HUNTER VALLEY
99ERS
USERS GROUP
HOME COMPUTER NEWSLETTER

A VERY SPECIAL

HUNTER VALLEY

WELCOME

Bob Carmany

REGISTERED BY AUST POST
PUBLICATION No. NBG8023

FEBRUARY 1990

YOUR COMMITTEE

PRESIDENT
Peter Smith
8 Glebe St.
East Maitland 2322
Phone 336164 V/tl 493261640

VICE PRESIDENT
John Paton
1 Parlen Close
Rutherford 2320
Phone 326014 V/tl 493260140

SECRETARY
Brian Woods
9 Thirlmere Pde.
Tarro 2322
Phone 662307 V/tl 493223070

TREASURER
Noel Cavanagh
370 Morpeth Rd.
Morpeth 2321
Phone 333764

SOFTWARE LIBRARIAN
Stewart Bradley
14 Hughes St.
Birmingham Gardens 2287
Phone 513246

EDITOR
Allen Wright
77 Andrew Rd.
Valentine 2280
Phone 468120

PURCHASING/HARDWARE CO-ORDINATOR
Alan Franks
822 Pacific Highway
Marks Point 2280
Phone 459120

SOCIAL SECRETARY
Robert (Bob) MacClure
75 Deborah St.
Kotara South
Phone 437431

PUBLICATIONS LIBRARIAN
Ken Lynch
9 Hall St.
Edgeworth 2285
Phone 585983

COMMITTEE MEMBERS
Don Dorrington
36 Nelson St.
Barnsley 2301
Phone 531228

Tim Watkins
36 The Ridgeway
Bolton Point 2283
Phone 592836

CONTRIBUTIONS

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

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

Please include along with your article sufficient information to enable the file to be read by the Editor eg. File name etc. The preferred format is 75 columns and page length 66 lines, right justified.

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

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

DISCLAIMER

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

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

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

From President Peter's Quill

Welcome back to all those , who like me had a great break over the summer vacation period. I trust you are refreshed and pounding those little TI keys at a great rate of knots.

Of major interest this month will be the visit of our proliffic writer from U.S.A. Bob Carmony. As you would be aware from Brian's note sent to you we have a range of activities planned for not only him, but us.

These activities, although aimed at welcoming Bob and showing off our lovely valley to him, will also serve to tie our members together in a closer bond and make the support which we give oneanother for our orphan, so much more valuable and easily gained.

Please support these activities to your fullest.

A number of interesting articles and reviews are enclosed in this magazine, however I must congratulate all who contributed to our bumper Christmas edition. What a bonanza. Special mention must be made of our new and in some cases, unknown contributors. Keep up the good work and don't be timid.

Discussion was held at the last committee meeting regarding our involvement in the Newcastle CAE (now part of the Uni.) Computer show and it was suggested that a well organised and specific display be mounted this year, showing just what our TIs are used for.

Your thoughts, suggestions and help in organising our stand would be gratefully accepted. Remember the show is just an excuse for us to show off our machines and let us as members of a club, work together with our computers to support one another and possibly a few who are not even lucky enough to belong to our club.

During my vacation on the north coast, I was privileged to be able to assist a young mother and her four children (yr7 to 4th class) to learn a little about a little grey orphan which "Grandad" had given the family.

If you can remember back to those first few days you had your machine and the joys and frustrations it brought you, you would know what those people were experiencing. It certainly made me realise what a debt I owe to our mighty little machine for the knowledge it has given me and the frienships it has caused.

I hope others can look back just as fondly on their machines.

Peter

Ron Klienschaffer

DEBUG A DEBUG

Reports have come in that the fix for the QED Module as published in the December issue of this newsletter MAY cause lockup after the modification in SOME consoles, this can be overcome by using two 1N914 silicone diodes in paralell instead of the 1N34A (OA91) germanium diode. The use of the silicone diodes does not seem to affect the operation of the module including keeping the Nicad battery charged. We are indebted to Gary Wilson for this information and after chewing the problem over with Gary we feel that the slower switching time of the germanium diode doesnt allow the Module to become fully operational before the console system Groms lock it up, anyway if any problems are encountered using the silicone diodes will effect a cure.

Brian Woods REPORTS From the Secretary's Desk

WELCOME BOB

After writing for the HV99ers for so long & corresponding on a regular basis with members of the Group, it's great to finally get the chance to meet Bob Carmany in person during his trip to Australia. Hope you have a good time during your stay here & in Brisbane. A fairly full(!!) program has been arranged to keep Bob amused during his stay in Newcastle, so we all hope you take home pleasant memories of your stay with us.

NEW PRODUCTS

From the December issue of The PUG Peripheral, the newsletter of the Pittsburgh Users Group comes the following details...

"Asgard Software has released `Pix Pro'. This program will allow you to load 8 different picture formats & save them in any of 6. It allows you to take files stored in Picasso, Graphx, TI-Artist Picture, TI_Artist Instance, Page Pro, MacPaint, RLE or PIX formats & save them as Graphx, TI-A Picture or Instance, Picasso, Page Pro or PIX pictures."

"The idea behind PIX PRO is that you shouldn't have to change programs because the pictures you want to use won't load into the program you want to use. This program, written entirely in Assembly, does the work of over 40 different conversion programs. You can order it from:
Asgard Software
PO Box 10306
ROCKVILLE MD 20849 USA

"It costs $US14.95 plus 75 cents for shipping & handling."

Also from The PUG Peripheral, January issue...

"Good news for Adventurers! Asgard Software has negotiated a contract to market 7, count them 7, Infocom games that were never released for the TI. All of these new games will be sold with original Infocom packaging and documentation for $US14.95. CAUTION! I understand that all games will require DSSD disk drives and two of them will require a Supercart. The titles that are being released are Lurking Horror, Hollywood Hijinx, Station Fall, Suspect & Plundered Hearts. Moonmist & Sea Stalker will require the Supercart. Look for the announcement of these new games in a future Asgard catalogue."

Gary Cox, writing in the January issue of TIdbits, the newsletter of the Memphis User Group gives the following details on another release from Asgard...

"Another addition from Asgard Software is Picasso Enlarger & Using Picasso. Picasso Enlarger allows the user to enlarge, reduce & ghost a picture as well as merge TI-Artist instances into your Picasso pictures & exchange fonts to and from Picasso & TI-Artist formats. Using Picasso is a 40 page book giving you a complete overview of Picasso and how to best utilize its features. Picasso Enlarger sells for $US14.95 and Using Picasso sells for $US5.95. Add extra for postage."

It is obvious from reading about the software that has been and is about to be released through Asgard Software that at least one company is supporting TI users. When you browse through MICROpendium and see the advertisements from Asgard (and others) you realize that the TI still has some life left in it! For example, the November issue of MICROpendium carries this ad for further software from Asgard...

KARATE CHALLENGE - Dragons have taken over the minds of the students & teachers at your Karate Dojo. In this fast paced action game you must defeat your former friends in order to get a chance to engage the dragons in mortal combat... The astounding graphics are both impressive & accurately depict techniques used in Karate...

MISSION DESTRUCT - Robots have taken over the moon! You are on a mission to destroy as many moon reactors as you can before the waves of Death Drones. Space Mines & the evil Draks can overtake your ship... A fast action game by Glenn Schworak, Mission Destruct! is a good old fashioned shoot 'em up that will keep you playing for hours on end!

Both games require 32K, Disk Drive & XB, E/A or TI-Writer. They cost $US9.95 each or both games for $US17.95. Add $US4.00 for Airmail delivery.

FUNCTION-QUIT

From the January issue of LA TopIcs comes this tip from Chick De Marti...

If FCTN QUIT is accidentally hit before you can save a typed program (and you are using XB & 32K Memory expansion) try typing:

```
CALL PEEK(-31952,A,B,C,D) :: PRINT A,B,C,D
```

The first two values shown point to the start of the line number table. The second pair of values point to the end of this table. Write down these numbers. Press FCTN QUIT and re-enter XB and type:

```
CALL INIT
CALL LOAD(-31952,W,X,Y,Z)
```

Replacing W,X,Y,Z with the values you have for A,B,C and D. Then type:

```
LIST
```

The program should then list.

NOTE:- Do the CALL INIT and PEEK commands immediately after fatally pressing FCTN QUIT...before entering anything new on the disk.

REDUCE WEAR & TEAR ON DRIVES

This tip appeared in the newsletter of the Ozark 99er User Group of Springfield Missouri, and appeared in the January issue of MICROpendium...

"Since a disk drive can wear out eventually, and since most of the wear and tear that can happen to a drive is in the area of the carriage and stepping, you can take a few steps to set up your diskettes to do a minimum amount of work every time you use them. After a diskette has been filled up and you no longer expect to make any changes, and are about to put a write-protect tab on it, why not make a copy of it using the file copy utility in your disk manager? This will serve two ends. First, you will benefit every time you catalog the diskette because it will run faster, and second, it will decrease the amount of head wear."

"To see this for yourself, run a directory of a diskette that of have many files on, and just listen to the drive work. Then, copy the files to a newly initialized diskette and see how much faster it creates the directory."

ADDING COMMENTS TO LISTINGS

From the Hints Tips & Answers column in the January issue of the Ottawa Newsletter comes this item...

Have you ever wanted to put comments on a catalog listing, especially when you are passing the disk to a friend? One way to do this is to write the comments in with a pencil, but a neater way is to use TI-Writer. Hers's how:

1) When you go into DM1000 to catalog the disk, first change the output device. Press FCTN 3 when the main menu screen appears and then change PIO default to DSKn.filename. You don't want the printer codes and you don't want to save this change permanently, so press N for both these questions.

2) Go through the normal steps to catalog your disk. (ie main menu option 2; disk menu option 1; once the catalog is displayed, then FCTN 7 to get a printing). However, instead of printing to the printer it will print to the disk, giving you a DV80 file under the name you gave it in 1.

3) You can now access this file with TI-Writer to add your comments and for printing out. It is suggested that you turn off Word Wrap. If comments take more than one line then you will have to insert lines - FCTN 8.

By using this method you can make the listing more informative to the person getting it. For example, programs that are related can be grouped regardless as to their names.

THE SHAPE OF THINGS TO COME?

I received a letter from the Ottawa User Group along with their latest newsletter which may give a pointer to what the future may hold for many groups. Parts of the letter are reprinted here...

"After much discussion, we came to the realization that our Group's numbers have so declined that it has become impossible to cover the costs of the newsletters at the volume we are mailing out at present."

"It is therefore with regret that we must announce that it has been decided to end our Newsletter exchange program. We sincerely hope you understand this decision and realize how hard it was for us to make. Unfortunately, this newsletter will be the last sent to your club under the program."...

The letter went on to say that they are going to encourage their members to become members of various groups so that they can still receive some newsletters and request that groups or individual members become members of their group so that we can still get their newsletter. As groups shrink, this method of newsletter exchange will probably become more popular.

