadget that has never failed before so I tried it. It is a small adaptor one end of which plugs into the I/O port of the console and a ramdisk connects into the other end. The +5v for the Ramdisk comes from a console connection on the adaptor via a flying lead that is clipped onto an appropriate point on the Ramdisk. Now it was a simple matter to plug the adaptor and the ramdisk in (less +5v supply), switch on, connect the Ramdisk's +5v and change the first byte of its DSR from >AA to some other value. With the Ramdisk back in the PEB, the ROS was reloaded and all the data was saved.

line and hit the Enter key, it looks to see if there is anything it can't understand - such as a misspelled command or an unmatched quotation mark. If so, it will tell you so, most likely by SYNTAX ERROR, and refuse to accept the line. Next, when you tell it to RUN the program, it first takes a quick look through the entire program, to find any combination of commands that it will not be able to perform. This is when it may crash with an error message telling you, for instance, that you have a NEXT without a matching FOR, or vice versa. And finally, while it is actually running and comes to something that it just can't do, it will crash and give you an error message - probably because a variable has been given a value that cannot be used, such as a CALL HCHAR(R,C,32) when R happens to equal 0.

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

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

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

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

## DEBUGGING

When you have finished writing a program, the next thing you should do is to run it. And, very probably, it will crash! Don't be discouraged. It happens to the very best of programmers, very often. So, the next thing to do is to debug it. And you are lucky that you are using a computer that helps you to debug better than some that cost ten times as much.

There are really three types of bugs. The first type will prevent the program from running at all - it will crash with an error message. The second type will allow the program to run, but will give the wrong results. And the third type, which is not really a bug but might be mistaken for one, results from trying to run a perfectly good program with the wrong hardware, or with faulty hardware. As for instance, trying to run a Basic program, which uses character sets 15 and 16, in Extended Basic.

First, let's consider the first type. The smart little TI computer makes three separate checks to be sure your program is correct. First, when you key in a program



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

You may create some really bad bugs when you try to modify a program that was written by someone else - especially if you add any new variable names or CALLs to the program. Your new variable might be one that is already being used in the program for something else, perhaps in a subscripted array. I have noticed that programmers rarely use in a variable name, so I always tack it onto the end of any variable that I add to a program. Also, the program that you are modifying may have ON ERROR routines, or a pre-scan, already built in. The ON ERROR routine was intended to take care of a different problem than the one you create, so it could lead you far astray - you had better delete that ON ERROR statement until you are through modifying. The prescan had better be the subject of another lesson, but if the program has an odd-looking command !P- up near the front somewhere, it has a prescan built in. And if so, if you add a new variable name or use a CALL that isn't in the program, you will get a SYNTAX ERROR even though there is no error. One way to solve this is to insert a line with !P+ just before the problem line, and another with !P- right after it.

When a program runs, even though it crashes or is stopped by FCTN 4 or a BREAK, the values assigned by the program to variables up to that point will remain in memory until you RUN again, or make a change to the program, or clear the memory with NEW. This can be very useful. For instance, if the program crashes with BAD VALUE IN 680, and you bring line 680 to the screen and find it reads CALL HCHAR(R,C,CH) just type PRINT R;C;CH and you will get the values of R, C and CH at the time of the crash. You will find that R is less than 1 or more than 24, or C is less than 1 or more than 32, or CH is out of range. In Extended Basic, you can even enter and run a multi-statement line in immediate mode (that is, without a line number), if no reference is made to a line number. So, you can dump the current contents of an array to the screen by FOR J=1 TO 100::PRINT A(J);: : NEXT J - or you can even open a disk file or a printer to dump it to.

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

You can temporarily stop a program at any time with FCTN 4, of course (the manual says SHIFT C, but it was written for the old 99/4), and restart it from that point with CON. Or you can insert a temporary line at any point, such as 971 BREAK if you want a break after line 970. Or, you can put a line at the beginning of the program listing the line numbers before which you want breaks to occur, such as 1 BREAK 960,970,980 Note that in this case the program breaks just BEFORE those listed line numbers. You can also use BREAK followed by one or more line numbers as a command in the immediate mode. The problem with using BREAK and CON is that BREAK upsets your screen display format, resets redefined characters and colors to the default, and deletes sprites. So, it is sometimes better to trace the assignment of values to your variables by adding a temporary line to DISPLAY AT their values on some unused part of the screen. If you want to trace them through several statements, it will be better to GOSUB to a DISPLAY AT. And if you need to slow up the resulting display, just add a CALL KEY routine to the subroutine.

Sometimes, your program will appear to be not flowing through the sequence of lines you intended (perhaps because it dropped out of an IF statement to the next line!) and you will want to trace the line number flow. This can be done with TRACE, either as a command from the immediate mode or as a program statement, which will cause each line number to print to the screen as it is executed. If used as a command, it will trace everything from the beginning of the program, so it is usually better to insert a temporary line with TRACE at the point where you really want to start. Once you have implemented TRACE, the only way to get rid of it is with UNTRACE. TRACE has its limitations because it can't tell you what is going on within a multi-statement line, and it will certainly mess up any screen display. Sometimes it is better to insert temporary program lines to display line numbers. I use CALL TRACE( ) with the line number between the parentheses, and a subprogram after



```
everything else 30000 SUB TRACE(X)::DISPLAY
AT(24,1):X :: SUBEND
```

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

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

When you are debugging a string-handling routine, don't take it for

granted that a string is really as it appears on the screen - it may have invisible characters at one or both ends. Try PRINT LEN(M\$) to see if it contains more characters than are showing; or PRINT "\*"M\$\*" to see if any blanks appear between the asterisks and the string.

There is no standard way to debug a program. Each problem presents a challenge to figure out what is going wrong, to devise a test to find out what is really happening. Don't debug by experimenting, by changing variable values just to see what will happen, etc. Even if you succeed, you will not have learned what was wrong so you will not have learned anything - and if your program contains lines that you didn't understand when you wrote them, you will have real problems if you ever try to modify the program. (Believe me, I speak from experience!)

## SHOP

All the software listed below is in stock. Prices are in Australian dollars and include postage.

