

HUNTER VALLEY 99ERS USERS GROUP HOME COMPUTER NEWSLETTER



Happy Christmas!

REGISTERED BY AUST POST
PUBLICATION No. NBG8023

DECEMBER 1989



YOUR COMMITTEE

PRESIDENT

Peter Smith
8 Glebe St.
East Maitland 2322
Phone 336164 V/t1 493261640

VICE PRESIDENT

John Paton
1 Parlen Close
Rutherford 2320
Phone 326014 V/t1 493260140

SECRETARY

Brian Woods
9 Thirlmere Pde.
Tarro 2322
Phone 662307 V/t1 493223070

TREASURER

Noel Cavanagh
378 Morpeth Rd.
Morpeth 2321
Phone 333764

SOFTWARE LIBRARIAN

Stewart Bradley
14 Hughes St.
Birmingham Gardens 2297
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 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.

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



PRESIDENT
PETER
REPORTS

As our community gets smaller it's good to hear of all the things happening around the world and here in Australia.

Reading of portable TI's, 8001 Graphics cards, super new programs and our own ever-improving, ever-fantastic "Funnel-web", hard disk adaptations, new data-bases and a host of other developments makes one wonder just why our computer just went "lay down and die" (must be doing something right somewhere eh?)

Ron Kleinshafer has come to the rescue of some club members again with his great electronic knowledge. He has, he thinks, corrected a small problem in some Quest-Ram-Disks which some members have. It appears that their probs may be over and I look forward to Ron's explanation appearing in the newsletter.

Our prolific-writing U.S.A. member, Bob Carmony, is heading this way early next year and we are busy organising a reception which will hopefully make him feel extremely welcome. An exciting time to look forward to as he has promised some nice pressies as surprises.

Joe has included a list of dates elsewhere in this newsletter which will show you when our next meetings are to be held.

Our children's christmas party was held at the Ambulance station at Booragul on 3rd December. Games and activities were organised for all and with no computers available we even had a good time.

Christmas is fast approaching and I guess it is time for me to wish you all a merry, and most importantly, a SAFE christmas season on my own and my family's behalf and on behalf of the club's committee.

We need you with us so remember ".if you drink... don't drive."

Review of TI-VIDITEL2

It's amazing how we all put up with things until we see something better or something that does a job better.

Viditel is a program which tormented me for some time until I finally obtained a copy from our club's librarian.

Having had my appetite wetted by John Paton with his description of the program's capabilities, I searched fruitlessly for sometime, until one night, a kind-hearted HV99er who had been THE ONE I wanted.

Running it later that night I was disappointed to find that I couldn't read the instructions or any of the screens.. Talk about "double-dutch"! Seriously it was ordinary dutch, and by good fortune my wife new an exchange student from the Netherlands at her school.

She was soon co-opted for the translation job, and a fine job she did too for a non-technical person. (I am still adapting the program so that we English speaking users can enjoy it as much as those clever Dutch people.)

The author, Bob Templemans-Plat, has provided his program with a number of capabilities which most of our VIDITEXT type programs have sadly lacked. Features include

4 DIFFERENT SCREEN DUMPS

SUBSTITUTION of "ENTER" key for
<shift><#>

SAVE SCREEN TO DISK

REPLAY SAVED SCREEN.

CONFIGURE SYSTEM

TRUE COMPATABILITY with VIATEL

COLOUR CODED WARNINGS.

I particularly like the program because it allows me to use a new board, whereas my old program (by the same author) would not .

The program handle VIATEL beautifully and being able to press <ENTER> at the end of an input (as opposed to <SHIFT><#>, sure makes life easier (and less complicated as I would often press <ENTER> when a # response was expected by VIATEL, leading to muchos confusion!!!???)###).

Screen dumps come as the usual small size in positive or negative form (black background, white letters) or full-sized positive/negative, all with the use of <CTRL><1>/<2>/<3> or <4>.

<CTRL><5> provides an ASCII screen-dump "which should work on

all printers" , but will not provide graphic images.

The substitution of <ENTER> as mentioned previously, makes it necessary (only sometimes.. like when setting up the modem etc) to send an <ENTER> signal from the key-board. To do this use simply <CTRL><E>.

I am really impressed by the ease of using this program and the way it operates and I would recommend it to anyone using VIATEL with our machine.

EDITOR'S LAMENT.

LATE!! by the time you read this newsletter Christmas will have well and truly gone. We pushed the general meetings back to the fourth tuesday in the month earlier in the year. This of course lead to the newsletter deadline being pushed back to the third tuesday of the month. Unfortunately the printers knock off early for christmas. Thus the combination of later deadline and printers going for an early break has left me stuck for getting the newsletter out before christmas. Please accept my apology, I do hope that you find this issue well worth the wait.

1989 has been a mixed year for the HV99'ers. We hit a low point during the early months of the year. The Annual general meeting in June seen the Group remarshal it's forces and we have proceded strongly into the christmas period.

What 1990 holds for us anybody can guess, there is no doubt that the group is tiring and shrinking. There is also no doubt that the thread which holds us together is the newsletter. Input is still strong and varied. Perhaps therein is the message for us all, keep the newsletter strong and we will still exist, let the newsletter die and the group will without doubt die.

I have used this column as a hole filler though out the newsletter, you will find more of my moaning and groaning sprinkled through out it's pages. Thought you would get away with only one thing to read by me didn't you!! Ha!!