Another group we exchange with, the MSP99 User Group in St Paul in the US have also written to our group...

"For cost and staff reasons, we are reducing our publication schedule to six issues per year, and eliminating the manually-inserted graphics we have used in the past. However, we intend to maintain the fine level of original information for which our newsletter is well known around the world, and we intend to adhere to the schedule of one issue every two months. We would be honoured if you will continue to consider us as exchange partners in the future..."

CRAZY LANGUAGE ENGLISH!

From the Lehigh 99'er newsletter comes this observation on the English language...

We begin with BOX - the plural is BOXES
then the plural of OX is OXEN, not OXES.

Then one is a GOOSE and two are called GEESE,
Yet the plural of MOOSE is never MEESE.

You may find a lone MOUSE or a nest of MICE,
Yet the plural of HOUSE is HOUSES not HICE.

If I spoke of my FOOT and showed you my FEET,
And gave you a BOOT would a pair be BEET?

Then one may be THAT and two may be THOSE,
Yet HAT in plural would never be HOSE!

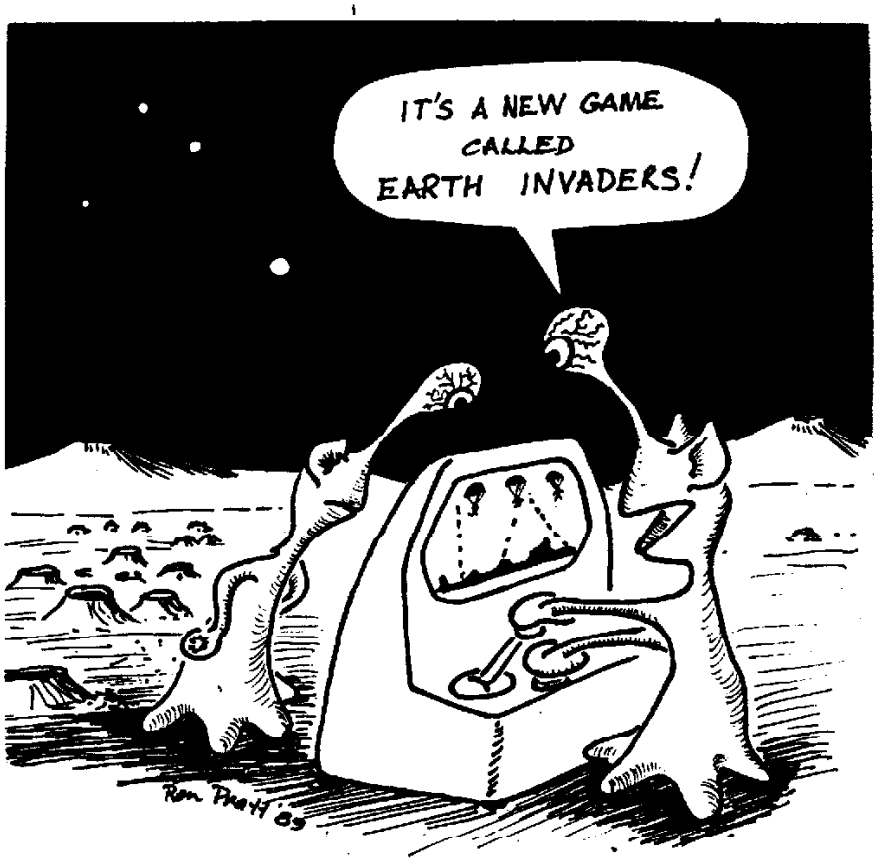
We speak of BROTHER and also of BRETHREN
But though we say MOTHER we never say MOTHEREN!

The masculine pronouns are HE, HIS & HIM,
Imagine the feminine, SHE, SHIS & SHIM!

So English I fancy that you will agree,
Is the one craziest language that you ever did see!

IN CONCLUSION

Well, that's it for another month. Don't forget that it is up to YOU to help the Group move into the 90's by attending meetings, offering suggestions and writing the occassional article for the newsletter. Remember, the future of the TI in the Hunter Valley depends on ALL members contributing something to the Group.



Welcome back.

W-AGE/99 * NEW-AGE/
99 * NEW-AGE/99 * N
EW-AGE/99 * NEW-AGE
/99 * NEW-AGE/99 *

* JACK SUGHRUE, Box 459, East Douglas, MA 01516 *

Dear Editor,

First, thank you for continuing the newsletter exchange during the long time I was mending.

Second, in case you were unaware of my accident but thought I was late and lazy with the latest IMPACT/99 articles, let me explain what happened. I was on my way home from school on a bright, sunshiny afternoon in May, when a tree leaped in front of my brand new car. Car was totalled, but I wasn't. My head was badly smashed in. There was a distinct possibility that I would lose my left eye and/or suffer severe brain damage if I lived. I lived. The left eye is not perfect, but it is see-able and will remain with me. Regarding the brain damage: my wife said, "If Jack has brain damage, who would know?"

The doctors, who were fantastic, put nine metal plates in my head to hold it together and performed lots of other miracles. Anyway, I appear to be getting back to what is normal for me. It's just taking a very long time, during which I went through exceedingly long (for an impatient patient like myself) times when I could not read or compute! I sure had a chance to reprioritize my life, however, and dropped lots of things that were not important. But the things most important to me (family, teaching, reading, writing, and, of course, computing) are where my time and energy and money will go.

So, I'm back. I missed my TI-ing and all my TI friends. I'm back to work and computing on a bit of a limited basis. My energy level hasn't worked its way up to the old levels, yet, but the enthusiasm for my 4A, however, is even higher than before, if that is possible.

Third, if you're interested in receiving a FREE (no gimmicks) monthly two-page article (with a graphical third page now and then) just keep me on your newsletter mailing list. I'll send them in batches similar to these. You don't have to publish them, if you don't care to, but I'll continue the exchange as long as your group would like to receive them. I was hoping to start #1 with your January issue but couldn't make it. I would appreciate your not publishing these articles BEFORE these 1990 dates: Feb - #1; Mar - #2; and all the way up to Dec - #11 for 1990. Publishing them after these dates isn't a problem. Thank you for your courtesy in this matter.

And, fourth, the TI Community world wide was so incredibly supportive and caring that my recovery was undoubtedly speeded up because of this show of sensitivity. There is no way I could ever begin to thank all the hundreds of individuals and groups who wrote and called and sent all kinds of TI thingies. But I most enjoyed the steady flow of newsletters, read to me by my wife and kids until I could read them myself. I felt connected and a part of something definitely worthwhile.

Newsletter editors are the soul of this community, without which we couldn't exist as an entity. I hope you are able to continue this monumental (and not always rewarding) effort. You are very much needed by all of us.

Again, thank you for your patience. Hope you find NEWAGE interesting and useful.

As always, Jack

~~~~~~~~~~~~~~~~~~~

W-AGE/99 * NEW-AGE/
99 * NEW-AGE/99 * N
EW-AGE/99 * NEW-AGE
/99 * NEW-AGE/99 *

~~~~~~~~~~~~~~~~~~~

* by JACK SUGHRUE, Box 459, East Douglas, MA 01516 *

#1

Why NEW-AGE/99? Well, it's been almost a year since my long-running IMPACT/99 columns were rudely interrupted by a car accident which kept me from my beloved ❤ little computing machine.

So why NEW-AGE/99? After my brush with death last spring I initially felt I had been given another shot at Life (definitely with a capital "L") and wanted to reflect this New Age even with my computer. Secondly, "impact" sounded too much like a crash. Thirdly, I think my view from the TI sidelines for so long gave me the distinct perspective of seeing a New Age arrive for us. And, finally, the sharing and caring that was shown to me during my repair time by 99ers worldwide made me realize that my commitment wasn't to the little computer (that kept me occupied in thousands of wee hours) but to the people.

Even now, so long after The Event (always capitalized in my family), I continue to get cards and letters and disks and TI-related thingies from all over the world. This past week I got cards from England, Belgium, and Australia. And a few more from this country and Canada. To all those well-wishers everywhere who were so supportive during my long recovery, I can only say, "Thank you." You were all very instrumental in my rapid comeback. Even the doctors were amazed.

Regarding my operations, the most oft-asked question was, "Will all those metal plates in your head set off alarms in airports?" I have no idea. I haven't flown since. I had the stainless steel one and one of the titanium ones removed in a recent operation, but I still have seven left. Permanently.

And I am back to work on a slightly limited basis. And I DO drive. A new car. My other two-month-old new car was totally demolished when I hit the tree. And - YES! - I was VERY scared the first time I got back behind the wheel once I could see okay again. It had been so long. I'm still cautious, but I'm no longer frightened.

I had also been drifting toward large blimphood, but I lost lots of weight after the accident and have been able to keep it off. Crash diet. So there's another positive.

I really have a thousand TI people to thank, but I must particularly single out Charlie Good, John Willforth, Chris Bobbitt, Timothy Dermody, Tony McGovern, Jim Peterson, Jim Cox, Sister Pat Taylor, and so many patient newsletter editors for support above and beyond the call of duty. Having inadequately given such little thanks when so much more was due, I'll now move right into N-A #1.

Each month I plan to explore in these two pages some TI fairware and some commercialware and a bit of the goings-on.

The first going-on is Bill Gaskell's. I am sorry to learn of his FOUR-A/TALK swan song. For the year he wrote it, it was the most interesting column around. The variety and the history and the enthusiasm and the good writing made FOUR-A/TALK the boost we all needed. It was a very sustaining column that each month also introduced us to all the new goodies available to us (and there are many). The TI World Community will sorely miss your writings, Bill. Hurry up with the

skiing and golfing and get back to your computer.

In Bill's last article he talked about John Johnson's most recent BOOT program as of October 1989, good for Hard Drives, etc. For those with the "normal" system of a drive or two but not the techie wizardry, you can still benefit from the BOOT. It may not be the ONLY way to go, but it sure is one of the very best. Someone sent it to me during the time I wasn't able to compute, but when I got back to the keyboard recently I popped it in and had my flabber gasted. First it says, "MUG BOOT LOADING." That's the Miami Users Group (6755 Tamiami Canal Road, Miami, FL 33126).

Then comes the menu: 1 SHOW DIRECTORY; 2 DISPLAY A FILE; 3 RUN A PROGRAM; 4-9 Options 1-6; C TI XB.

#1 shows a directory of any disk (including RAMs) and permits marking of files for viewing, running, deleting. Marking auto-sets files when you return to Main Menu. A text file appears when #2 (DISPLAY) is pressed. An XB or E/A file runs automatically if #3 (RUN) is pressed.