Rock Runner .....	\$14.00
Waterworks .....	\$14.00
Rattlesnake Bend .....	\$ 7.75
Castle Darkholm .....	\$ 9.75
Doom Games I .....	\$ 7.75
Doom Games II .....	\$ 7.75
Doom Games III .....	\$ 7.75
Page Pro .....	\$27.00
Page Pro Pics #1 .....	\$ 8.50
Page Pro Pics #2 .....	\$ 8.50
Page Pro Pics #3 .....	\$ 8.50
Page Pro Pics #4 .....	\$ 8.50
Page Pro Pics #5 .....	\$ 8.50
Page Pro Pics #6 .....	\$ 8.50
Page Pro Pics #10 .....	\$ 8.50
Page Pro Pics #14 .....	\$ 8.50
Page Pro Pics #15 .....	\$ 8.50
Page Pro Borders #1 .....	\$ 8.50
Page Pro Borders #2 .....	\$ 8.50
Page Pro Fonts #1 .....	\$ 8.50
Page Pro Fonts #2 .....	\$ 8.50
Page Pro FX .....	\$16.50
Page Pro Headline Maker .....	\$11.50
Page Pro Headline Fonts #1 ..	\$ 8.50
Page Pro Headline Fonts #2 ..	\$ 8.50
Page Pro Templates #5 .....	\$ 7.70
Page Pro Templates #6 .....	\$ 7.50
Page Pro Templates #7 .....	\$ 7.50
Quick Run .....	\$11.00
Microsoft Multiplan .....	\$20.00
LOGO II .....	\$20.00

## TRADING POST

FOR SALE - As a complete unit, TIBUG's fully expanded system including console, modulator, transformer, Expansion box, RS232 card, Memory expansion card, Disk controller card, 2 slimline DSSD disk drives, and Extended Basic module and manual. \$450 or \$470 with free club membership. Phone Col 284 7783.

FOR SALE - Geneve 9640. Complete with manuals and lots of software. \$500. Phone Larry 07 202 1884

FOR SALE - BMC Micro Graphic Impact Dot Printer. Good second printer for labels etc. Spare ribbon. \$85. Phone Hilary 397 5926.

FOR SALE - Expanded system with two consoles and Extended Basic. Has one double sided disk drive but no RS232 cord. Two full height single sided drives included but not connected. \$275 the lot or make an offer. Phone Ross 285 2173.

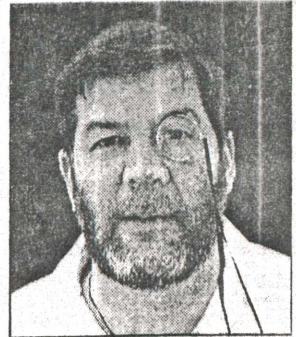
FOR SALE - Compumate CP-80 printer, 80 column, 9 pin dot matrix, 16K buffer, includes manuals, cable for TI, 2 ribbins. Quick Sale. \$150 Phone Chas on (07) 273 6254.

WANTED - FOR SALE - FOR EXCHANGE  
What do you have or need that can be listed in this column? Contact Garry or Col with details.



# Moffat's Madhouse...

by TOM MOFFAT



## GUI? Phooey!

This is a story about computers. Yes, I know that this isn't a computer magazine, but it's unlikely any self-respecting computer magazine would publish what you are about to read here. It would be seen as absolute heresy, because I intend to question the direction computer development is taking. Commercial pressures say we must move ever ahead to bigger, more sophisticated, and more expensive computer systems. But do we really need to?

Dedicated computer people have been trained (brainwashed?) to always demand the biggest and best. But you *Electronics Australia* readers are most likely 'electronikers' — realists, who see a computer as a tool instead of a status symbol. You (we) see the computer as a means to an end, instead of the end itself. Our computers are usually hooked up to something such as radio gear, or they're used to replace a typewriter, or (horrors!) to play games.

Our computers are NEVER placed on spotless desks next to a trendy pot plant, like in the current computer ads. Instead of wearing a Country Road shirt and tie and trendy braces and round wire-rimmed glasses, the guy behind the keyboard is more likely in shorts, a T-shirt, and thongs. (*Speak for yourself, Tom—I hate thongs! — Ed.*)

Instead of the slinky black woman wearing a man's suit looking over our shoulder, we probably have a matronly office worker telling us that another of our creditors is on the phone.

It seems that most of the industry sees its user base as the fancy desk and pot plant brigade, and the shorts and thongs segment doesn't exist. Or if they do, they're ignored.

Reason: they don't spend much money. They're quite happy with what they've got; it works fine for their purposes; if it ain't broke, don't fix it. So let's concentrate on selling to the pot plant brigade where there's lots of corporate or government money (yours and mine) to be spent. And what's the hottest

seller at the moment? GRAPHIC USER INTERFACE!

Buzz, Buzz! That's the sound of a hornet's nest, folks. We've just whacked it with a great big broom handle, and in a moment all hell is going to break loose. Are we about to actually CRITICISE Graphic User Interface? Yup.

Graphic User Interface is usually abbreviated as GUI. This is pronounced as 'gooey' for very valid reasons. Using it on anything other than the very latest, fastest, and most expensive computer is like tripping along through a bowl full of molasses.

GUI refers to that system where you make your program do things by pointing a mouse at a little picture on the screen and pressing a button on the mouse. You don't have to be able to type; you don't even have to know the name of the program. This is similar to going to the loo in a foreign airport. You don't have to know the word 'men' in Spanish or Arabic, you just look for the little icon of a man on the door and then you can go in there and whizz away with confidence.

The antithesis of GUI is CLI, meaning 'Command Line Interface'. Here the computer presents you with a prompt such as 'C\_', and you then use the keyboard to type in the name of your program, sometimes followed by what you want it to do.

For instance if you wanted to use your word processor called 'Edit' to work on a document called 'Story' you would type EDIT STORY and hit the Enter key, and the editor would instantly pop up on the screen with the words of STORY displayed for your perusal.

To do the same thing under GUI, you would first have to wait for the GUI program itself to come to life (slowly). You would then see several pictures or 'icons', including perhaps one of a little typewriter. You would move the mouse pointer to the typewriter icon and click the mouse button twice.

Next the word processor would be

loaded in, ON TOP OF the GUI which stays there as well (goodbye, memory). The word processor would then display a menu of files on the disk, and you would shove the mouse pointer to the word STORY and click the button again. STORY would now be loaded, ready for action.