BLACK VELVET.

by Al Lawrence

Out of MOURNING for a departed KING came a wonderfull drink which was in " daze " gone bye, drunk on the battlefields at the wake by the IRISH to usher H.M. into the nearby hereafter.

Passed down from the survivor to the survivor's sons and daughters, ect!.

I now make this freely known as a JESTURE of goodwill for the festive season to all TI'ers around the TI world at no e"X"tra cost. So the pirates can feel free to pass this " Publick Domain " Tipple for all those with access to" GUINNESS and CHAMPAGNE " the perfect partners at a wake for a " Lil 'old FOUNDLING " who still lives 6 years down the track HIC! HIC! HOORAY!.

As this was all they had in the Quarter Masters store on the day in question the IRISH not wanting to pass up the wake and be thought ignorant made do and invented an historic drink.

Don't worry if you do not have the GUINNESS.! then any home brew or a commercial BLACK STOUT will do.

Have a glass for CLAGG or a bucket if putting a " Shrimp on the Barby as they say down under ".

Mix the twain in equal portions and pour into a suitable size MUG if you are still mourning or still celebrating some EVENT in history DRINK and ENJOY but as usual the AUTHOR and all " disinterested " persons accept no responsibility or liability for any damage to any property, persons known or unknown as the direct result of the use or misuse of any the information contained therein.

AND A HAPPY HOGMANNY TO ALL.

Futher modification of this receipt renders all warranties null and VOID like your head if you misuse the secrets of the long departed.

thank you from MESELF.

This pro uti yea int as and gen pro

Bot man use wit opt -FL the pro app ele inc nee mos SYN for

Exa wil

(SYSTE

BASE->R
BASE->R
: VSBW
: VSBR
: VNTR
: XNLLM
: CLS A
: VFILL
: VIXR
CODE MD

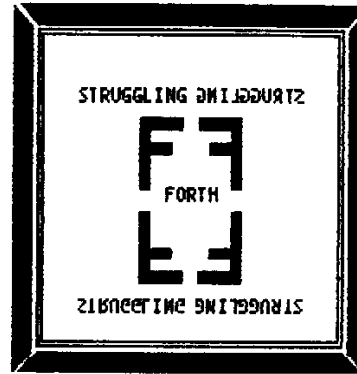
: RNDW
: RND R
: RANDB
R->BASE

SCR #144

```

0
1 -
2 -
3 -
4 - STRUGGLING FORTH
5 -
6 - HV99ERS DECEMBER/89 ARTICLE
7 -
8 - MORE GENERAL PROGRAM CODE
9 - NUMBER AND STRING INPUT
10 - FORTH SCREEN I/O ERROR ROUTINES
11 - HOMEWORK!!!
12 -
13 -
14 -
15

```



This month I'll again break my promise to release my Forth<>files utility and leave that until the new year. We will continue our journey into the realms of program writing as promised in the October magazine, and look a bit deeper into writing general codings useful in any program.

Both the Forth and Superforth manuals emphatically state that the user should not only not muck about with the kernel words, but load the options needed in the program Eg -FLOAT, -COPY, -VDPMODES etc in their entirety, and write your program over the top. I find this approach unsatisfying and lacking elegance. What I tend to do is only include those definitions I really need in my application. For example most applications need the so called SYNONYMS, defined after boot-up in forth as

```
: -SYNONYMS 33 LOAD ;
```

Examining screen 33 from your manual will reveal the following:

```

( SYSTEM CALLS 09JUL82 LCT) 0 CLOAD RANDOMIZE
BASE->R DECIMAL 74 R->BASE CLOAD ;CODE
BASE->R DECIMAL
: VSBW 0 SYSTEM ; : VMBW 2 SYSTEM ;
: VSBR 4 SYSTEM ; : VMBR 6 SYSTEM ;
: VMTR 8 SYSTEM ; : GPLLNK 0 33660 C! 10 SYSTEM ;
: XMLLNK 12 SYSTEM ; : DSRLNK 8 14 SYSTEM ;
: CLS 16 SYSTEM ; : FORMAT-DISK 1+ 18 SYSTEM ;
: VFILL 20 SYSTEM ; : VAND 22 SYSTEM ; : VOR 24 SYSTEM ;
: VXOR 26 SYSTEM ; : HEX
CODE MON 0200 , 4E4F , 0201 , 2000 , CC40 , 0281 , 4000 , 16FC ,
      0420 , 0000 .
: RNDW 83C0 DUP 8 6FES * 7AB9 + 5 SRC DUP ROT ! ;
: RND RNDW ABS SNAP NOD ; : SEED 83C0 ! ;
: RANDOMIZE 8802 C0 DROP 0 BEGIN 1+ 8802 C0 90 AND UNTIL SEED ;
R->BASE

```

In many applications you will actually be using only a few of these definitions Eg VSBW, VMBW, CLS, MON, but may not need VWTR, FORMAT-DISK, VAND, VOR, RANDOMISE etc. If you use this screen in its entirety, your dictionary once compiled will be cluttered full of words you never ever use, taking up valuable byte space. What I tend to do then, is to re-type the codings of only what I need onto my forth screens, omitting those inapplicable and adding a few others of my own. Refer to screen 145 as an example -written in Super4th but the principle is identical in Forth.