When you press the spacebar another menu (with Options 7-15 and TI XB) is displayed. Press again and a third menu (with 16-24 and TI XB) is shown. This means you can type in anything on the options to VIEW or RUN including LOAD itself, which is handy if you have a bunch of LOAD programs (such as FUNNELWEB) that you want to operate off ANY drive. If you have a DSDD and two drives, for example, you could load onto this MUG BOOT disk all your favorite programs AND another whole disk of favorites on Drive 2. (And a zillion more.) Then you can put your top 24 on this menu and your other top 20 onto your FUNNELWEB menus and on and on.

With this program you could build the perfect environment for yourself, but that's only scratching the surface of this February '89 version I have.

Here are just a few of the single keypress things this will do for your minimum XB disk system: turn screen off; load disk directory; print directory; view text file; print text file; run XB or EA program; EASILY configure up to 24 (actually, unending number) files for autoload; run gram/grom modules; delete files; cycle through and mark groms; toggle XB color interrupt routines off and on; do a CALL routine; change print device; get and display ROM header at >6000 (whatever that means); configure BOOT tracking; display version # and author; toggle instantly between all options; save all configurations; use additional active keypresses in sub menus; and so on.

Could you have dreamed that something so wonderful and so simple to operate could ever exist for our TI? Get the latest version from your user-group library or write to Miami. Don't forget a decent fairware donation.

In his final column Bill Gaskell was surprised by the ingenious FUNNELWEB built-in to change upper to lower case by holding CONTROL/period and running the cursor over the characters (words, sentences, etc.) to be speedily converted. I don't think he realized that Tony McGovern also did a reverse - lower to upper - by holding CONTROL/semi-colon. Tony told me he added this because he was not a good touch typist and found retyping from one case to the other too time consuming. The day after I got this version (4.0 and up), I was typing along without looking at the screen. When I looked up I had 20 lines or so of upper case. Zip. With the cursor and Tony's ingenious keypress changer I was back to typing in seconds. Try it the next time you use FWB (speaking of which, have you tried the new 4.2? It's got some great LARGE changes.) NEW-AGE/99 will detail it all soon.

(If you use NEW-AGE/99 please put me on your exchange list.)

NOT ANOTHER SORTING ARTICLE

well - sort of

Perhaps, like me, you've come across sorting programs of various kinds, looked at the way they'd been cleverly worked out (algorithms), and then moved right along to more relevant material.

Recently I found that I had need for a sort (of the computer program kind). It came about when, after a rather lengthy period of unsuccess as a part-time punter, I had a liquidity problem (1) and decided to try to recoup some (or all) of the outlays as a practising mendicant.

Each day, after a few hours of slumping behind the battered begging Akubra, I'd tot up the takings - less the float and operating expenses (2) - and enter the amount in my little black book. This record of earnings, of course, is for passing along to my tax agent. (3)

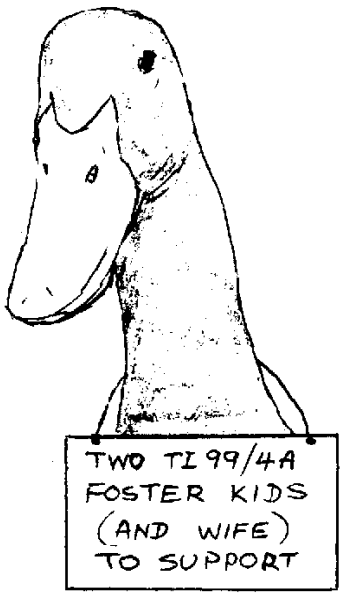
After a couple of months of mendicating I thought 'why not run the daily gleanings through a sort?' This would give me a record of how many

good days ($100+)
extra good days ($200+) and
you-rippers (4)

there'd been.

Knowing naught about sorts, I went to my filing system and looked under L - mother duck told me when I was but a mere tad 'you'll find all sorts under Licorice'. I unearthed one that seemed to fit the bill. It was not a 4A program so I TI Basiced it, checked it out and decided to tizzy it up a bit to accept and sort dollars and/or cents and then neatly display the end result.

Anyway that was the intention - it didn't quite work out as I had hoped. Well, win some lose some I guess. So here is a sorting program that still needs some sorting out. The bare bones program consists of a mere 20 lines. At line 20 the DIMensions are purely nominal and should be set at>=the value of A at line 90. The inability of TI Basic to allow for an inputted variable to dimension an array is a pain in the tail feathers.




```
10 CALL CLEAR
20 DIM N(50),N$(50)
30 PRINT "IF INSTRUCTIONS AR
E REQUIRED";:"KEY Y FOR YES
- N FOR NO"
40 CALL KEY(3,R,S)
50 IF ABS(R*2-167)<>11 THEN
40
60 ON INT(R/20-2) GOTO 80,70
70 GOSUB 350
80 CALL CLEAR
90 INPUT "HOW MANY ENTRIES?
":A
100 CALL CLEAR
110 FOR L=Ø TO A-1
120 INPUT "ENTER AMOUNT ":B
130 T=T+B
140 FOR C=1 TO L
150 IF B =N(C) THEN 170
160 NEXT C
170 FOR K=L+1 TO C STEP -1
180 N(K)=N(K-1)
190 NEXT K
200 N(C)=B
210 NEXT L
220 CALL CLEAR
230 FOR K=1 TO L
240 PRINT N(K)
300 NEXT K
310 INPUT "TAP ANY KEY FOR T
OTAL":A$
320 CALL CLEAR
330 PRINT "TOTAL IS"::"$";T
340 STOP
350 CALL CLEAR
360 M$="ENTER AMOUNTS"
370 GOSUB 460
380 M$="IN DOLLARS $$ OR"
390 GOSUB 460
```

```
400 M$="DOLLARS & CENTS $$.c
c"
410 GOSUB 460
420 M$="KEY ENTER AFTER EACH
 ENTRY"
430 GOSUB 460
440 CALL SOUND(3000,40000,30
)
450 GOTO 80
460 Y=Y+4
470 FOR M=1 TO LEN(M$)
480 CALL HCHAR(Y,4+M,ASC(SEG
$(M$,M,1)))
490 NEXT M
500 RETURN
```

The N$ array was DIMensioned for use in the program, but things didn't quite turn out as expected (story later).

Lines 30 to 80 and the subbies at 350 and 460 are purely cosmetic and were included mainly to demo. my version of YES/NO via CALL KEY with ABS and INT(5).

The reason for including the N$ array was to tidy up the dollars and dollars and cents entries. If you INPUT 6.98 and 55 (as 55 or 55.00) it OUTPUTs as

```
 6.98
 55
```

and the aim is for

```
 6.98
55.00
```

by using a Stephen Shaw clever-bit. To explain, let's slip in another line

```
185 N$(K)=SEG$(STR$(N(K)+.001),1,
    LEN(STR$(N(K)+.001))-1)
```

and amend line 240 to

```
240 PRINT N(K),N$(K)
```

I reasoned that, as the amounts were allocated their (K) spot in the N array at line 180, converting the amounts to strings by the immediately following line 185 should give these N$ strings an equivalent (K) spot. WRONG!!!!

Just for a trial run I INPUTted 2.2, 3.3, 4.4, 5.5 and 6.6. Output was:-

in that sequence (2.2 to 6.6)		in reverse sequence (6.6 to 2.2)	
N	N$	N	N$
2.2	.00	2.2	.00
3.3	2.20	3.3	3.30
4.4	3.30	4.4	4.40
5.5	4.40	5.5	5.50
6.6	5.50	6.6	6.60

I tried to reason out the problem but it made my head hurt. Perhaps I should have taken Max Walker's advice and stayed with only 2.2(6).

Of course the clever bit can still be used to produce a neat and tidy output by replacing line 240 with

```
240 N$(K)=SEG$(STR$(N(K)+.00
1),1,LEN(STR$(N(K)+.001))-1)
250 P=8-LEN(N$(K))
260 FOR J=1 TO P
270 N$(K)=" "&N$(K)
280 NEXT J
290 PRINT N$(K)
```

which works fine and gives the desired

```
 6.98
55.00
```

type result, but for more than 20 odd items some means of halting the screen scrolling is required.

As my requirement was to process around 30 or so entries at a time, I ended up by 3-columning the output by adding line 225 and changing lines 230 and 240

```
225 Q=INT((A+2)/3)
230 FOR K=1 TO Q
240 PRINT TAB(4);N(K);TAB(12
);N(K+Q);TAB(20);N(K+Q*2)
```

which gives an untidy (but sufficient unto my needs) 3 column output.

As I've mentioned, my original intention was to knock up a sorting program to provide for inputting monetary amounts in any format and then produce a 3 column output on the screen with all the .s neatly lined up.

Well - it didn't quite happen that way. However, if someone knows how it should be done, an amended program (via the ed. for publication) would be appreciated. As a matter of fact, the chance of finding out how it should be done (if anyone answers the plea) was my only reason to continue with the article, as the subject matter is of no particular significance. (7)

Then again, perhaps I should have perservered and checked a few alternatives until I jagged it myself. Unfortunately I'm still somewhat despondent after the disappointments of Christmas. It seems that Santy didn't get my letter (AGAIN!) 'cos I didn't get a RAMdisk anna modem anna printer anna Geneve and not even a little rubber manny for my pond. Well, whatever - happy computering for the last year of the decade and the rest of the 90s.

(1) I refer, of course, to financial liquidity as I am fully aware of the need

(2) to take precautions against dehydration.

(3) no doubt, like me, you are always grateful for the opportunities which are provided for us to make contributions to the public moneys pool so that our elected representatives can bestow largesse upon various unworthy causes!

(4) to be honest I haven't had a you-ripper yet. Still - ever the optimist, that's me. For instance, I even hold out hope of being able to produce truly really original computer programs one of these days.

(5) ABSing and INTing the Return variable of CALL KEY only requires the appropriate maths. eg the idea can be used for DESX keys like so

```
200 CALL KEY(3,R,S)
210 IF (ABS(R*10-685)<>5)
+(ABS(R*2-171)<>5) THEN 200
220 ON INT((R+21)/9-8) GOTO etc
```

(6) for OS readers, and perhaps some locals also for that matter, the reference to Max Walker (former test cricketer and now all-round sports enthusiast) and 2.2 (%ALC /VOL) is merely a touch of current Australiana.

(7) remember - tidying up untidy programs counts as a Keep Australia Beautiful activity.

POSTSCRIPT

As I was reading through the first part of this article, it occurred to me that the (little black) duck references may seem obscure (putting it mildly) to some people. So - an explanation is in order. As a graduate of the Queensland Under-Achieving Computerers Kindergarten it's natural to use the LBD appellation.

It's fairly obvious that those who attended this establishment would come to be referred to as ducks - although, more correctly, most were drakes (I'm an old drake myself). However, the general public is used to duck as the common generic term. For instance, nobody refers to drakelings - not that it matters at that age (or does it?).

And so it was at QUACK that, having been lumbered with this designation, we decided to adopt Daffy as a role model. And , in no time at all, we were all little black ducks. Of course Daffy is a drake, but to make it big in cartoons, allowances must be made for the awareness-level of the viewing public.