It's interesting to note that there is much polarization over whether GUI or CLI is 'best'. You have GUI people and you have CLI people, and there appears to be little middle ground between them. You may have discovered by now that I am a CLI person, and I just can't take to GUI at all. I get confused and frustrated by mice. Other people swear by GUI's and think that people who refuse to use them are the same people who refuse to use flush toilets.

But even some very pro-GUI people are starting to have second thoughts, even in the prestigious American computer magazine *BYTE* which has been plugging GUI's for all they're worth (along with trendy pot-plant desks and slinky black women in men's suits).

One *BYTE* commentator perpetrated the ultimate outrage when he wrote "I'm getting the very uneasy feeling that GUI's are actually the biggest con job that the computer industry has ever put over on us".

Eh? How could this be? Well, there are a couple of key words in the previous paragraphs — 'slowly' and 'memory'. If you can be convinced that GUI is indispensable for you, you'll probably find it it pretty 'gooey' running on your existing computer. And if you've got a lowly one megabyte of memory that seemed gigantic when you bought the machine, you'll find that 'out of memory error' is a daily feature of your life.

How to overcome these problems? Buy a new computer! Ahh — another sale for the beleaguered computer industry, and YOU have helped Australia's economic recovery (or perhaps Taiwan's economic boom).



You now have a \$3500 computer instead of a \$1500 computer. It still runs your word processor called Edit, although a lot of flashy stuff happens in the GUI before Edit comes to life. So Edit itself now looks a bit dull.

Perhaps it's time to upgrade to Edit Version Five or whatever, written specially to run in the GUI. You spend a few hundred dollars more, you install Edit Version Five only to find that it now uses twice as much space on your hard disk as the earlier version. And as well, it runs SLOWER!

Golly! Well, the salesman says that to speed up your program you're going to need an even faster computer. And as well, all this new GUI software demands a lot more memory and more disk space. So let's upgrade to a \$5000 computer with a double-sized hard disk and six megabytes of memory. Now you find that Edit Version Five runs much faster — almost as fast as Version One did on your \$1500 computer. And that, my friends, is 'progress'.

As for that speed question, there are certain advantages in some applications. For instance desktop publishing requires an awful lot of things to happen in a big way, and it seems to take forever on an earlier 'AT' class computer. Same goes for computer-aided design work.

If you're designing a house or a bridge, you frequently want to 'regenerate the screen' to get an overall look at what you've done so far. This involves lots of internal calculation, and it's pretty frustrating to sit there as the new picture dribbles onto the screen line by line. Here a fast computer certainly is justified.

Most of the software I use is small and blindingly fast, particularly the 'Video Display Editor' word processor I use for all my magazine writing. Most of this is done on my trusty and horribly obsolete Toshiba XT-class laptop. This is just like my trusty and horribly obsolete Holden station wagon; I intend driving them both until they fall apart.

The Video Display Editor does its stuff in the blink of an eye on the XT laptop. It doesn't appear any faster on a '386 machine, because you can't get much faster than the blink of an eye!

The occasional speed problem does crop up. I've recently started using a big mathematics program to calculate the orbits of satellites for a weather satellite project I'm working on. You can tell it to work out the dates and times of all the passes of the NOAA 9, 10, and 11 satellites, receivable in Tasmania, for the next week.

The program then goes away and

cogitates; waiting for a result is like watching grass grow. Buy a new computer? No, I just set the laptop working and then come back after lunch to collect the finished product. No time lost there at all.

Maybe we've been a bit coy about brand names here. The original GUI came from Apple Computer as the operating system for the ill-fated Lisa, and later for the popular Macintosh. IBM decided they had to have a 'point-and-click' system to match the Mac, so Microsoft came up with a system called *Windows*. In the Amiga world, the GUI is called the *Workbench*.

In the IBM and Amiga at least it is quite easy to short-circuit the GUI and revert straight to a CLI system. This is the first thing I learned when I got an Amiga to do project development on — how to nobble the Workbench. On the IBM you can get rid of the GUI by simply not starting it up. Apparently this happens quite a lot. A recent statistic pointed out that only one third of people who own *Windows* actually use it.

*Windows* is now 'bundled' with many new computers; in other words you get it for free, as part of a package deal. Is this generosity, or is there some hidden motive? Is it intended that new down-market 'AT' machine owners get the message that *Windows* and all its application programs would run oh-so-much better if they upgraded to a 386 or 486 computer?

Perhaps it's a bit like supplying a free sample of heroin with a purchase of marijuana, to get you hooked on something a bit more flash.

Applications! Well, *Windows* exists, so we MUST have applications to use it. Big fat sluggish applications, offspring of things that were so slick and elegant in their DOS/CLI versions.

I heard the other day that the popular and excellent PCB design packages from Protel would be henceforth available in *Windows* only. I was on the phone to them in a shot, to learn that yes, future product development was *Windows*-based only, but they certainly weren't going to kill their existing CLI-based programs like *Autotrax* and *Easytrax*. They're going to wait and see what the market does.

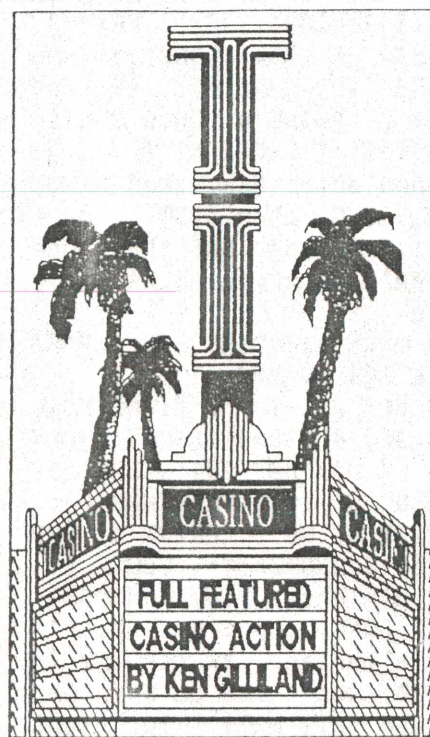
As for myself, I've got Protel *Easytrax* and *Schematic* running on my AT-class computer, and I wouldn't use anything else for PCB and circuit design work. A mouse sits nearby, idle, because I prefer to drive the editing cursor around with the cursor keys instead of a mouse.

I'm a perfectionist and I like to physically click the keys six times to move

the cursor six tenths of an inch along the PCB. Then I KNOW it's where I want it. I guess it gives me a feeling of security.