In fact prior to compiling my application I FORGET most of the dictionary back to the kernel of words, which in superforth leaves me over 22,000 bytes and in Forth over 17,000 bytes of programming space. Add to that the 11,000 odd bytes of usable VDP Ram, and the 5-6,000 bytes of disk buffer area, the total space for your use adds up to a respectable 33-39,000 bytes depending on your forth version. As the contents of the two versions kernels are different, different sets of words will be needed in your application. In Superforth you can easily forget out the IF's WHILE's etc, also wiping out little used words such as some of the synonyms, random words, in the process. On Screens 146 147, I've included the definitions of IF, THEN, ELSE etc, which are interesting to examine and to see how many bytes they take up. Substitute IMMEDIATE for IMM for Ti-forth.

In the next jumbled screen 148. I include some words which may differ from program to program. Though I often tirade against such messy code, once you have got your routines debugged and sorted out, and you are not passing them on to others, writing compressed non commented code does save space and time in compiling. This screen is a typical example of the first screen in one of my programs - containing whatever synonyms I need, Bsave and a few other useful words. BSAVE is needed of course or you won't be able to save your program! Perhaps I'd include !", FORGET, VCHAR, HCHAR, SCOPY, and SMOVE, but again only if my application uses them. You will find their definitions in the Ti-forth manual or on the distribution disk.

Other words I've found useful include L>U (convert lower case to upper - expects an ascii code), NDROP (expects n - drop next n stack values), NDUP (expects n - duplicate top n stack values), PICK. <> (not equal), ZOVER etc.

KEY-DETECT ROUTINES

To enable your program to flow, you will need to develop routines to detect various key presses. These will use either the forth routine ?KEY when you want to detect a key press without a cursor flashing on the screen, or KEY when you want the cursor to come up flashing waiting for an input which will be EMITed to the user such as at a Y/N prompt. I will expand on these routines as they are quite complicated if they are to be kept general.

?KEY-ALLOW (Screen 149)

This takes the form of an indefinite BEGIN...UNTIL loop. You can put on the stack any number of key combinations you wish to detect followed by the total number Eg:

```
: Function5,6,9
```

```
12 14 15 (ascii of these keys)
3          (3 keys to detect)
?KEY-ALLOW
;          (leaves ascii of key)
```

It is worthwhile dissecting this routine. It uses ?KEY to scan the keyboard, converting lower to upper case for those instances detecting

aA bB etc. validates the detected key is one of those the routine expects and if ok leaves a copy of the ascii code and a 1 on the stack for UNTIL. If an incorrect key has been pressed the ascii value is dropped and a 0 left for UNTIL to force a return to BEGIN. There is no exit until a correct key is pressed!

KEY-ALLOW (Screen 151)

This is almost identical except for the complication that once a key is pressed, forth automatically increments the cursor position, hence the routine stores the original position for re-use if the input is not correct.

Screens 153/154 show examples of these versatile routines. On its simplest level KEY-ALLOW can help allow only a single input eg 1 or differentiate between two options such as in the definition

```
: Y/N 78 89 2 KEY-ALLOW ;
```

MORE COMPLICATED INPUTS.

(It's times like these I wished I'd seen a Lutz Winkler tutorial on the subject! If anyone has one please pass it on as I don't pretend to be an expert here. the codings of my routines are ghastly as they were written during a period of frustration eons ago and have never been updated as they work ok).

?KEY-ALLOW and KEY-ALLOW are fine for simple inputs, but things start getting vastly more complicated once we have to accept say a string with a count for example if you wanted to input and validate a printer name or say a 3 digit screen no.:

```
RS232.BA-
```

```
342.
```

At this point I've got to confess I've never found this easy to do. I've always felt this is one of the paradoxes of FORTH, ie that one is able to do incredibly powerful things with a couple of simple bytes such as pull in a disk block and flash it to the screen, yet number and string handling (in my hands at least), is difficult. The level of difficulty depends upon the level of validation you want. If you are

writing a program you do not want to release, and can predictably enter you data without error, then your routines need not have much in the way of error checking. Once you write the program for others it is mandatory to use error handling. For example if your prompt is expecting a three digit number, and the user enters ABC, when the forth word NUMBER tries to convert this to a number the program would crash!

STRING INPUT.

By this I mean a string of numbers/letters preceded by a count. Lets say we have accepted the data to 6 consecutive bytes of memory, it would look like this

```

Bytes
-----
0 1 2 3 4 5
5 M O U S E

```

The simplest way is to use EXPECT with which you have all experimented, if not, do so. There are several problems with this word (see Struggling Forth June 88). Firstly it doesn't leave a preceeding count, it ends the input in nuls, not ascii blanks needed by NUMBER, and lastly and perhaps most fatally due to the bad kernel coding, it allows the input of the down arrow key, effectively scrolling your input off the screen, and the program with it. Try this, type:

```
0 0 960 65 HCHAR PAD 80 EXPECT
```

then at the flashing cursor hit function X. Imagine the screen full of A's is your cleverly designed well presented program screen disappearing before the users eyes and you'll get the picture. If you must use expect use either that listed in the above mentioned article or the fix coded on screen 155-156. Unless you write a completely different EXPECT you'll need this anyway for one version of our later string words.

GET\$

Refer to screen 150 bearing in mind the forth equivalent of TRL is -TRAILING. Many inputs will already be displayed on the screen as

defaults with the cursor flashing over the first character. My word allows the input checking for arrow keys and once enter is hit does a fixed length VDP read which is then converted to the correct count by using -TRAILING, and an ascii blank added in case NUMBER is later used on the string. Once you have a string preceeded by a count it is easy to print out data by adr COUNT TYPE; check the characters are all digits; use NUMBER to leave a valid number on the stack; or check for specific inputs such as DSK1. etc.