The Kindy, set up in the heyday of the TI Home Computers, only lasted a couple of years. In the main it was for those who couldn't come to grips with Basic, although there were a couple who couldn't get the hang of their joysticks. And once, after the LBD label was established, there was one chap who wanted to enrol because he really thought he was a duck. But that was no problem, he was just put in charge of the swimming pool.

Not only were we learnt proper how to program in Basic but we were shown how to take care with our foundling 4As. Before I went there this is how I was.

"HIT ANY KEY TO CONTINUE"

Unfortunately I did not receive a graduation certificate - probably because I graduated three weeks before the rest of the class. The headmaster called me up and informed me "you passed. You are not required to attend this kindergarten again." I appreciated this simple ceremony as I am naturally shy and retiring - although I would have liked to attend the dux of the kindy presentation. Actually the head was a bit confused that day because he really said "you passed again" and it was only my first attendance at the kindy.

At one time I had hopes of continuing on and enrolling at QUAXB School. Of course they copped the duck designation also - however they adopted Donald as their role model (we got Daffy first!) and called themselves the Dons.

Unfortunately they would not accept me at QUAXBS as I did not have a graduation certificate from QUACK. And so, I never had the chance to continue on from there and attend QUAIAL U. Yep - you guessed it - they were the quails. Of course, when production of the 4A ceased and its popularity waned, all three of these establishments wound down. Poor old QUAIAL U never did get around to adopting a role model - just think, if they'd lasted a few more years, they could have been the Dans! And that's this little black duck's explanation and you're stuck with it.

AXIOMATIC APHORISMS

From The Foster Parent

Coruscating matter in toto will not necessarily produce an auriferous assay as a logical consequence.

Mendicants are incapable of possessing the ability to become selective recipients.

Pulchritude possesses singularly cutaneous profundity.

A geographical position confirmed by the actuality of vapour composed of particles originating from the combustion of flammable substances verifies conflagration.

No testimony is forthcoming from male cadavers.

The male human being who cachinnates ultimately cachinnates optimally.

Avian species possessing conforming epidermal growths congregate collectively. However, a couple located somewhere within a tract of land covered predominantly with fruticulose perennial plants, would each have an estimated worth of only 50% of that pertaining to one located in the manual extremity of the arm below the wrist.

Manual propelling of petreous projectiles by inhabitants of vitreous edifices is ill-advised.

There is a myriad of a woman's undergarment worn immediately under the dress in the region between the small round drinking vessel usually manufactured from porcelain or similar earthenware and one of the two fleshy outer edges of the mouth.

You are forbidden to utilise the precipitation of lacteal secretion as justification for ululation.

The consequence of ethical morality is reflex gratification.

A sharp sudden pain in the side occurring in the conception of past, present and future as a sequence, effects the spiritual salvation of one more than eight.

Freedom from defilement is contiguous to a righteous condition.

FAR OUT (In the bush)
By Dick Schaydel

As you might imagine, night-life here in the bush is not up to "Gold Coast" standards. In fact, after dark there isn't much to do. There's a bloke who comes up once every 6 months for a load of pigs for market (in exchange for "Redbacks", of course) but that is about the extent of commerce in the area. The good part is that you don't have nosy neighbors poking about or inane regulations to deal with. Really, this newsletter article idea has given me something to do when darkness blankets this place. The most excitement lately was when a Dingo wandered in and got his head stuck in the farrowing stall --- at 3 in the morning!! That reminds of another yarn --- get Ron K to tell you about his encounter with the Emu!

Anyway, I make the obligatory run into Lightning Ridge about every fortnight to buy supplies (fuel and tucker, mainly). That dispensed with quickly the cold store of the L/R bowling club has to be checked out to make sure that all is in order and maybe to suck the sides in on a tinny or two -- strewth! to the task at hand!

The first article was a taste of things to come. Agreed, it was a bit superficial but we'll correct that! A programs utility should not be judged on how good it looks or how complete the documentation is but rather on what you would do if it were suddenly lost. I have been using the SPELLCHECK program from Dragonslayer Software for several years. Quite simply, it is the ONLY spellchecking offering for the TI independent of a word processor. Besides the 20,000 word dictionary that comes with it, you can optionally add your own specialised vocabulary lists. A doctor, for example, could create a dictionary of medical terminology with the SEEDGEN program and add to it as time went by. In fact, every time you check a document, you have the option of adding as many unrecognised words to a customised dictionary as you wish. There are simple step-by-step on-screen instructions and the screen prompts are easy to follow (ie. 'N' for Next, 'A' for Add, etc.). Reading the documentation is very helpful but the program can be mastered with a 'trial and error' approach.

Another interesting attribute of the program is that the specialised dictionary files are saved in a D/V 80 file format. That means two things: 1) the dictionaries themselves can be compared and duplicate entries eliminated, and 2) the entries can be deleted or new ones inserted directly with a TI-Writer type of word processor. Although the supplementary dictionaries are limited to 2,000 entries per dictionary, there is no limit (except for disk space) on the number of individual dictionaries used to check a document.

The only drawback to the program is one that I consider minor because it can be easily overcome. The main dictionary was written in American English. Our spelling of some words ---ie. "tyre" instead of tire, "colour" instead of color will show up as mis-spellings. You can do one of two things (or both) press 'N' for Next and ignore the word and/or use 'A' for Add to put it in a specialised dictionary. SPELLCHECK is a very versatile program.

Just as SPELLCHECK considers our native tongue to be a bit eccentric, it has problems with foreign words and phrases. You could create a French dictionary (or several) and my favorite a German dictionary. As I said, it is extensible virtually without limit.

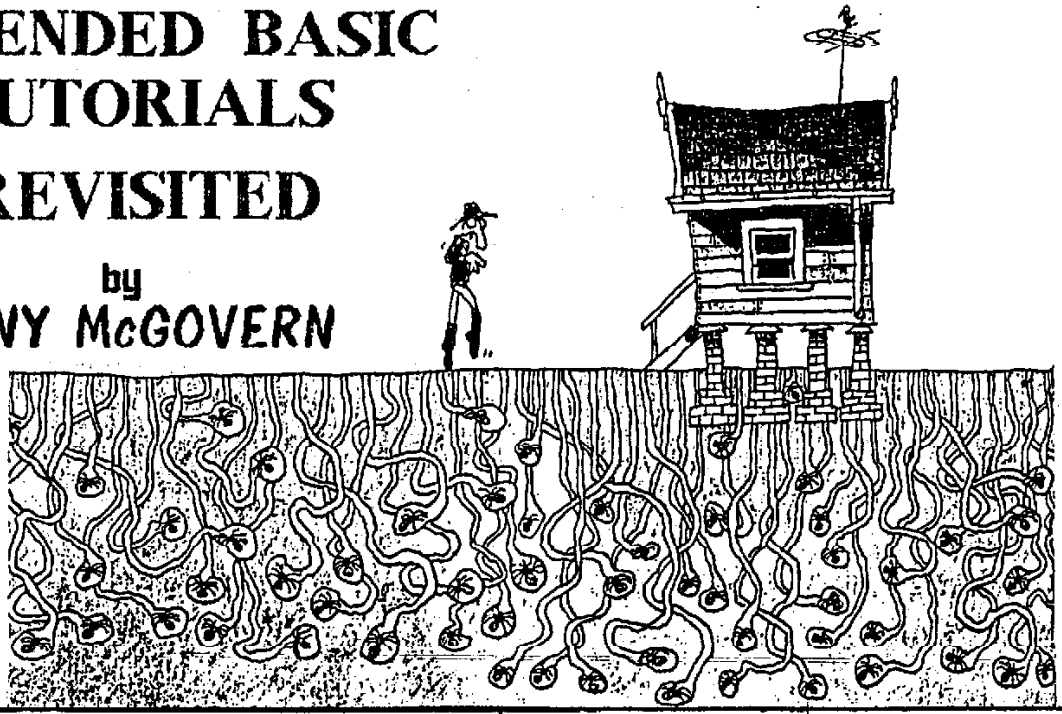
That brings me to the end of this. If I write anymore I'm going to try to limit these to one or two programs for the sake of completeness. Remember, fossick about the mullock heap --- you might find a bit of colour in the form of a program you didn't know you had!

Back to the task at hand -- the 'roos found my veggies last night. It's all out bloody war this time, mate!

EXTENDED BASIC TUTORIALS REVISITED

by

TONY McGOVERN



EXTENDED TUTORIAL #7

We now continue the Tutorials with detailed ways and means of scrunching program length. As I remarked before, it's a subject I'm not completely comfortable talking about because, while this series has been devoted to better XBasic programming, most things you can do to scrunch Basic programs make them less readable by ordinary mortals, given reasonable programming skill in the first place. The other reason for my reluctance is that this kind of discussion tends to degenerate into a collection of unrelated items, yet another set of "Tips", when I really want this series to be a gentle but systematic look at the workings of the machine and its language(s).

Anyway let's start at the small end of things and work up to the larger scale. Last time we looked at the space taken by simple variables. The most obvious thing is to keep variable names short. I don't recommend this until late in the piece because it is such a cheap and obvious way of gaining bytes that you might as well have the help of descriptive variable names until you are absolutely desperate for bytes. Absolute desperation has not occurred until you have had several rounds of byte saving already. The shortest variable name has only one letter character, but TI Basics also officially allow "" (shift-2) and "_" (fctn-U) as variable names. It has to be a fairly long SUBprogram before you need more than 26 simple numeric variables but it can happen. On this console there are 3 other single characters which can be used as variable names. Experiment to find if they exist on your machine. The nagging problem is that they are not documented.

There is another way to use variable names to shorten a program. Remember from last time that a one digit numeric constant is treated as a string and takes 3 bytes, while a single letter variable takes only 1 byte. If a particular numeric value occurs frequently in a SUBprogram, 0 or 1 being common examples, then it may be worth the overhead, 14 bytes plus the defining statement, for a new variable of that value if you can then save 2 bytes on numerous occasions. A frequently used longer numeric constant, as might occur in CHAR or SPRITE manipulations, yields more bytes each time. It is a matter of doing careful book-keeping and byte counting in each SUBprogram. Once you start down this track be alert for further gains -- if you have defined S=7 and F=5 then it saves a byte to write S*F instead of 35. If you can reuse an already defined variable name then the investment is paid back faster, but this requires keeping very careful track of program flow. Go back to the example of a

Key/Joystick routine in an earlier Tutorial and see if you can shorten it by reducing the number of variables used.

Replacement of numbers by variables has precedents in other languages. In TI-Forth the numbers 0,1,2,3 are not treated directly as numbers but are defined words in the language.