*Windows* does have certain advantages for program developers; for instance there is a standard printer interface so they don't have to write drivers for every printer in the land. But developers shouldn't forget the guys in shorts and thongs who populate the small businesses of this world.

I know for a fact that lots of excellent design work comes out of 'obsolete' computers in garages and back rooms, and there's no way those people are going to jump out and buy a new computer just to run new software. They'll quite happily stick with what they've got. If it ain't broke, don't fix it! ♦



Tired of saving the universe and slaying dragons with your joystick? Want to go to a place where the action is always hot and your credit is always good? Well then, NOTUNG Software has just the place for you... TI CASINO.

TI Casino is a collection of eight Casino games, all interlinked, so you can play any of the games with the same money. Each game is virtually joystick operated with no confusing keypresses. At the end of your gambling session, take your winnings to the cashier and get a printed check!

ACEY DEUCEY ♦ BLACKJACK ♦ BACCARAT ♦ CRAP TABLES  
KENO ♦ DRAW POKER ♦ ROULETTE ♦ SLOT MACHINES

Blackjack includes Doubling down, Splits & Insurance, there's unlimited betting in Roulette and on the Crap Tables, and above all, be sure to dine at the Green Parrot Restaurant while playing Ken!

\$15 through NOTUNG Software

Specify SSSD or DSSD Version, Joysticks, 32k & ExtBasic Req'd, Printer Opt'l



# TIPS FROM THE TIGERCUB

#39

Copyright 1986

TIGERCUB SOFTWARE  
156 Collingwood Ave.  
Columbus, OH 43213

Answer to last month's challenge - for the longest possible one-liner, run the following "program to write a program" -

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

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

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

Again enter NEW, and MERGE DSK1.LONG, then RUN. You'll get a message BREAKPOINT IN 32510 (don't ask me why! Can anyone tell me?) but just enter RUN again. Then LIST it - over 24 lines long!

Explanation? Programs are saved in token code similar to MERGE format code. The maximum length of a record is 163 bytes - which is why MERGE files are D/V 163. The token for RANDOMIZE is ASCII 149, for the double colon is 130. Repeating that 79 times takes only 158 bytes, plus one more RANDOMIZE, the two-byte tokenized line number and the mandatory ASCII 0 to end the record, totals 162.

Here's a spooky one for Hallowe'en -

```
100 CALL CLEAR :: CALL MAGNI
FY(4):: CALL SCREEN(2) ! The
Blob by Jim Peterson
110 CALL CHAR(96,RPT$("3C7EF
FFFFFF7E3C",4)):: J=-1
120 FOR L=1 TO 28 :: CALL SP
RITE(#L,96,16,L*4+20,10,0,L+
8):: NEXT L
130 FOR L=1 TO 28 :: CALL MO
TION(#L,0,L*J):: NEXT L
140 J=J*-1 :: GOTO 130
```

Wes Johnston published an unusual sprite 2-liner in the Charleston Area 99ers newsletter. It is based on a CALL LOAD which freezes all sprite motion until they are turned loose by another CALL LOAD -

```
100 R=PI*2/28 :: CALL CLEAR
:: CALL SCREEN(2):: CALL INI
T :: CALL LOAD(-31806,96)::
FOR I=1 TO 28 :: CALL SPRITE
(#I,46,16,96,128,COS(I*R)*10
,SIN(I*R)*10):: NEXT I
110 CALL LOAD(-31806,0):: GO
TO 110
```

You might like to try adding my "jewels" to that -

```
100 FOR CH=33 TO 60 :: FOR A
=1 TO 4 :: X=INT(8*RND+1)::
T$=SEG$("18243C425A667E81",X
*2-1,2):: A$=A$&T$ :: B$=T$&
B$ :: NEXT A :: CALL CHAR(CH
,A$&B$):: A$,B$="" :: NEXT C
H
110 R=PI*2/28 :: CALL CLEAR
:: CALL SCREEN(2):: CALL INI
T :: CALL LOAD(-31806,96)::
FOR I=1 TO 28 :: CALL SPRITE
(#I,32+I,INT(14*RND+3),96,12
8,COS(I*R)*10,SIN(I*R)*10)::
NEXT I
120 CALL LOAD(-31806,0):: GO
TO 120
```

Also try CALL MAGNIFY(2)