Other string handling words include ADD\$, SEG\$, MOVE\$ as in these older definitions written back in 1985:

SCR #7

```

0 ( STRING:Basic Word definitions 18Jly85) ( STACK EFFECTS)
1 : GET$ TIB # SWAP EXPECT 0 IN ! 13 WORD ( Adr to put new )
2 HERE OVER OVER CE DUP ROT C! 1+ ! ( string,count )
3 DO 1+ OVER OVER CE SWAP I + C! LOOP ( ---- )
4 DROP DROP ;
5 : MOVE$ SWAP COUNT !+ SWAP I- ROT ROT CMOVE ! ( Adr1 Adr2--- )
6 : ADD$ DUP >R SWAP COUNT ROT COUNT DUP ROT
7 + SWAP ROT DUP ROT + R) C! CMOVE ; ( Adr1 Adr2-- )
8 : SAME$ COUNT ROT COUNT ROT MAX I SWAP 0 ( Adr1 Adr2--flag
9 DO >R DUP CE ROT DUP CE ROT = ( Where these adr
10 R) MIN SWAP I+ ROT I+ ROT ( contain strings
11 LOOP >R DROP DROP R) ! ( with counts
12 : SEG$ ROT DUP >R OVER SWAP C! SWAP ROT + R) ROT 0
13 DO OVER OVER SWAP CE SWAP I+ C! I+ SWAP I+ SWAP
14 LOOP DROP DROP ! ( From adr, adr to put, start,num-- )
15 : GET$I OVER OVER SWAP C! SWAP I+ SWAP EXPECT ; ( adr,count--- )

```

It is worthwhile referring to HV99ers newsletter no.5 from back in 1985 prior to the newsletters having the month on the cover, as the string article goes into the subject in great detail.

@DIGITS (screens 152-153)

As an example of combining string input and digit validation this is a general word which expects a lower limit, upper limit, max number of digits to allow, start col, start row --- and leaves the number on the stack.

```
EG 5 10 2 0 0@DIGITS
```

will allow the user to enter a 2 digit number between the values of 5 and 10 starting at row 0 column 0 of the screen. This routine has a multitude of uses for example simple validation in a situation needing the user to input disk drive numbers, through to inputting page numbers, or forth screen numbers etc.

7BA4 DROP
7BA6 2+
7BA8 :

Ie, a routine called R/W does the work:

7C66 LFA -->
7C68 NFA R/W
7C6C CFA 8334
7C6E PFA

7C6C DOCOL
7C6E B/BUF
7C70 SWAP
7C72 OBRANCH [7C7C]
7C76 **RDISK**
7C78 BRANCH [7C7E]
7C7C **WDISK**
7C7E DUP
7C80 ?ERROR
7C82 ;

This in turn calls either RDISK or WDISK, primitive routines which do the actual read/write disk<>cpu ram, leaving an error code in the process which is assessed by ?ERROR:

7198 LFA !CSP
719A NFA ?ERROR
71A2 CFA 8334
71A4 PFA

71A2 DOCOL
71A4 SWAP
71A6 OBRANCH [71AE]
71AA ERROR
71AC ;

?ERROR in turn branches to ERROR:

75B6 LFA (ABORT)
75B8 NFA ERROR
75BE CFA 8334
75C0 PFA

75BE DOCOL
75C0 WARNING
75C2 @
75C4 OK
75C6 OBRANCH [75D0]
75CA (ABORT)
75CC BRANCH [75F0]

75D0 ECOUNT
75D2 @
75D4 0=
75D6 OBRANCH [75F0]
75DA 1
75DC ECOUNT
75DE !
75E0 HERE
75E2 COUNT
75E4 TYPE
75E6 ." ? "
75EE MESSAGE
75F0 0
75F2 ECOUNT
75F4 !
75F6 SP!
75F8 IN
75FA @
75FC BLK
75FE @
7600 QUIT
7602 ;

which examines the value held in the user variable WARNING. When an I/O error occurs this routine presents you with the error message it grabs off screens 4 or 5. If you store the value of -1 into WARNING it will branch to the routine (ABORT):

75A6 LFA -FIND
75A8 NFA (ABORT)
75B0 CFA 8334
75B2 PFA

75B0 DOCOL
75B2 ABORT
75B4 ;

which left untouched will simply call ABORT and leave you with the QUIT message ie SUPER 4TH ok, or the equivalent in forth.

77C2 LFA QUIT
77C4 NFA ABORT
77CA CFA 8334
77CC PFA

77CA DOCOL
77CC SP!
77CE DEC
77D0 0
77D2 ECOUNT
77D4 !
77D6 CR
77D8 ." SUPER 4TH"
77E4 FORTH
77E6 DEF
77E8 QUIT
77EA ;

Try typing:

-1 WARNING !

Now open the drive door and try editing any screen and watch the resultant message. Afterwards restore normal handling with:

1 WARNING !

What we can do then, is to write a new error routine to say come up with a message telling the user he has a forth-screen i/o error and to press any key, after which we can re-enter our program say at the start of the segment the error occurred, without crashing into the immediate mode. Refer to screen 158. Type it in and try it, not forgetting to type FIX-(ABORT) afterwards. This routine still returns us to the immediate mode because it is not within the flow of a program.