There is another little way that cunning entry of characters can shorten programs. This is in the entry of graphics characters with ascII values above 127 in the upper color groups of XB by writing strings with DISPLAY AT instead of H VCHAR CALLs. Characters in this range can be entered in strings in program statements by use of the CRTL key, rather than by using the CHR$ function. It does tend to make the program incomprehensible as these echo as blanks to the screen. They will appear with their defined shapes if the line is called up for editing after RUNning the program. These codes are also used as XB tokens and can only be used within strings. I should add in passing that I am in total agreement with the TI designers' choice not to allow abbreviated (direct token) entry of Basic keywords. If you want that sort of thing you should be back on your Sinclair or Commodore, and you probably don't believe in relocatable object files either,

The use of arrays to represent small collections of numbers needs detailed working out. The gains from less variable table overhead and simplified parameter passing to SUBprograms have to be balanced against the extra bytes needed for each program reference. Let the program logic be your initial guide.

This idea of using fewer bytes to represent quantities leads on to the larger subject of data compaction. One byte can carry 256 different values, and one third to one half of those can be conveniently entered from the keyboard. It's sheer overkill to use an 8 byte floating point number to represent just a few values, or even just a logical (Boolean) variable which really needs only one bit. Some languages compact Boolean variables as bits in a word or words. The CRU single bit bus of the TMS-9900 provides an ideal mechanism for bit storage and testing, but as in so many other areas the 99/4a hardware does not do justice to its CPU. The later TMS-9995 in fact has a little on-board CRU memory for just this purpose.

Opportunities for data compaction are limited in XB both because of the structure of the language (it has only character strings, floating point numerics and arrays of these as data types) and the convoluted, slow way it is implemented via GROMs and VDP memory. Any scheme for coding or compacting needs computation to pack and unpack the data. At the machine code level the tradeoffs between memory use and speed are different from those in Basic, especially TI-99 Basics, because Basic is so much slower. In my experience the use of string variables to compact data in active parts of a program is almost always doomed to failure because of slow string handling by XB and pauses for garbage collection. Data compaction can be useful though in setting up initial graphics designs or for music data. There are only so many different notes, in pitch length and volume used in any given short musical piece, and since each note takes time to play and is handled by the machine on an interrupt driven basis, this time can be used to do the computations needed to unravel the data for the next note.

Let's have a look at the graphics screen example. Suppose that in setting up a game screen, either one of two characters, maybe the same pattern in two different color groups, has to be written to 20 locations in various parts of the screen. The simplest way is a whole succession of CALL HCHARs - assuming the display is not suited to generation with DISPLAY ATs - and that's the way you will find it done in many programs (just like long lists of CALL SOUNDs). What is totally unforgivable is to find incompetent magazine or commercial programs with inefficient coding that force inconveniences like CALL FILES(1) or (2) on the user when it could have been avoided.

1000 CALL HCHAR(23,12,105) 1010 CALL HCHAR etc etc

This takes over 600 bytes. How can it be shortened ? One way, a bit of a dead end in this example, is to

use multi-statement lines. This would be shorter by 30 bytes or so, and marginally faster. The real improvement is to eliminate the repetition of CALL HCHAR - remember CALL is cheap but HCHAR is expensive - by using a loop and DATA statements.

```
1000 FOR I=1 TO 20 :: READ A
,B,C :: CALL HCHAR(A,B,C)::
NEXT I
1010 DATA 23,12,105, etc etc
```

Now all but one of those HCHARs have gone. The price paid is loop and DATA execution overhead and the increased possibilities for clerical errors since the DATA items have been divorced from their proper context. At this stage you may be feeling very pleased with yourself, but then you find that to add another feature to your program you need more space. Now is the time to reflect seriously on data compression. A column index for HCHAR can only have the values 1 to 32 and rows 1 to 24. One of these values can be expressed by 1 byte with possibilities to burn. Say you use 1 byte for each row or column value then. Expressing the bytes efficiently as DATA is the next problem - there are a few bytes of overhead for each item in a DATA list, and DATA lists of a lot of short items are notorious for causing a "line too long" error. So let's pack them in a single string and use SEG$ to unpack them, with ASC to turn a ASCII character back to a value for HCHAR. A minor problem is that characters 1 to 32 can't be entered directly in XB, so just use characters starting with "A" and subtract 64. The opposite problem may occur with the string for the character values if upper graphics sets are being used. Then just use lower values and add a correction. So now the code might look like

```
1000 READ A$,B$,C$ :: FOR I=
1 TO 20 :: CALL HCHAR(ASC(SE
G$(A$,I,1))-64,ASC(SEG$(B$,I
1))-64,ASC(SEG$(C$,I,1)+32)
: NEXT I
1010 DATA "W...  ","L...  ","I.
```

You could further pack the data into a single string and modify the SEG$ statements accordingly, but it might not be worth it. Remember now that the problem posed involved writing only two different characters and work out how you could compact things still further for this limited case. This example is based on one of methods that was used to squeeze TXB into console memory. An extreme example of data compression comes when the data is regular enough that it can be generated by a formula or procedure. This is something that has to be worked out in each case.

The use of loops as in the examples above applies in other situations, particularly in CHAR definitions. XB allows the use of multiple arguments in CHAR, COLOR, SPRITE and suchlike SUBprograms. This is better and faster than using individual SUBprogram CALLs for each item in the list. The real dilemma comes when you try to use a loop to compact the program further. Critical parts of the program may be slowed down unacceptably so that you may find yourself using compact slow code in some parts of a program and longer but faster forms elsewhere. Just in passing I should remind you to null out on exit from a SUBprogram, any string variables not required to keep their value till the next CALL. This particularly applies to string variables used for READ, INPUT, PRINT etc operations involving long strings. Remember that it is the length of a program while RUNning that really counts.

IX More Bugs in XB

Now funnelweb spiders aren't exactly the nicest critters to be found around Funnelweb Farm, but they do have the virtue that if you don't botner them they leave you alone too. Unfortunately the bugs that infest Extended Basic aren't nearly so accomodating in keeping out of the road in the first place, and insist on making their presence felt.

I have looked at a few in previous XB tutorials where they came up naturally in the subject matter, mostly in ACCEPT AT. This time we have two beautiful specimens. The first of these was exposed in the Spring 85 issue of TI*MES from the UK by John Bingham. To see it at work, enter the following little program

```
100 I=1 :: IF I=1 THEN J=1 :
```

```
: GOSUB 200 ELSE K=1 :: J=2
110 PRINT K;J :: STOP
200 RETURN
```

Before you run this, predict what it is going to print out ! Then run it and see what you get. Next reverse the order of the statements between THEN and ELSE and run it again. Now it should work the way you expected, K=0 and
J=1. If your XB gets it right the first time, it's different from the one I have. The presence of the GOSUB just before ELSE seems to have upset XB's mechanism for keeping track of ELSE, and the program has gone ahead and ignored the ELSE and the statement after it and executed the following statement which it should have ignored entirely while proceeding to the next line. Try substituting a dummy SUBprogram CALL for the empty GOSUB. XB then works just as expected. Yet another reason for using SUBprograms instead of GOSUBs.

The XB manual lays down a few prohibitions on what can go into IF doesn't mention this little beauty. It does seem, despite the warnings, that FOR..NEXT loops can follow the final ELSE without problems, but this usage is not to be recommended as it may not hold good for all XB modules.

I must admit that reading this news had me a little worried, as I have written long and thoroughly debugged XB programs with some tricky IF..THEN..ELSE footwork, and had never picked up this problem. How come ? The first saving grace is that the Tutorial advice I gave is for real, and I use very few GOSUBs and very many SUBprograms unless I am absolutely desperate for more bytes. This was frequently the case in the writing of TXB, and the central SUBprogram, one of 12 in the program, itself contains 12 subroutines written in to save bytes. Careful study of the code for TXB showed that none of the GOSUBs was written in a way that would let this evil bug loose. When you look back at something like this, you wonder whether you had scrapped particular pieces of code that never quite worked properly, for entirely wrong reasons.

The second bug mentioned has not yet been fully explored. It showed up in a CALL LINK from XB to an assembler routine which also passed in a string variable to be examined by the routine. It happened in the COLIST program, the all-singing all-dancing version of the SIMPLIST program which appeared in an earlier XB Tutorial, The symptoms were that the machine crashed utterly when it was printing out a line of its own listing, after it had already printed several hundred lines, many quite similar to the one that caused to the trouble. Not just an error caught and reported by XB, but a full blown paralytic seizure. It turned out after some TRACE work to be in the very line that was being processed for printing, and to be asociated with the LINK name being at a particular position in the line of text being passed in with STRREF. The problem seems to be CALL LINK extending its link name search over places it shouldn't orter, but more research is needed. Further reports in Entomology Corner in a future issue.

One disappointing discovery came up in the bug hunt. I disassembled the XB machine code utilities loaded by CALL INIT to see if the problem was in STRREF. The good news is that that code looks OK, but the bad news is that STRREF reads strings out of VDP in the same slow way that the console does, 1 byte at a time, resetting the VDP addresses each time. This is the tortuously slow way GPL does it because the console has hardly any CPU RAM, but it scarcely seems necessary in STRREF which can only be used when there is expansion RAM present.

Another bug in CALL LINK has been reported in the US of A in some older XB modules, I suspect prior to the models of V110 sold in Australia. This comes when the link name is supplied as a string variable, as in CALL LINK("A$"). If a garbage collection is performed in VDP RAM by the XBasic interpreter in between assigning A$ and using it in CALL LINK, this routine would lose track of where A$. I have not encountered this bug myself, but I will try to stir it up in the XB modules I have. This reported bug brings to mind the strange state of affairs in XB where it is possible to DELETE a file by string variable reference but not to RUN a program file by similar reference, leading to a small cottage industry of ways around this deficiency.

X A Few Last Little Items

As you are now no doubt aware, not all Basic programs will run correctly in XG, and some fail altogether, usually because XG supports fewer color groups than console Basic. Suppose you want a program that can run under either Basic or XB, and determine from either without crashing which interpreter is in control. If it can run under either, then you should be able to edit it in either! Some suggestions have been made to use PI which is a reserved word in XB giving the usual math value. This runs afoul of edit processing though. A better way is to look at the first value returned by RND. The initial seeds have quite distinct values. Just don't call RANDOMIZE first.

If you are writing such a program it will also be necessary to take care in using colons in PRINT statements to cause multiple line scrolling. XB will enter two colons in a row in a statement as a statement separator token. It is necessary to use semi-colons in between to stop them being run together when a line is being called up for editing as in

200 PRINT "A":::::"B"

XI In Retrospect

These Tutorials were originally written about a year after TI orphaned the 99/4a. They have even been translated into Swedish since then. Now it is over 5 going on 6 years since Black Friday in late 83 and the old 99/4a is still going strong, and the industrial strength of the expanded system has allowed all sorts of new developments, RAMdisks by the megabyte if you want, a hard disk controller, and best of all now 80 column displays using the Yamaha development of the original TI video processor. The machine's potential is by no means exhausted and it remains a fascinating one to work on, and even the original console/XB remains a viable and now very low cost introduction to computers for new beginners.