And, here is a companion program to the TAKE AWAY in Tips #35 -

```
100 CALL CLEAR :: CALL TITLE
(5,"ADD & CARRY")!by Jim Pet
erson
110 DISPLAY AT(3,10):"COPYRI
GHT":TAB(10);"TIGERCUB SOFTW
ARE":TAB(10);"FOR FREE":TAB(
10);"DISTRIBUTION":TAB(11);"
```

```
SALE PROHIBITED"
120 CALL PEEK(-28672,A@):: I
F A@=0 THEN 160
130 DATA FINE,NO,GOOD,UHOH,R
IGHT,TRY AGAIN,YES,THAT IS N
OT RIGHT
140 FOR J=1 TO 4 :: READ RIG
HT$(J),WRONG$(J):: NEXT J
150 FOR D=1 TO 1000 :: NEXT
D :: CALL DELSPRITE(ALL)
160 CALL CLEAR :: CALL CHAR(
95,"FFFF"):: CALL MAGNIFY(2)
:: RANDOMIZE :: CALL SCREEN(
14):: FOR SET=5 TO 8 :: CALL
COLOR(SET,16,1):: NEXT SET
170 CALL CHAR(120,"E70042001
8007E0000E700420099423CE7004
20099423C00E7004218003C4200"
)
180 CALL CHAR(124,"0E0004010
007080070000208000E01000")
190 DISPLAY AT(3,8):"ADD AND
CARRY" :: CALL CHAMELEON
200 CALL COLOR(14,2,2):: CAL
L HCHAR(4,4,143,2):: CALL HC
HAR(5,4,143,2):: CALL SPRITE
(#25,120,11,25,25)
210 T=T+1 :: IF T=6 THEN T=0
:: GOTO 250
220 Z=INT(8*RND+2):: IF Z=Z2
THEN 220 ELSE Z2=Z
230 Y=INT(Z*RND):: IF Y=Y2 T
HEN 230 ELSE Y2=Y :: X=Z-Y
240 N=1 :: GOSUB 470 :: GOTO
210
250 T=T+1 :: IF T=11 THEN T=
0 :: GOTO 290
260 X=INT(10*RND):: IF X=X2
THEN 260 ELSE X2=X
270 Y=INT(10*RND):: IF Y=Y2
OR X+Y<10 THEN 260 ELSE Y2=Y
:: Z=X+Y
280 N=1 :: GOSUB 470 :: GOTO
250
290 T=T+1 :: IF T=11 THEN T=
0 :: GOTO 330
300 X=INT(90*RND+10):: IF X=
X2 THEN 300 ELSE X2=X
310 Y=INT(90*RND+10):: IF Y=
Y2 THEN 310 ELSE Y2=Y :: Z=X
+Y
320 N=2 :: GOSUB 470 :: GOTO
290
330 X=INT(900*RND+100):: IF
X=X2 THEN 330 ELSE X2=X
340 Y=INT(900*RND+100):: IF
Y=Y2 THEN 340 ELSE Y2=Y :: Z
=X+Y
```



```

350 N=3 :: GOSUB 470 :: GOTO 330
360 R=96 :: CC=96 :: FOR J=1 TO N :: CALL SPRITE(#J,48+A(J),11,R,CC):: CC=CC+16 :: NEXT J
370 R=116 :: CC=96 :: FOR J=1 TO N :: CALL SPRITE(#4+J,48+B(J),11,R,CC):: CC=CC+16 :: NEXT J
380 CALL HCHAR(18,12,95,N*3) :: CC=CC-16 :: CALL SPRITE(#22,43,16,R,80):: RETURN
390 R=140 :: FOR J=LEN(STR$(Z)) TO 1 STEP -1 :: CALL SPRITE(#20,63,11,R,CC)
400 CALL KEY(3,K,ST):: IF ST<1 OR K<48 OR K>57 THEN CALL PATTERN(#20,32):: CALL PATTERN(#20,63):: GOTO 400
410 CALL DELSPRITE(#20):: CALL SPRITE(#12+J,K,11,R,CC)
420 IF K-48<>C(J) THEN GOSUB 480 :: CALL DELSPRITE(#12+J) :: CALL SPRITE(#20,63,11,R,C) :: GOTO 400
430 IF A(J-W)+B(J-W)>9 THEN CALL SPRITE(#28,49,16,80,CC-16)
440 CC=CC-16 :: NEXT J :: GOTO 510 :: RETURN
450 FOR J=1 TO LEN(STR$(X)) :: A(J)=VAL(SEG$(STR$(X),J,1)) :: NEXT J :: FOR J=1 TO LEN(STR$(Y)) :: B(J)=VAL(SEG$(STR$(Y),J,1)) :: NEXT J
460 FOR J=1 TO LEN(STR$(Z)) :: C(J)=VAL(SEG$(STR$(Z),J,1)) :: NEXT J :: W=LEN(STR$(Z))-LEN(STR$(X)) :: RETURN
470 GOSUB 450 :: GOSUB 360 :: GOSUB 390 :: FOR D=1 TO 200 :: NEXT D :: CALL DELSPRITE(ALL):: DISPLAY AT(18,1):: CALL CHAMELEON :: CALL SPRITE(#25,120,11,25,25):: RETURN
480 DATA 123,124,125,123,124,125,123,120
490 IF A@=0 THEN 500 :: CALL SAY(WRONG$(INT(4*RND+1)))
500 RESTORE 480 :: FOR JJ=1 TO 8 :: READ P :: CALL PATTERNRN(#25,P):: XX=2^250 :: NEXT JJ :: RETURN
510 DATA 121,122,121,122,121,122
520 IF A@=0 THEN 530 :: CALL SAY(RIGHT$(INT(4*RND+1)))
530 RESTORE 510 :: FOR JJ=1 TO 6 :: READ P :: CALL PATTE
RN(#25,P):: XX=2^250 :: NEXT JJ :: RETURN
540 SUB CHAMELEON
550 M$="1800665AC342DB667E188100995AC3A5E78142BD24DB660081429924007E5AC3A53C241800FFDB5AFF7EFF0099188100660018"
560 RANDOMIZE :: CALL CHAR(128,SEG$(M$,INT(43*RND+1)*2-1,16)) :: X=INT(14*RND+3)
570 Y=INT(14*RND+3):: IF Y=X THEN 570 :: CALL COLOR(13,X,Y)
580 CALL HCHAR(1,2,128,30):: CALL HCHAR(24,2,128,30):: CALL VCHAR(1,31,128,96):: SUBEND
590 SUB CHAMWIPE
600 T=T+1+(T-2)*2 :: ON T GOTO 610,620
610 CALL VCHAR(1,3,128,768) :: GOTO 630
620 CALL HCHAR(1,1,128,768)
630 CALL CLEAR :: SUBEND
640 SUB TITLE(S,T$)
650 CALL SCREEN(S):: L=LEN(T$):: CALL MAGNIFY(2)
660 FOR J=1 TO L :: CALL SPRITE(#J,ASC(SEG$(T$,J,1)),J+1-(J+1=S)+(J+1=S+13)+(J>14)*13,J*(170/L),10+J*(200/L)) :: NEXT J
670 SUBEND

A mathematical curiosity -
100 !MAGIC NINES by Jim Peterson
110 CALL CLEAR
120 INPUT "TYPE ANY 3-DIGIT NUMBER OF 3 DIFFERENT DIGITS":N :: IF N<>INT(N) OR N>999 OR N<0 THEN 120
130 N$=STR$(N):: IF N<100 THEN N$="0"&N$
140 IF SEG$(N$,1,1)=SEG$(N$,2,1) OR SEG$(N$,1,1)=SEG$(N$,3,1) OR SEG$(N$,2,1)=SEG$(N$,3,1) THEN PRINT ">>>THREE DIFFERENT DIGITS<<," :: GOTO 120
150 PRINT :: N2$="" :: FOR J=1 TO 3 :: N2$=SEG$(N$,J,1)&N2$ :: NEXT J :: N2=VAL(N2$) :: D=ABS(N-N2)
160 PRINT N$;" BACKWARDS IS";N2$:
170 N3=ABS(N-N2):: N3$=STR$(N3):: IF N3<100 THEN N3$="0"&N3$
180 IF N>N2 THEN PRINT N$;"
MINUS ";N2$;" EQUALS ";N3$:
:ELSE PRINT N2$;" MINUS ";N3$;" EQUALS ";N3$:
190 FOR J=1 TO 3 :: N4$=SEG$(N3$,J,1)&N4$ :: NEXT J
200 PRINT N3$;" BACKWARDS IS";N4$:"N3$;" PLUS ";N4$;" IS 1089": "I KNEW THAT WOULD BE THE": "ANSWER!": "LIST THE PROGRAM AND SEE!"
210 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
220 ! THE ANSWER WILL BE !
230 ! 1089 !
240 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

100 DISPLAY AT(8,10)ERASE ALL:"SHENANDOAH": : : " Across the wide Missouri": : : : : : "programmed by Jim Peterson"
110 FOR D=1 TO 1000 :: NEXT D :: CALL CLEAR :: DIM S(24) :: RANDOMIZE :: M$="4218005A007E9981005A24DBC31824243C5A7EA56618003CDB66BD3CA542187E5AC324425A18A51866810081187E423CBDDBC3" :: R=1
120 FOR CH=40 TO 136 STEP 8
130 CALL CHAR(CH,SEG$(M$,INT(43*RND+1)*2-1,16)) :: CALL HCHAR(R,1,CH,64):: R=R+2*ABS(R<23)
140 NEXT CH :: R=0 :: FOR SE T=2 TO 14 :: X=INT(14*RND+2)
150 Y=INT(14*RND+2):: IF Y=X THEN 150
160 CALL COLOR(SET,X,Y)
170 NEXT SET :: CALL CLEAR : : CALL COLOR(1,5,5):: CALL VCHAR(1,29,1,192):: CALL SCREEN(16):: F=262 :: FOR N=0 TO 23 :: S(N)=INT(F*1.059463094^N):: CALL SOUND(-999,S(N),0)
180 NEXT N
190 DATA 2,1,1,1,6,1,1,1,6,2,6,1,1,1,6,1,8,8,1,10,10,1,1,1,11,1,15,6,3,13,6,2,13,11
200 DATA 1,18,10,1,17,17,4,1,5,11,1,11,15,1,13,13,1,15,11,1,13,13,1,10,10,3,13,10
210 DATA 2,13,13,2,13,10,1,1,5,10,1,10,15,2,15,15,1,15,10,1,10,10,1,13,13,1,10,10
220 DATA 1,8,3,3,6,3,2,6,6,2,8,8,4,10,1,1,10,6,1,6,6,1,1,0,10,1,15,15
230 DATA 2,13,1,2,13,5,2,13,10
240 DATA 1,6,6,1,8,8,6,10,6,