Your probably sitting there wondering how we find our re-entry point. There are many ways to do this, all I do is I allocate a buffer at the start of the program for the number of name field address re-entry points I anticipate I will need such as on screen 160. Then, as I compile each segment I save the name field of that definition into the appropriate spot in the array. For example say we are in the editor part of our program and have just finished the final word to run it :

```
: EDIT 2 PFLAG ! (program flag)
  DO-EDIT (guts of defn)
  etc ;
```

Then added on the screen below this code, the following saves its re-entry point:

```
' EDIT RE-ENTRY 4 + !
```

As the program starts running, SET-ERROR places the value of -1 into WARNING, and the CFA of our new abort routine (MYABORT) into the parameter filed address of (ABORT) which previously would point to ABORT:

```
75A6 LFA -FIND
75A8 NFA (ABORT)
75B0 CFA 8334
75B2 PFA
```

```
75B0 DOCOL
75B2 (MYABORT)
75B4 ;
```

Once we enter the editor segment, the variable PFLAG standing for program flag is set to 2. Lets say we now encounter an forth screen access error, such as the drive without a disk etc, whilst within the EDIT routine. If you re-examine the decompilation of ERROR you will see that when the routine gets to WARNING it will jump to (ABORT) which now contains the new CFA of (MYABORT) which will execute just as we have designed it, and will display our error message " Forth i/o error - Press F9" above. When we reach the routine RE-START, it will take the value of PFLAG, here 2 at the time of the error, to jump into the RE-ENTRY array to find the re-entry address of EDIT.

Phew!!!! A bit heavy???? I hope I havn't confused you too much.

Thats all for Xmas. Perhaps in my 10 day break I may be able to do some computing!!! We are nearly up to the guts of our program - ie the codings for our design editor. Fortunately I've already done most of the work. Now for your homework - refer to Struggling Forth August 1988 - Analysing the Ti-editor, and an any-size editor and do some revision.

ADDRESS FOR CORRESPONDANCE:

```
R.TERRY
141 DUDLEY RD
WHITEBRIDGE
NSW 2290 AUSTRALIA.
```

049 436511

Screen

\ Str

DEC

: VSB

: VSB

: CLS

: ?TER

: PAGE

HEX

30 USE

36 USE

: CODE

CODE

DEC

Screen

\ Stru

Core

: ENDO

: OF 4

: CASE

: ENDC

Screen

\ My co

? all

: ?DIG

n1 n2

: ?INRA

stack

: VALID

Eg 10

\ A B C

: ?KEY-

Screen # 145

Screen # 146

\ Struggling forth Dec89

\ Struggling forth Dec/89 - superforth core words

```

DEC
: VSBW 0 SYSTEM : : VMBW 2 SYSTEM :
: VSBR 4 SYSTEM : : VMBR 6 SYSTEM :
: CLS 16 SYSTEM : : VFILL 20 SYSTEM :
: ?TERMINAL QTE : : ?KEY OK :
: PAGE CLS 0 0 AT : : FREE SP@ HERE - . :
HEX
30 USER SCRN WIDTH 32 USER SCRN START 34 USER SCRN END
36 USER ISR 38 USER ALTIN 3A USER ALTOUT 800 CON PDT
: CODE ?EXEC CREATE SMUDGE LATEST PFA DUP CFA :
  [COMPILE] FORTH :
CODE MON 0200 , 4E4F , 0201 , 2000 , CC40 ,
      02B1 , 4000 , 16FC , 0420 , 0000 ,
DEC

```

```

: IF ?COMP COMPILE OBRANCH HERE 0 , 2 : IMM \ 24 bytes
: THEN ?COMP 2 ?PAIRS HERE OVER - SWAP ! : IMM \ 28 bytes
: ELSE ?COMP 2 ?PAIRS COMPILE BRANCH HERE 0 , SWAP 2
  [COMPILE] THEN 2 : IMM \ 36 bytes
: BEGIN ?COMP HERE 1 : IMM \ 18 bytes
: BACK HERE - . : \ 18 bytes
: UNTIL ?COMP 1 ?PAIRS COMPILE OBRANCH BACK : IMM \ 24 bytes
: WHILE [COMPILE] IF 2+ : IMM ( 16)
: AGAIN ?COMP 1 ?PAIRS COMPILE BRANCH BACK : IMM ( 24)
: REPEAT ?COMP >R >R [COMPILE] AGAIN R) >R 2-
  [COMPILE] THEN : IMM ( 20)
: DO ?COMP COMPILE (DO) HERE 3 : IMM ( 20)
: LOOP ?COMP 3 ?PAIRS COMPILE (LOOP) BACK : IMM ( 24)
: +LOOP ?COMP 3 ?PAIRS COMPILE (+LOOP) BACK : IMM ( 24)

```

Screen # 147

Screen # 148

\ Struggling forth Dec89

\ Ti-forthorint core words

```

\ Core words - superforth
: ENDOF 5 ?PAIRS COMPILE BRANCH HERE 0 ,
  SWAP 2 [COMPILE] THEN 4 : IMM \ 36 bytes
: OF 4 ?PAIRS COMPILE (OF) HERE 0 , 5 : IMM \ 28 bytes
: CASE ?COMP CSP @ !CSP 4 : IMM \ 22 bytes
: ENDCASE ?COMP 4 ?PAIRS COMPILE DROP BEGIN SP@ CSP @ -
  WHILE 2 [COMPILE] THEN REPEAT CSP ! : IMM