THANK YOU AL!!

Al Franks is one of the really active back ground workers within our group. Al is always ready to be of assistance to our members if he possibly can. His major contribution has been the repair of console for club members. Al has at this point repaired all of the consoles that he had been given to repair.

I can assure you that repairing consoles **IS NOT EASY.** Al has a mastery of the soldering iron which makes me envious, he also has great patience, another thing which I envy.

SO! recently when I found myself without a printer, this was going to make putting the Newsletter together a bit difficult, guess who marched in one evening with his printer under his arm. YEP! Al, so we collectively should thank Al for his assistance in getting this month's newsletter together AND for his efforts (which go mostly unnoted) for repairing consoles.

From me Al thank you very much, it is members like you that keep this group alive and kicking.

THANK YOU RON!!

While talking about thanking people I can't let Ron Kleinschaffer be passed by. Ron has done a power of work with the QED and QUEST software. So much so that if you have either or both of these you **SHOULD** be using Ron's software, it is in the library.

More recently Ron has done a lot of work on the QUEST Ram disc which resulted in a fix for a nagging loss of DSR problem. The results of this work is in Ron's december article. Now Ron has been the driving force behind making possible the presentations which will have been made to Bob Carmany by the time you read this article.

SO!! From me and all our fellow members of the Group thank Ron, as with Al you people are the backbone of our group.

JOE WRIGHT

A PROGRAMME PACKAGE
From BOB CARMANY

To co-incide with his visit to Newcastle for the February General meeting, Bob Carmany has released a software package. The package is to be distributed through the Hunter Valley 99'ers.

We cannot say enough how pleased we are to have Bob coming to Newcastle. To make the trip Bob has put himself through some considerable personal inconvenience. This alone tells us something of his affection and committment to the HV 99'ers. The distance he is travelling to get here can only but re-enforce our appreciation of his committment to the Group.
The package will be in the club library as of the February general meeting, please contact Stewart Bradley for your copy. The package consumes 257 sectors of disc space.

Bob has two documentation files on the disc. They are DOCS/1 and DOCS/2, rather than me tell you here what the programmes do, I have reproduced both sets of documentation on the pages following Bob's regular article Random Bytes. They themselves make for interesting reading let alone the use one can make of the programmes contained on the disc.

The Program writer programme contained on the disc will be demonstarted at this montn's (February) meeting by Bob. Certainly hope we get a good attendance that night.

In addition to visiting us here in Newcastle Bob will also travel to Queensland and vist his long time pen pal Larry Reid, another HV 99'er. Bob and Larry have become good friends through their exchange of letters and TI information. It would be fair to say that Larry is looking forward to Bob's visit equally as much as we are here in Newcastle.

How to get here!!!!

I thought it might be useful to have some instructions here for Bob on how to get to Newcastle from Greensboro in North Carolina.

Firsity fly due west (or there abouts) from Greensboro. Fly for about 3/4 hours until the west coast of the USA is encountered, you should cross the coast somewhere between San Diego and Seattle, if that doesn't happen you could be a bit lost. On sighting the coast turn left a bit and continue sort of south west-ish. Assuming that you are on the correct heading you will in all probability discover Hawaii, that is the small island chain whare they have really good surf.

Once again turn left-ish a bit more, you should now be heading a bit more south west-ish. This bit is the long hop (note the Aussie animal term). After a bit of a sleep, some tucker, a bit of a drink and a few trips to the loo you should be getting close to one of the bits that broken away from Gondwanda land millions of years ago. We here think that it is the most important bit.A note of caution is well in order at this point. On sighting land have a really close look out the window, if you see lots and lots of sheep roaming the hills and very very few people you have arrived at the wrong spot. This is Bruce land you have found, not Oz. Immediately turn right and at all cost don't land. If however all you see out the window is what look to be a place of great wonder and interest, land as you have arrived in Oz. You will be made most welcome and not allowed to leave until many wonderous things have taken place.

THE ROOSTER!

random bytes

with

BOB CARMANY

Another New Year is upon us! I hope that the holidays were kind to everyone and that there were a lot of TI "goodies" under the tree.

This month we are going to look at one of my least favorite XB programming areas ---string variables. String variables can be of considerable use when writing a program. Besides the obvious use of equating them to bits of text, they can be used to define graphics characters. In a long character definition, they can save space in the line and also memory space. That is something that appeled to me when I first started writing programs. Like most programmers, I try to take every short-cut that I can when it comes to saving keystrokes. I rationalize it by mainating that it "saves wear and tear" on the keyboard (not to mention reducing the chance for error).

The first example that we are going to examine is the RPT$ or repeat string function. The syntax is easy to learn and equally easy to use in a program. The syntax looks like this:

```
RPT$("character",# of repetitions)
```

So, the application would look like this:

```
RPT$("F",16)
```

and would repeat the letter "F" 16 times. By extending the logic a bit we could combine it with a string variable to get a complete line of code.

```
100 A$=RPT$("F",16)
```

This would set the string variable A$ equal to 16 repetitions of the character "F". We can build on this line of code by adding a CALL CHAR to define the character. The line of code now becomes:

```
100 A$=RPT$("F",16) :: CALL CHAR(32
,A$)
```

Now, the line of code redefines the space character as a solid block (16 repetitions of hexidecimal character "F"). Ok, let's look at some other examples of code that we can use.

```
100 A$=RPT$("A5",4) :: B$=RPT$("F",
8)

110 CALL CHAR(32,A$&B$)
```

Character 32 would now be defined by "A5A5A5A5FFFFFFFF".

RPT$ can be used directly in a character definition as well. In XB, you can define up to 4 characters at a time. If the initial character value is followed by 64 hexidecimal character codes, you define the initial character and the next 3 in succession. So, you could define characters 32-35 all as solid blocks with the following simple line of code:

```
100 CALL CHAR(32,RPT$("F",64))
```

It is a whole lot easier to do it that way than to type in "CALL CHAR (32," followed by 64 "F"'s. Think about that for awhile!

You can even concatenate two RPT$ commands in a character definition. For example:

```
100 CALL CHAR(32,RPT$("F",48)&RPT$
("A5",8))
```

Again, a whole lot easier than typing in 48 "F"'s and 8 "A5"'s to define 4 characters.

In addition to the applications that we have touched upon here, RPT$ and other string variables can be used to output strings to peripheral devices like printers, disk drives and other external files. Simply OPEN the file and PRINT the material with either RPT$ or a string variable depending on your program application.

As the country philosopher said "If the good Lord is willing and the creek don't rise", I'll SEE you at the next UG meeting.

PROGRAM WRITER

By R.M. Carmany
Hunter Valley 99'er Users Group

Before we discuss what this software package will do, we need to discuss why it was written. **PROGRAM WRITER** is NOT the "pot-of-gold at the end of the rainbow". It was written because of sheer laziness!! Why do something yourself when you can have your computer do it for you? The idea of this program and the documentation is to stimulate a bit of thought amongst those of you who have picked up a copy of it. Use this program as a basis to write your own utilities to write lines of code. Believe me, it's easier than trying to find all the "typos" in 300+ lines of XB code! Now to the program!

Perhaps the biggest problem about programming is debugging the program that you have just written. Basically, there are two kinds of "bugs" to deal with. The first is a logic error. Logic errors are created when the program structure is faulty (ie. an improper branch statement). Sorry, **PROGRAM WRITER** won't help you with these! Fortunately, in a well-planned program this kind of bug comprises a very few of the total errors. The predominant error type in XB programming is the simple "typo". How many times have you gotten a "syntax error" to find that you had mis-spelled a word somewhere within the program? It sure would be nice if this type of error could be eliminated at the very beginning instead of having to go back through the program and correcting these silly errors. That was the reason that this series of programs was written -- to take some of the headaches and problems out of writing an XB program. No program is "perfect" and this isn't either. I think that it serves a definite need of being able to write segments of program code that are free from "typo bugs".

PROGRAM WRITER is really a group of four seperate programs plus a **LOAD** program. The package auto-boots from the standard XB environment very nicely. The minimum system requirements are XB, a console, monitor, 32K, and a single disk drive. There are no requirements for anything more and these seem to be the standard basic TI system as the years have worn on.

The programs on this disk are: **WRITER** (the XB line writing program), **TOKEN** (a program to create a list of token values), **COMPACT** (a program to compress the single lines created by **WRITER** into multi-statement lines), and **ANALYZE** which provides a printout of the token values in a MERGE file. **PRINT** prints out **DOCS/1** and **DOCS/2** (these files) through the F'WEB or TI-WRITER formatter.

Let's start with the programs in the order that they would be used.

TOKEN

As you probably know, printable characters take up only a part of the total number of characters available. The first 31 characters (0 - 31) are reserved for control characters. In fact, the only one of these characters that is of use when creating an XB program line is CHR$(0) which sends a "null" and signals the end of a program line. Characters 32 - 127 comprise the range of printable characters. These are the characters that are used for the alphabet, numbers, and various symbols that we see in print. This range includes CHR$(127) which is the DELETE character and, although not visible, is still considered a printable character. The last segment of character values is the one that we are interested in. It is the range from 128 - 255. When an XB program is converted into a D/V 163 file using the MERGE option, the XB reserved words are replaced with characters in this range.

Some years ago, some of TI's "deep dark" secrets started slipping out and making their way into various publications. One of the most promising of these was the way that XB programs were SAVEd in the MERGE format. TI chose to use an intermediate between straight XB and Assembly Language. The D/V 163 files had a good number of the XB reserved words "tokenized". That is, unused character values were used

in place of the actual words themselves. If you asked TI about it on one of the telephone "hotlines" that they had going, a blunt "that is proprietary information" was the response! To be sure, some tantalizing "tidbits" appeared from time to time and you could have compiled the entire list if you were able to keep up with all of them. There had to be an easier way!!

Line numbers are another matter! They are stored as two-byte multiples of 256 (the maximum number of 8 bit combinations). The only exception is when the line number is a reference in the middle of an XB program line --then it is three bytes in length. The formula to create line numbers looks something like this: LNUM=INT(NUM/256)-INT(NUM-256)*INT(NUM/256). This formula gives us line numbers that are whole integers from 1 to 32767. It has been around for a long time and all of the programs that write lines of XB code or read MERGE files use some variant of this same formula. So, by converting the formula to print the line number as a string we come up with:

```
LNUM$=CHR$(INT(LNUM/256))&CHR$(LNUM-256*INT(LNUM/256))
```

Basically, **TOKEN** tricks the computer into thinking that it is writing a MERGE program file to disk. A simple loop is used to input both the line number (using the above formula) and the corresponding CHR$ value. The loop limits are set by user input for flexibility. An "=" is placed between the program line number and the corresponding CHR$ value. The result is a series of program lines that look like this:

```
65 = A
66 = B
67 = C
```

As the program runs, the number of the character currently being processed is displayed and the program will tell the user if the number is a character value (under CHR$(127)) or a token value (above CHR$(127)).