```



```

2,3,3,2,8,5,1,8,1,3,6,1,7,6, +1)/2)*8,T*4):: CALL HCHAR(A T=2 TO 14 :: X=INT(15*RND+2)
1 +1,E,32+INT((A+1)/2)*8,T*4): 310 Y=INT(15*RND+2):: IF Y=X
250 A=1 :: B=1 :: E=5 : CALL HCHAR(B,E,32+INT((B+1 THEN 310
260 FOR J=1 TO 144 STEP 3 :: )/2)*8,T*4) 320 CALL COLOR(SET,X,Y):: CA
CALL HCHAR(A,E,32,T*4):: CA 280 CALL HCHAR(B+1,E,32+INT( LL SOUND(-999,S(6),LL,S(1),L
LL HCHAR(A+1,E,32,T*4):: CAL (B+1)/2)*8,T*4):: FOR D=1 TO L):: LL=LL+2
L HCHAR(B,E,32,T*4):: CALL H T :: CALL SOUND(-999,S(A),0 330 NEXT SET :: RESTORE :: G
CHAR(B+1,E,32,T*4):: READ T, S(B),7) OTO 260
A,B :: E=17-T*2 290 NEXT D >>>>>>>MEMORY FULL<<<<<<<<
270 CALL HCHAR(A,E,32+INT((A 300 NEXT J :: LL=0 :: FOR SE

```

# BASIC COMPUTERS

What did he say?

by Garry Christensen

Hello again, and welcome to the final article in this series about the inside of the TI computer. In the earlier articles I have described the different types of memory and how the CPU makes use of memory, how the CPU 'runs' a program, and how information and programs can be moved into and out of the computer. The last article specifically examined the CRU and how this chip helped control the devices (keyboard and disk drive etc) that allow the computer to communicate with the outside world.

This month I would like to briefly look at the computer as a whole, with a little emphasis on software, and close by talking about some of the enhancements that are available for the TI.

When you switch the computer on, there is some electronics that reset the CPU. This is done by holding a particular line at 0 for a short time (in computer time frame). When the CPU is reset, it looks at the the first two words in memory, >0000 and >0002, to see what value to place in the program counter (remember that there are 2 bytes in a word). The adress space from >0000 to >2000 is dedicated to ROM so the values never change and the program is always there.

The power-up sequence is quite simple. First, turn off the sound generators because these could be in any state. This is a good clue when we are repairing TI99/4As. If there is a howling noise when it is switched on, then even the first instructions are not being executed. This means that either the CPU is no good (often), the ROM is defective (rare), or there is something that is stopping the data from being transferred from the ROM

chips to the CPU. This could be a defective chip that is keeping some data or address lines from operating.

The next thing to happen is the screen is cleared and the colour set to cyan. Then the CPU enables each card in the PE box, using the CRU, and checks to see if there is any setting up that they need to do. It then goes on to display the title screen that you all know so well.

All this is not in the ROM though. Texas Instruments was able to fit a lot of programming into a little memory space by using Graphics Programming Language (GPL). This is the code that is stored in GROM. In the ROM (>0000 to >2000) is what is called the interperator. Most of ROM is the code for performing specific functions, like moving the data in one part of memory to another, or adding the values in two memory locations. For those of you who have done some programming in Extended BASIC, you can consider them like GOSUBS.

The data in GROM is a list of which subroutines to perform, sort of like a recipe I suppose. The CPU reads the byte from GROM, decodes it to see which subroutine to use, gets any further data from GROM if necessary, then executes that subroutine. When that one is done, it gets the next byte from GROM and does that one.

By way of an example, instead of writing the machine code to clear the screen each time, in GPL there need only be a single command. That command is decoded each time the screen needs to be cleared and the routine is performed. In this way, all of the code needed to run BASIC programs could fit into 12K of GROM. A big memory saving. There is a down side though. Speed. Why is BASIC so slow when compared to assembly?