```

```

: VSBW 0 SYSTEM : : VMBW 2 SYSTEM : : VWOR 26 SYSTEM : : VSBR 4
SYSTEM : : VMBR 6 SYSTEM : : VWTR 8 SYSTEM : : CLS 16 SYSTEM :
: VFILL 20 SYSTEM : : TRL -TRAILING : : DSRLNK 8 14 SYSTEM :
: AT GOTOXY : : DEC DECIMAL : : PAGE CLS 0 0 AT : : VAR VARIABLE
: : CON CONSTANT : : EBS EMPTY-BUFFERS : : 2DUP OVER OVER :
: MON 32 10 SYSTEM : : NDROP 0 DO DROP LOOP : : 2DROP 2 NDROP :
: BSAVE FLUSH BEGIN SWAP >R DUP 1+ SWAP OFFSET @ + BUFFER DFDAT
DUP B/BUF ERASE R OVER ! 2+ HERE OVER ! 2+ CURRENT @ OVER ! 2+
LATEST OVER ! 2+ CONTEXT @ OVER ! 2+ CONTEXT @ @ OVER ! 2+ VGC-
INK @ OVER ! 2 + 29801 OVER ! 10 + HERE R - R) DUP 1000 + >R SW
P >R SWAP R) 1000 MIN MOVE R SWAP HERE R) < UNTIL SWAP DROP
FLUSH : : L>U DUP 96 > IF 32 - THEN : : 3DROP 3 NDROP : : -VDU
BLOCK 0 960 VMBW : : 5DROP 5 NDROP : : PICK 2 ! SP@ + @ : : NDU
DUP 0 DO DUP >R PICK R) LOOP DROP : : 3DUP 3 NDU : : 2OVER 3F
6 + @ SP@ 5 + @ : : < > = 0 = : : P PICK :

```

Screen # 149

Screen # 150

\ My core routines

21Sept88

\ modified get\$

RHT22Sept88

```

\ ? all bytes at adr are digits exp adr -- leaves flag !=true
: ?DIGITS 1 SWAP COUNT 0 DO DUP 1 + @@ DUP 47 >
  SWAP 58 < MIN ROT MIN SWAP LOOP DROP ;
\ n1 n2 n3 is n2 in range exp no,ulimit,limit --flag !=true
: ?INRANGE 1+ SWAP 1- SWAP >R >R DUP R) > SWAP R) < MIN ;
\ stack Eg n3,n2,n1,3(of them) n4 flag true if n4 = n1,n2,n3
: VALIDATE 0 SWAP ROT 0 DO ROT OVER = ROT MAX SWAP LOOP SWAP ;
\ Eg 10 13 14 (funct keys) 3 (of them) 10 (funct key) VALIDATE
\ A B C = 65 66 67 3 ?KEYS-ALLOW,exec stopped till valid input
: ?KEY-ALLOW BEGIN DUP 1+ NDU ?KEY L>U VALIDATE IF 1
  ELSE DROP 0 THEN UNTIL >R 0 DO DROP LOOP R) ;

```

```

\ adj fixed length string at adr to order cnt. expect adr
: ADJNT COUNT TRL SWAP 1- C! ;
\ expects nil accepts to vdu only expects n
: VEXPECT 0 DO KEY DUP 13 = IF DROP LEAVE 0 ELSE DUP @ = IF
  DROP 1 0= IF 7 EMIT 0 ELSE -1 CURPOS +! R) 1- >R 0
  THEN ELSE DUP 10 = IF DROP 32 EMIT ELSE DUP 9 = IF
  DROP 1 CURPOS +! ELSE EMIT THEN THEN R) >R 2DUP >R >R
  SWAP - 1 = IF 0 -1 CURPOS +! ELSE 1 THEN THEN THEN
  +LOOP 1 CURPOS +! ;
\ Accepts fixed string via vdu vmbw read. expects adr..
: GETF$ CURPOS @ >R 2DUP R) OVER VEXPECT ROT ROT
  SWAP 2DUP C! 1+ SWAP VMBR
  1 + + 32 SWAP C! ;
: GET$ CURPOS @ >R 2DUP R) OVER VEXPECT ROT ROT SWAP 2DUP C!
  1+ SWAP VMBR OVER SWAP 1+ + 32 SWAP C! ADJNT ;

```


Screen
Ti-
:
: BOO
:
: FRO
ent
: FRO
ent
: TO-
Men
: FIL
tel
: WRD
:
: BU
Screen
Str
VAR
VAR
: RE-

Screen # 151

Screen # 152

```
\ My core routines      21Sept88
\ A B C = 65 66 67 3 ?KEYS-ALLOW,exec stopped till valid inout
: KEY-ALLOW CURPOS @ >R BEGIN DUP 1+ NDUP R CURPOS !
  PAD 1 GETF% PAD 1+ @ L>U VALIDATE IF DUP
  R CURPOS ! EMIT 1 ELSE DROP 0
  THEN UNTIL R> DROP >R 0 DO DROP LOOP R> ;
\ save screen contents to block,exp scr #,leaves nil
: ->BLOCK BLOCK UPDATE 0 SWAP 960 VMBR FLUSH ;
\ delay loop
: DELAY 0 DO NOP LOOP ; ( eg 1000 DELAY ) : 3DROP 2DROP DROP ;
\ inverse video to any line, expects line number
: INV-LINE 40 * DUP 40 + SWAP DO 128 I V XOR LOOP ;
\ pause till user hits any key
: ANY-KEY BEGIN ?KEY 0 > UNTIL ;
```

```
\ expects llim,ulim, adr with string, n, leaves flag
: @&VAL-NUMB OVER DUP ROT GET% ?DIGITS IF NUMBER DROP ROT ROT
  ?INRANGE ELSE 3DROP 0 THEN ;
0 VAR CURRENT 0 VAR FROM 0 VAR TO
\ adjusts val in CURRENT back within limits, exp nil,ly nil
: LAST CURRENT @ 1- DUP FROM @ < IF 1+ THEN CURRENT ;
\ adjusts val in CURRENT forward within limits, exp nil,ly nil
: NEXT CURRENT @ 1+ DUP TO @ > IF 1- THEN CURRENT ;
:5 these last two routines used to page back and forth
  between displays eg documentation for your program.
```