These lines are SAVEd in MERGE format on disk by **TOKEN.** Once you have the file on disk, follow the instructions on the screen to create a D/V 80 file that can be processed by F'WEB or TI-Writer. If you LIST the "program" at this point you will discover that some of the CHR$ values from 128 - 255 have garbage instead of a token value. These are either unused values (there are several of them) or token values that are used as flags. Fortunately, the values that are used as flags are right together. CHR$(199) is used to tell the computer that a quoted string is to follow (ie. "Hello, how are you?") and that it should be enclosed by paired quotation marks. CHR$(200) informs the computer that an unquoted string is to follow (ie. 32) and it should not be enclosed by quotation marks. CHR$(201) tells the computer that a line number reference is to follow and the line number algorithm is coming next. So, when you use your word processor to manipulate the D/V 80 file, put something appropriate in place of the garbage beside these CHR$ values. There will be more discussion of these values and how they are used later in this documentation. After a brief delay, LOAD will be re-run for further processing.

WRITER

WRITER is the **PROGRAM WRITER** selection on the main menu. There are two distinct segments within the program itself. The first screen that you will encounter will allow you to enter the "set-up" options. These are simple inputs of the beginning line number, line increment (updated after every line of code if processed), and the device and filename for your output file. After that is done, the selection menu will appear. It looks like this:

1) DISPLAY AT
2) ACCEPT AT
3) CALL STATEMENTS
4) CALL STATEMENTS II
5) CALL LOADS
6) MISCELLANEOUS
7) SAVE AND EXIT

DISPLAY AT

DISPLAY AT is a dedicated section that allows the user to define the DISPLAY AT statement by simply selecting a choice from the menu. All of the options possible with DISPLAY AT are available. A menu screen is presented with a number of choices:

1) NONE
2) ERASE ALL
3) ERASE ALL BEEP
4) ERASE ALL SIZE
5) ERASE ALL BEEP SIZE
6) BEEP
7) BEEP SIZE
8) SIZE

The first thing to do is to enter your choice by entering the number at the prompt. Then, you will be allowed to enter the row (1 - 24) and the column (1 - 28) for your DISPLAY AT coordinates. There is some error checking at this point and "out of range" values will not be accepted.

If you have chosen one of the options that includes SIZE, the screen will clear and you will be prompted to enter the field-length of your text. Again, the program will check and you will be unable to enter a value less than 1 or greater than 28. This step is skipped if your chosen option doesn't include SIZE and you will go directly to the text entry prompt.

The next step is a prompt for the entry of your text string. DO NOT USE QUOTATION MARKS!!! That will be taken care of when the program writes the line of code to disk. If you have chosen an option that includes SIZE, the program will not let you enter a string longer than the field-length. Otherwise, the maximum length for your string is 28 characters. There is currently NO checking for line wrap (it maybe added to a later version) so plan the number of characters carefully!!

At this point, the line of code has been written to VDP and is ready to be written to disk. The program will prompt you to see if you want to write another DISPLAY AT line immediately. If you answer "Y" then you will be given the opportunity to construct another line, etc. A "N" at the prompt takes you back to the main menu screen for other options or to save what you have written to disk. The code for the line that you have just constructed remains in VDP RAM along with any others that you may have written until the SAVE AND EXIT option is chosen at which time it is written to disk.

ACCEPT AT

ACCEPT AT is similar in structure to the DISPLAY AT segment. There is, however, a notable difference. The first screen that is presented is a 15-choice screen detailing all of the VALIDATE options. All of the combinations are available including the facility to create your own user-defined limits (ie. ACCEPT AT(10,10)VALIDATE("YN") . .). These user-defined options are designated by DEF in the selection. Once again, DO NOT use quotation marks --- the program will take care of that! The only limitation in a user-defined validation string is a limit of 12 characters. If you do not desire to use VALIDATE, simply enter "0".

Like DISPLAY AT you will be able to enter the row/column coordinates after you have made your menu choice. And there is error checking to eliminate unacceptable values.
After you have made your VALIDATE choice and entered your coordinates, a second screen will appear that is identical to the DISPLAY AT screen. Again, make your choice with a single keypress. If you have chosen an option with SIZE, you will go to a field-length prompt. Otherwise, you will go to the final two prompts.
The first of the final prompts will ask you for a variable name. IT MUST BE TWO CHARACTERS IN LENGTH. There are a couple of reasons for this. First, if a variable name is longer than two characters, it can slow program execution. Second, two characters increases the number of possible combinations. After you have made your entry, you will be asked if the variable is a numeric variable (able to accept numbers only). If you answer "Y", you get a numeric variable in the line of code that is written. If you answer "N" then a string variable will result (able to accept text) and the "$" will be appended to the variable name when the line of code is written.
Once again, the line of code is stored in VDP ready to be written to disk and you will be asked if you wish to write another ACCEPT AT line of code. It's just like DISPLAY AT.

CALL STATEMENTS

This program segment contains a series of CALL statements. Some require a value or values to be entered from the keyboard and some do not. This is what the menu looks like:

```
 1) CALL HCHAR
 2) CALL VCHAR
 3) CALL SCREEN
 4) CALL MAGNIFY
 5) CALL CLEAR
 6) CALL INIT
 7) CALL CHARSET
 8) CALL KEY
 9) CALL SPRITE
10) CALL PATTERN
11) EXIT TO MAIN MENU
```

Simply make your selection with a single keypress to write the line of code. In the case of those options that require an input, you must enter the appropriate values before the line of code is written to VDP. Once again, the program will not accept any values outside of the acceptable range.
CALL KEY will generate a multi-statement line when it is selected. It assumes that if no key is pressed (ie. S=0) that the program is to wait for input. It generates the following type of line (your line number will be different depending on the progress within the program):

```
600 CALL KEY(0,K,S):: IF S=0 THEN 600
```

Another compromise was made in the case of HCHAR and VCHAR. The number of repetitions must be entered instead of being optional in single character entries.

CALL STATEMENTS II

This program segment is a continuation of the previous segment and contains options to allow the generation of user-specific CALL statements. Once again, you will be prompted when an input is required and there is error checking throughout to eliminate "out-of-range" values. The menu looks like this:

```
1) CALL MOTION
2) CALL COLOR
3) GENERIC CALL I
4) GENERIC CALL II
5) GENERIC CALL III
6) EXIT TO MAIN MENU
```

The first two options generate a line of code just as the XB manual requires. The next three options are of particular interest. Option 3 (GENERIC CALL I) requires the input of a user-defined subroutine (ie. CALL WAIT). There are no other parameters allowed. Option 4 (GENERIC CALL II) does much the same except that it will generate a line of code with two NUMERIC parameters enclosed in parentheses. The line of code would look like this:

```
500 CALL WAIT(10,20)
```

Option 5 (GENERIC CALL III) allows the user to enter two variable names after the CALL (ie. CALL ERR(ES,ET). With these three generic CALL options, it should be possible to enter almost all of the CALLs that were not included in the previous two program segments.

CALL LOADS

There are several options here that are all menu-selectable. This is what the screen looks like:

```
1) DISABLE FCTN=
2) ENABLE FCTN=
3) QUIT & RUN DSK1.LOAD
4) GENERIC CALL LOAD I
5) GENERIC CALL LOADS II
6) EXIT TO MAIN MENU
```

The first three options are self-explanatory. They write a line of code using CALL LOAD that does exactly what the option says. Option 4 allows the entry of a user-defined CALL LOAD. The only restriction is that that it will be negative (ie -31806) but you need not include the minus sign --the program will do it for you. The second parameter is limited to 1 number 3 digits or less. Similarly, Option 5 allows the entry of the address and two parameters of 3 digits or less. After the entries are made, the line of code is written.

MISCELLANEOUS

This program segment presents a group of options that really do not fit any where else. The menu looks like this:

```
1) DELAY LOOP
2) ON ERROR (Linenum)
3) ON WARNING NEXT
4) OPTION BASE 1
5) OPEN #1:PIO
6) EXIT TO MAIN MENU
```

This segment is patterned after the previous segments and the line of code is processed when the appropriate key is pressed. The "DELAY LOOP" choice will generate a multi-statement line based on the loop limit that you have chosen. An example would be:

```
500 FOR DELAY=1 TO 500:: NEXT DELAY
```

This would be based on an input of "500" as the loop limit, of course. There is one minor problem associated with the DELAY LOOP entry. It seems that the CHR$(0) (null) used to end each program line generates a syntax error when it is used with this menu choice. I have tried by varied and devious means to eliminate this problem without success. Rather than eliminate the choice entirely, I decided to keep it and pass along the "fix". The line of code will appear correct but BEFORE YOU RUN it, tab your cursor to the end of NEXT DELAY and press FCTN-1 (Delete) and the problem is solved. This solution must be used with each of the delay loops generated by **WRITER.**

SAVE AND EXIT

This is the last option on the main menu. Before we look at it, let's analyze how the computer organizes memory with the 32K attached. The memory expansion unit is divided into two parts. The first 8K is used to store assembly language programs. The upper 24K is where your XB program resides and runs from. That leaves the 16K of VDP RAM free to store variables and strings. This is where the lines of code you have written are stored. They aren't saved to disk until you exercise this option. So, after you have created several lines of code and are ready to move on to something else, choose SAVE AND EXIT and everything will be neatly saved to disk. If you are writing a long application, the program will automatically write the stored lines to disk when the VDP buffer is full. Depending on the length of the individual lines, that is about every 10 lines of code.

Incidently, the program is constructed so that you can write a couple of DISPLAY AT statements. Go back and write a corresponding ACCEPT AT statement and then add anything else that you wish from any of the main menu options. Then exercise SAVE AND EXIT and the lot will be saved to disk. The line numbers will begin at the one you chose in the setup options and they will be incremented automatically before the next line of code is processed.

WRITER will "crash" if you do not have a disk in drive 1 for it to re-boot LOAD when you choose the SAVE AND EXIT option and the BREAK key (FCTN-4) is active throughout the program. But at this point, everything has been saved to disk anyway and no damage can be done. It provides a quick way to quit if you do not want to wait for LOAD to re-boot.

COMPACT

This little "gem" has been around for quite some time. There are other programs that will do the same thing but not as simply and easily. The author of the program is J.R. Dew and **COMPACT** has been making the rounds of the public domain circuit since 1985 at least. There were some changes and modifications made to the original copy that I had to make it fit into the menu system of **PROGRAM WRITER.**

COMPACT takes a D/V 163 file (like the output from **WRITER**) and combines single lines into multi-statement lines. For example, if you wrote four or five short DISPLAY AT statements with **WRITER** to a disk file and then run it through **COMPACT**, you would see the lot compressed into a single line!! Just write as many statements with **WRITER** as you wish and then run everything through **COMPACT** and RESequence. It enables the user to compress the lines that have been created and end up with a faster running program.