Lets take a line in BASIC say CALL SOUND(100,110,0). To make this work the



computer must look at each letter of the statement, C-A-L-L {it now knows that this will be an internal command} -S-O-U-N-D {it's going to be a sound generation statement} -(-1-0-0-, {duration will be 100 so set the counter} -1-1-0-, {frequency will be 110 Hz so calculate the code to load onto the sound chip} -0-) {volume setting 0 so calculate the code to load onto the sound chip}, now write the values to the sound chip and start decrementing the counter for the time delay and when it reaches 0 get the next BASIC statement.

That is a much simplified form of what actually happens but you get the idea. The process is called interperating the statement, and BASIC is an interperated language. We write it in a language we can understand and it has to be converted to machine code ( the computer's language) by a program. To execute a BASIC statement the 'interperator' works out what you want done then does it. That is why you need everything in the right place otherwise it will not know what you want. Remember "SYNTAX ERROR IN LINE ...".

It is the code in the GROMs that is the interperator. It reads your BASIC programs and does what you ask it to. Now here's the crunch - the interperator is written in an interperated language. The description that I gave earlier of clearing the screen shows that GPL must be interperated as well. The GPL interperator is a section of machine code in the ROM.

Try to visualize what is happening when you enter a CALL SOUND statement. The CPU is working like crazy to interperate the GPL commands, which are working just as hard to interperate your BASIC statement. GPL gave Texas Instruments the opportunity to provide a very powerful version of BASIC but the cost was speed.

How can we get it to go faster. 1 - a faster computer like the Geneve or 2 - bypass the interperator process. To do this we need a compiler. A compiler takes a statement and converts it to the machine code that is necessary to perform that function. The machine code is then stored on disk so that it can be loaded and executed directly by the CPU. The compiler means that the statement need only be decoded once and that is done before the program is even started.

So where is the BASIC compiler? There have been a number of rumours over the years but nothing ever came about. Maybe one day.

When it comes to add-ons, the device that has had the most impact has probably been the RAMdisk. These have given speed and large volume storage to the TI computer by simulating a floppy drive but using memory to store the data instead of disk. You have the option of setting the CRU address on most RAMdisks so you can select which address they use. When that bit is set to 1, the card is enabled. It operates much like the disk controller except that instead of the data coming from disk to a port where it can be read, the RAMdisk moves blocks of memory into the address space. This is called 'paging'. Each block of memory is called a page and after selecting the correct page, the data can be simple transferred into normal memory.

Before programs like MENU and BOOT came along, it was common to set a RAMdisk up to have the name "DSK1" and to set it's CRU address to >1000. When the computer starts looking for a particular device, it starts at CRU address >1000 and turns on that card. If the name is the one that it is looking for then it carries on with what it has to do. If it isn't the correct name, the card is turned off (reset the bit to 0) and the next one turned on, using steps of >100. Last month I mentioned that the disk controller was at CRU address >1100 so in this instance, the RAMdisk would be checked for the name DSK1 before the floppy drive. This system was used to get the computer to access the RAMdisk instead of floppy.

The concept of paging is not unique to the TI. Most computers use it to access large amounts of memory. The original 128K memory expansion cards for the TI had 4 pages of 32K. You could select which page of memory expansion that you wanted to use. The idea was good but it did suffer from one big problem - your program had to be in one page of the memory. When you change pages, your program is gone.

The 9640 has a better method of using all its memory. It divides the 64K of address space into 8 blocks of 8K each. There is 8 bytes from >8000 that control which page of memory appears in each block. The pages are numbered from >00 to >FF



(256). A bit of quick maths will reveal that 256 pages x 8K per page gives a total of 2 Meg. It also has the ability to put the same page of memory into 2 different blocks. That means that you can change a byte at one address and the corresponding byte at another address will also change. I can't think of any use for this but its an interesting point for more thought.

The 128K card and the 9640 have one problem in common. There is very little software that will make use of the extra memory. The most common use for the memory in the 9640 is to set it up as a temporary RAMdisk.

Finally I would like to mention 80 column devices. These are devices that use more recent video chips to give a higher resolution output to the screen. They support the lower resolution modes so they are compatible with all of the old programs and there are more and more programs that are coming out for the 80 column mode. In the case on the TIM from OPA, they actually replace the old VDP chip, but others simply respond to the same addresses that the VDP chip uses and essentially run in parallel to the old chip. They need much more memory and also make use of memory pages.

Well that just about wraps it up. I hope that through this series I have been able to explain a little of what happens in your computer and I have tried to keep it as simple as possible. It's hard not to get a bit technical at times and I will admit to introducing some minor inaccuracies in order to hold things down to a reasonable level. If you have any questions or problems, please don't hesitate to contact me.

.....

## IN THE P.O. BOX

LA99, May 1992: XB Miscelany, Call Waiting and Your Modem, Touch Tone Frequences, Comparing Health Insurance, Ramblin' Thoughts from the President, Managing your Money, Routine Interweaving.

TI Shug (Sydney), May 1992: Editor's Comment, Co-ordinator's Report, Treasurer's Report, Genealogical Data Base, Secretaries Note Book, Assembly Class, Letter to the Editor, Shop, Software Column, Games Information, Techo Time, To See or Not to C, TI-Bits, XB Tips, Why I Write in Machine Code. Using Arrow Keys in XB Programs,

Reformatting, Sorting, Decoding EPROM Files, TIA/Link, Riemann Sphere Graphic Program (TML), Appending to DV 80 Files, Max-RLE, Beginning Forth.

Micropendium, January 1992: Comments, Feedback, Learning German (B), Wallet Automation (XB), File Handling (Ass), Tigercub Calculator, Reviews of Dumpit Disk of Pyrates, Bride of the Disk of Dinosaurs, and Disk of Horrors, Newsbytes, Repairing XB cartridge, Planning for your Retirement (XB).

Micropendium, April 1992: Comments, Bugs and Bytes, Feedback, Myarc Working with Consumer Office, Miller to Coordinate MDOS Buyout, Raglan Pullover (B), TI and PC Basic Comparisons (XB), Newsbytes, Micropendium Index, Video XOP 6 and My-Basic, Structured Data (Ass), Review of Artist Slide Show, Casino Games, and Harrison Software Word Processor, User Notes.

.....