Screen # 153

Screen # 154

```
\ My core words
: +PAD PAD 40 + ;
\ exp llim,ulim,no,dig,col,row--n
: @DIGITS BEGIN 5 NDUP 3 NDUP AT 32 HCHAR +PAD SWAP @&VAL-NUMB
  UNTIL 5DROP +PAD NUMBER DROP ;
\ Eg 5 10 2 0 0 @DIGITS ,2digit no between 5->10 at col0,row0
\ halt program execution until either YNyn pressed
: Y/N 89 78 2 KEY-ALLOW ;
\ leaves 1 if Y or y key pressed, 0 if any other
: YES 89 = ;
\ blank n lines starting from row r exp n,r
: BLNK-LINES 40 * SWAP 40 * 32 V FILL ;
\ leaves disk lo, hi on the stack
: SCREENLIMITS DISK_LO @ DISK_HI @ ;
```

```
\ Magazine article dec 89
: ?F9/F5 15 14 2 ?KEY-ALLOW ;
: ?F6/F9/F5 12 14 15 3 ?KEY-ALLOW ;
: ?F7/N:6,8,9 12 14 15 6 4 ?KEY-ALLOW ;
: ?F4,F6,F5,F9 2 12 14 15 4 ?KEY-ALLOW ;
: ?E,X,F9 69 88 15 3 ?KEY-ALLOW ;
:5 various examples using ?KEY-ALLOW
```

Screen # 155

Screen # 156

```
\ Corrected EXPECT      RHT22Sept88
: EXP
  0 DO KEY DUP
  13 = IF DROP R> R> 2DUP >R >R \ drop code,index,counter
  SWAP - 1 = IF 1 ELSE 0 THEN LEAVE 0 \ l,0 ads address
  ELSE DUP \ kval,kval
  8 = IF DROP \ if a backspace,drop kevva
  1 0 = IF 7 EMIT 0 \ if at start beep user
  ELSE 8 EMIT \ if not do backspace
  R> 1- >R \ decrement loop counter
  1- 0 \ decr deposit adr,0 for +LOOP
  THEN
  ELSE DUP
  10 = IF DROP 32 THEN \ down arrow,replace with bin
  DUP EMIT OVER C! \ save byte to address
  R> R> 2DUP >R >R -->
```

```
\ Corrected EXPECT      RHT22Sept88
  SWAP - 1 = \ 1 = at end of accept area
  IF 0 -1 CURPOS +! \ decrement cursor position
  ELSE 1+ 1 \ not at end,inc adr,1 for +LF
  THEN \ not at end,inc adr,1 for +LF
  THEN \ then for second IF
  THEN \ then for 1st IF
  +LOOP \ inc loop by top of stack
  + 0 SWAP 2DUP C! 1+ C! \ add two nuls on the end
  1 CURPOS +! : ;5
note the problem with expect as subolled in both Ti-forth and
Super4th is that it allows inputting of the down arrow key which
causes a line feed and scrolls the input off the screen. It also
does not allow a positive (enter) press at the end of the input,
hence if user makes a mistake at the last character he cannot
fix it. This routine corrects both these faults.
```

```
\ Ti-forthprint graphics RHT9Aug89
\ * insert Ti-forth compatible disk, press F6 or F9"
: BOOTMSG 24 208 5 760 WSCRN 1 + FILL-WINDOW ;
\ * View from screen to screen "
: FROM/TO 40 840 1 240 WSCRN 2 + FILL-WINDOW ;
\ enter from filename etc
: FROM-FILE 17 325 5 320 WSCRN 1 + FILL-WINDOW ;
\ enter from filename etc
: TO-FILE 17 325 5 560 WSCRN 1 + FILL-WINDOW ;
\ Menu for file options
: FILES-MENU 16 201 10 337 WSCRN 1 + FILL-WINDOW ;
\ tell user that the original disk not in drive 1
: WRONG-DSK 40 840 1 520 WSCRN 4 + FILL-WINDOW ;
\ * Building file list ....."
: BUILD-PROMPT 40 840 1 0 WSCRN 4 + FILL-WINDOW ;
```

```
\ Struggling Forth Dec 1989
: EBS EMPTY-BUFFERS ; \ expects nothing
\ our new abort routine
: (MYABORT) 0 0 GOTOXY CLS \ clear the screen
0 10 GOTOXY \ display message here
." Forth i/o error - f9 to continue"
BEGIN ?KEY 15 = UNTIL ; \ only allow F9

\ detects inability to Edit,Load,Flush due to disk/device error
: SET-ERROR -1 WARNING ! (MYABORT) CFA ? (ABORT) ! ;

\ restore error messages to original system mode of handling
: FIX-(ABORT) ? ABORT CFA ? (ABORT) ! 1 WARNING ! EBS ;

;S to run load this screen type SET-ERROR, then try accessing a
disk with the door open. Restore with FIX-(ABORT)
```

```
\ Struggling forth Dec 1989
0 VAR PFLAG \ program flag
0 VAR RE-ENTRY 18 ALLOT \ stores 10 nfa's for re-entry

: RE-START SPI RPI \ clear stacks
EBS \ empty-buffers
PFLAG @ 2 # \ offset to add to RE-ENTRY
RE-ENTRY @ + \ address with re-entry nfa
CFA EXECUTE \ re-start program at point.
; \ expects nil, leaves nil
```

```
\ Ti-forthprint Error handling RHT9Aug89
\ prompt for forth screen access errors
: ERROR-PROMPT 0 22 AT 0 22 40 32 HCHAR
." Forth i/o error - f9 to continue"
22 INV-LINE ;
\ program dependent routine upon disk access error
: DISK-ERROR 0 ALTOUT !
ERROR-PROMPT ?FY EBS
RE-START ;
\ re-set error handling for forth screen access
: (MYABORT) DISK-ERROR ;
\ detects inability to Edit,Load,Flush due to disk/device error
: SET-ERROR -1 WARNING ! ? (MYABORT) CFA ? (ABORT) ! ;
\ restore error messages to original system mode of handling
: FIX-(ABORT) ? ABORT CFA ? (ABORT) ! 1 WARNING ! ;
```