The program is menu-driven and if you have been able to follow everything so far, it won't be any trouble to use.

ANALYZE

ANALYZE is a neat little program written by Tony Kleen of the Guilford 99'ers UG and first appeared in the April 1989 issue of their newsletter. Once again, some error trapping and cosmetic changes were made to the original. It is, however, the product of Tony's imagination! **ANALYZE** takes a MERGE file and dumps the tokens that make up each line to the printer of your choice for further analysis. The tokens are presented in the form of a three digit number seperated by a colon. It will first dump the line number, of course. The output looks like this:

```
000:100:157:200:004:067:072:065:082:. . .. .
```

The null at the end of each line is displayed (ie. 000) and the end of file marker is also displayed (255:255).
The program is prompt-driven. Simply enter the input filename at the appropriate prompt and then your printer parameters if they are different than the default PIO. Like the other programs in the series, it will reload **LOAD** from DSK1 when it is through. This little beauty should be of help in analyzing lines of code that you have created or just taking a peek at MERGE files!

PROGRAM NOTES

Writing a program like **WRITER** was an "interesting" experience. It gave me the opportunity to do one of my least favorite things ---work with strings. Programs that write lines of code invariably write the code as string expressions and they are not the easiest things to manipulate. But it can be done!
One of the first chores that has to be done is to allow for the input of a beginning line number for the starting point of your program. I chose to use a simple "setup" screen for the input of the beginning line number (LN) and the line increment (I). The next order of business was to allow for the input of the device/filename. So far, the program was coming along great! A real piece of cake!!!
I had my basic values input into the program, the output file was opened and I had the string defined to write my line numbers. I just had to remember to increment the line number with a LN=LN+I statement after each line of code was processed.
Now, for the actual construction of a line of code! Let's start with a particularly EASY one. How about ON WARNING NEXT?

```
PRINT #2:CHR$(INT(LN/256))&CHR$(LN-256*INT(LN/256))&CHR$(155)&CHR$(166)
&CHR$(150)&CHR$(0)
```

Print to file #2:Line number & token for ON & token for WARNING & token for NEXT & null to end the line

That was easy! The only trouble is that lines of code like that are not the ones that save the most keystrokes. The more complicated the line of code, the more keystrokes are saved (and the greater the chance of a typo if you have to type the line in by hand).

CHR$(200)

CHR$(200) is the flag that is used to tell the computer that an unquoted string is to follow. That is a string expression that is NOT to be enclosed in quotation marks. There are certain requirements that must be met before the computer can correctly process the string. First, you must tell the computer how many characters there are going to be in the string by sending a value in the form of CHR$(X) where "X" is the string length. Then, if the variable is a numeric variable, it must be converted into a string expression with the STR$ function. Let's look at a couple of examples:

```
PRINT #2:CHR$(INT(LN/256)&CHR$(LN-256*INT(LN/256))&CHR$(157)&CHR$(200)
&CHR$(6)&"SCREEN"&CHR$(183)&CHR$(200)&CHR$(LEN(STR$(SCR)))&STR$(SCR)
&CHR$(182)&CHR$(0)
```

Print to file #2:Line number & token for CALL & flag for unquoted string & length byte for following word & SCREEN & token for open parenthesis & flag for unquoted string & length byte & string of SCR & token for closed parenthesis & null to end the line

The explanation of the code is simple. The value of SCR can be from 1 to 16 and, therefore either 1 or 2 digits. So we used CHR$(LEN(STR$(SCR))) as the length byte to take care of either case. Then we printed the string of SCR with STR$(SCR). Notice that the token values for the opening and closing parentheses were used rather than the character values for the printable versions. The CHR$(200) tells the computer that the STR$(SCR) should not be enclosed in quotation marks.
Notice that we used the flag for an unquoted string for the word "SCREEN" as well as for the numeric value. You MUST use CHR$(200) for anything that appears in the line that is not a token value or that you do not want enclosed in quotation marks. And, a length byte must precede the word itself as well! This was a lesson that was learned the hard way!!!
Let's look at one more --- ACCEPT AT(R1,C1)SIZE(SZ) . . We will ignore the rest of the line (hint: look at a listing of the **WRITER** program).

```
PRINT
#2:CHR$(INT(LN/256))&CHR$(LN-256*INT(LN/256))&CHR$(164)&CHR$(240)&
CHR$(183)&CHR$(200)&CHR$(LEN(STR$(R1)))&STR$(R1)&CHR$(179)&CHR$(200)&
CHR$(LEN(STR$(C1)))&STR$(C1)&CHR$(182)&CHR$(235)&CHR$(183)CHR$(200)
&CHR$(LEN(STR$(SZ)))&STR$(SZ)&CHR$(182). . . &CHR$(0)
```

Print to open file #2:Line number & token for ACCEPT & token for AT & token for open parenthesis & flag for unquoted string & length byte & string of R1 & token for comma & flag for unquoted string & length byte & string of C1 & token for closed parenthesis & token for SIZE & token for open parenthesis & length byte & string of SZ & token for closed parenthesis . . . null.

It really isn't as difficult as it seems -- believe me!!

CHR$(199)

CHR$(199) is the flag to tell the computer that a quoted string is to follow. It also uses a length byte but since text is usually input into a string variable, there is seldom a need to use STR$ to convert a numeric variable to a string. Anyway, the principle for the use of CHR$(199) is the same as for CHR$(200). Here is an example -- DISPLAY AT(R1,C1):TEXT$ -- where TEXT$ is a previously input phrase of variable length.

```
PRINT #2:CHR$(INT(LN/256))&CHR$(LN-256*INT(LN/256))&CHR$(162)&CHR$(240)
&CHR$(183)&CHR$(200)&CHR$(LEN(STR$(R1)))&STR$(R1)&CHR$(179)&CHR$(200)&
CHR$(LEN(STR$(C1)))&STR$(C1)&CHR$(182)&CHR$(181)&CHR$(199)&
CHR$(LEN(TEXT$))&TEXT$&CHR$(0)
```

Print to file #2:Line number & token for DISPLAY & token for AT & open parenthesis & flag for unquoted string & length byte & string of R1 & token for comma & flag for unquoted string & length byte & string of C1 & token for closed parenthesis & token for colon & flag for quoted string & length byte & TEXT$ & null

This example shows how both flags can be combined in a single line of code. If you examine the **WRITER** program, you will see that none of the lines of code are as long as these examples. The fact is, that by using numeric variables and reducing common code into a string variable, the line length can be greatly reduced. The following string definition reduces everything in the DISPLAY AT statement to the parenthesis after the row and column values to a single string variable.

```
DIS$=CHR$(162)&CHR$(240)&CHR$(183)&CHR$(200)&CHR$(LEN(STR$(R1)))&STR$(R1
&CHR$(179)&CHR$(200)&CHR$(LEN(STR$(C1)))&STR$(C1)&CHR$(182)

CHR$(201)
^^^^^^^^^^
```

This is the last of the flags that we will discuss. The procedure for using it is similar to the other two that we have previously seen. For example, to write a line of code like ON ERROR (linenumber) would be written like this:

```
PRINT
#2:CHR$(INT(LN/256))&CHR$(LN-256*INT(LN/256))&CHR$(155)&CHR$(165)&
CHR$(201)&CHR$(INT(LNUM/256))&CHR$(LNUM-256*INT(LNUM/256))&CHR$(0)
```

Print to file #2:Line number & token for ON & token for ERROR & line number based on LNUM & null

As you can see, this flag is just a bit different than the other two. It uses the line number algorithm following the flag itself instead of string manipulations (is that a relief!!). The only change is to substitute the numeric variable that you used to input your line number reference in the original formula (see page 2).
One more note. If you used **TOKEN** to prepare a list of the token values, you will notice that CHR$(253) is "#". This is the "#" that is used for sprite numbers and file numbers, etc. It should be used whenever the "#" is required for these uses. CHR$(35) is NOT interchangable with it.

FINAL WORDS

I realize that the code for **WRITER** is not "classic" in the sense of structured programming. It was written that way with a specific purpose in mind. None of the code has been optimized to make the structure of the individual sections as transparent as possible. Certainly, the code could be significantly compacted to save program space and increase execution speed slightly. Remember, this software package is meant to be a starting point for your own explorations of "programs that write programs".
The setup options were followed by DISPLAY AT and ACCEPT AT and then the main menu was written. The program ending portion followed and then CALL STATEMENTS, CALL STATEMENTS II and CALL LOADS were written. In fact, the DISPLAY AT and ACCEPT AT portions of the program were used to write some of the program segments of **WRITER** after they were finished. The result is not the best-structured program. I find it much easier to enter a DISPLAY AT statement with a couple of keypresses than to type the whole thing in. Invariably, DISPLAY AT becomes DSIPLAT AT as my fingers sometimes outstrip my mind. Those are just the kind of errors that I hope to eliminate with this program. Other segments could be written to make the program more complete and even more versatile than it is currently.
I tried to weigh the relative value of each of the prospective options as I contemplated writing the code for each. CALL CHAR seemed like a choice until I realized that there were just too many internal options available and the number of keypresses required to enter the necessary

THE INFORMATION PAGE

GENERAL MEETING.

The March general meeting will be held on 27 th at Jesmond Community Centre. We are currently trying to arrange a demonstration of Viatel the Telecom computer database and personal communications system. Also hope to be able to show two TI's talking via their RS232 interfaces. Hope you can make it along to these demos.

COMMITTEE MEETING.

The second tuesday in March sees the next committtee meeting at the Boolaroo ambulance station. The proposed demonstration for the march meeting will be discussed and details finalised, subject of course to Telecom coming to our aid. Also we will be doing some more work on the Computer Show at Newcastle University, we see our group manning a stand again this year. We are looking for ideas for the stand. Please feel free to attend this meeting and have you input to the running of your User Group.

EXTENDED BASIC CLASSES.

Again Bob MacClure's house on the third tuesday, (20-3-90). The group is into writting a programme to help Don Dorrington do some analysis of lotto results. Hope it can also help Don strike it lucky.

FORTH CLASSES.

Richard and Joe have not yet got their act together as yet to get these classes started for 1990. Joe is having trouble finding time to do much computing at all and Richard has also been very busy. Hopefully though a class will be held on the first tuesday in March 6/3/90. Please ring Joe before attending, hopefully they will be at Richard's surgery gain this year. Time is a valuable item around here lately!!!!

CALENDER 1990.

Last month I mentioned some proposed activities, WELL! Bob MacClure informs me that the theatre I had mentioned was damaged in the earthquake and might not re-open. Hope to have more info for the next newsletter, if we can't get to see those shows we can always go somewhere else. Keep and eye on this page for more thoughts on the matter.