## DSK.DISKNAME.FILENAME

(Written for the Swedish user group PROGRAMBITEN 90-6)

by Jan Alexandersson, Springarv(gen 5, S-142 61 TR]NGSUND, Sweden

It is possible to call a file by DSK.DISKNAME.FILENAME instead of DSK1.FILENAME. The program will start the search on DSK1 for a disk named DISKNAME. If this disk is not found on DSK1 the search will continue on DSK2 and so on until the disk is found and then the file FILENAME is loaded from that disk. I find this search from drive to drive very slow so the only use is for disks that should be put in DSK1. This is very useful if you only have one drive, hard disk with DSK emulation or RAM disk on CRU >1000.

If you have a TI controller (on CRU >1100) then the search from Basic will begin with any RAM disk (if present) on CRU >1000 and then in the order of numbers before an error message is given: DSK1 - DSK2 - DSK3 - I/O ERROR 57. RAM disk DSK9 on CRU >1200 or higher will not be reached.

I have a Myarc HFDC (DSK5-8) on CRU >1000 and a TI controller (DSK1-3) on CRU >1100 which give the following search order: WDS1 - DSK5 - DSK6 - DSK7 - DSK8 -







# Page Pro 99

and related utilities

by Bob Relyea

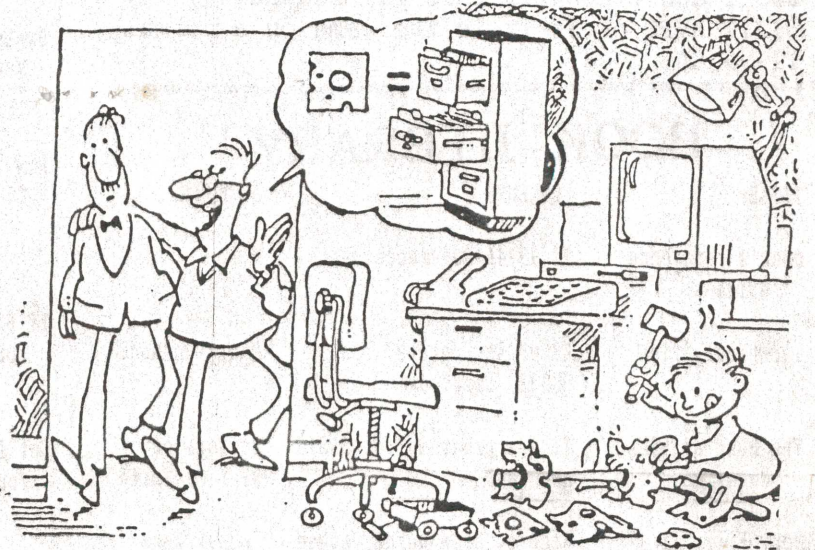
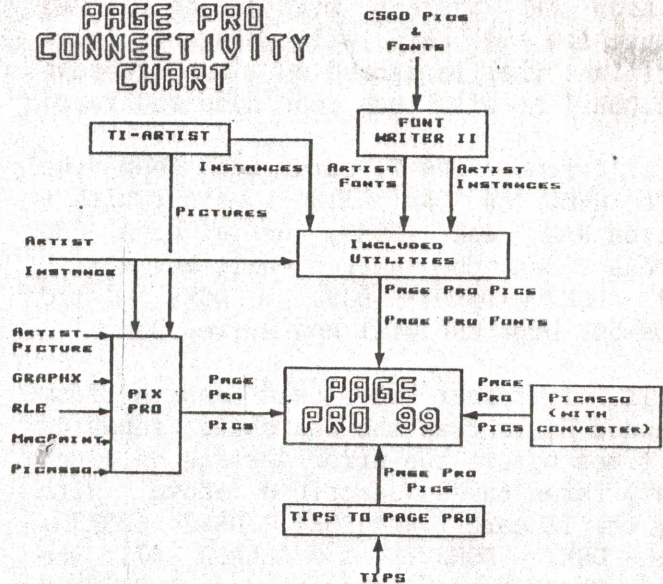
Courtesy of TISHUG

Many nice graphics programs have been written for our computer. Not everybody is aware of the fact that, even though they are not all compatible, there are many conversion programs available to enable the work of one program to be utilised by one of the others. Pix Pro is one of the best examples of that. Page Pro is one of the best graphics programs ever written for the TI-99/4A and the table below is the best that I have seen that shows how Page Pro can be used in conjunction with other graphics programs. The idea was taken from the Page Pro Times, Fall 1991, page 5 and it was redrawn with the Page Pro program. It clearly illustrates the various popular formats found on the TI-99/4A, and the utilities necessary to get them into Page Pro 99 format. I trust that it is of use to somebody.

If you do not have a copy of Page Pro, I suggest that you get one. Apart from Funnelweb & related utilities, I use this program more than any other one that I have. If you are a student or teacher or just interested in graphics, this program is a must. I used it many times last year just helping out my kids make title pages up for school work alone. They looked great! One feature that Page Pro has that you may not be aware of is the ability to use more than one font on the same page. If you are not using a very large part of the page, then to use more than one font is relatively quick. If you have a full page then it is slow. There is virtually no limit to how many different fonts (i.e. small, large and line fonts) that can be used on the same page. Once you have used all you want of one font then use Fctn 7 to initiate the process of saving your work in 'picture' format (explained in the utilities manual). This is a way of saying that that part of the page has now been 'permanently fixed'. So, if you call up this newly-saved picture you can do more work on it with other fonts without altering what you have already done, and so the process can be repeated to your heart's content.

It is an easy utility to use and the package sold under the title Page Pro 99 has other utilities with it such as printing programs in double column format. With more people upgrading their computers to 80 columns, it is of interest that some of the Asgard utilities, such as Banner Maker, have 80 column capabilities.

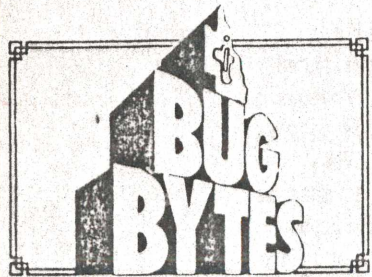
## PAGE PRO CONNECTIVITY CHART



PAGE 22

BUG-BYTES

MAY 1991



TI BRISBANE USER GROUP  
P.O. BOX 3051  
CLONTARF M.D.C.  
QLD AUST 4019.



use

43  
GRAEME MORRIS  
P.O. BOX 371  
CLEVELAND QLD 4163