;S example of a re-entry routine after forth i/o error

OF TRANSFORMERS

POWER SUPPLIES

AND THINGS

Tony McGovern

The discussion to follow on external power supplies is not strictly relevant to the usual self-contained TI-99 system, but might well be if you decide to hang external devices off your system, more floppies or perhaps hard disk drives which are starting to appear on some TI systems around Australia using the Myarf HFDC. It came about because Will's Amiga power supply was showing distress. The Amiga A500 power supply is a true successor to the old C64 supply in that it is on the edge of collapsing even with just the basic computer attached. Add another megabyte or two and external disk drives and it gets very dodgy indeed. So a new power supply was his birthday present, and his sister tied it up with a green ribbon for the occasion.

The logical choice by watts and amps for the buck has to be in power supply units for IBM PC clones. I did a little checking around and Richard Terry mentioned the name of a friend of his who was in the PC clone business. So I followed up that lead and ended up buying a 150 Watt XT power supply there. I can recommend this outfit, Penta Cad down in Cardiff, as good people to do business with. Competitive price, know what they are about, and very helpful with information. No doubt it helped that I knew just the questions I needed to ask and what to do with the answers, but a complete

NLP
NLP

which
also
put.
it

contrast to the norm in computer salesmen. Just try Computer Spot in Charlestown for that total contrast, and you will know what ex used car and hi-fi salesmen are flogging this year. Talking of such computer sales outfits, have you noticed that all of them in Newcastle seem to agree that 5 inch floppies are going to cost you \$10 a box this Xmas.

The typical XT supply already comes in a completely enclosed metal box with its own power switch and needs an IEC type mains power lead. A switched power outlet is usually used for mains power to the monitor in PC clones. Dick Smith carries both the lead and the special plugs for the extension power. What I did was remove the standard plug from a multi-way extension and replace it with this new plug. Now the monitor, modem, printer etc etc are all plugged in to this board, and the husky power switch on the XT supply is used as master control. That about completes the mains power side of things. These boxes are just installed as is in the main unit of PCs. One thing to watch for that a reputable supplier will have checked out is that the mains switch is 240 volt rated.

On the low voltage side the output comes on a number of flying leads with connectors attached. In normal PC installations the various low voltage leads are plugged onto the main circuit board or else into disk drives within the main enclosure. In our supply there were 4 disk drive power connectors and 2 with snap on connectors for the PC motherboard. It is easy to identify the ground return (usually black), 5V (usually red), and +12 volt (usually yellow) lines as these come out on the disk drive connectors. Other lines are not so easy to pick by inspection unless you have the detailed information. In the Amiga application only the -12V line is of interest also. You cannot however just turn it on and use your trusty Voltmeter to pick which is which, because unlike linear supplies these don't even work without a load. Further if you run them without a load you may do subtle damage internally, because of excessive voltage peaks on the primary side components.

This problem has to be taken care of anyway, because if you are going to use this box as an external supply it will no doubt become detached from its load at some stage. The best solution is to install a power resistor load permanently connected inside the power supply box. The minimum load quoted for this supply was 6 watts, and it didn't matter which output was loaded. The one to use is the 5V output as this has the most amps to spare. The supply in question will give 15A at 5V and the Amiga supply was rated at only 4A. A 3.9 or 4 ohm power resistor will do nicely on the +5V line. Its nominal power rating should be much higher than 6 or even 10 W so it doesn't run hot enough to do any local damage. The fan and air circulation will easily accomodate the extra 6W in a lightly loaded box. I mounted one on a bracket to a free stud inside the box. A better solution would be to use a chassis mounting power resistor and drill a couple of mounting holes in the box in a position which avoids fouling anything inside and is close to the main airstream. If you do this make sure no metal slivers fall on to the circuit board or are left floating around in the box.

Once the power resistor is mounted in place connect it to the 5V supply. The easiest way to do this is to take one of the disk drive connectors - usually you won't need all 4 of them outside the box - clip it off and solder the 5V and the ground return wires to the power resistor, and just tape up the other leads out of the way safely. How short you cut off all these wires depends on whether you will ever want to reconnect the lead outside the box. If a long length is left it clutters up the box inside. This avoids further disassembly of the power supply or soldering to the circuit board.

Now it is safe to turn on and you can double check the + and -12 on the mother board lines. There will be several wires in parallel for each of the 5v and ground leads. With everything turned off and unplugged again, clip off the mother board connectors, with the same compromise as before on length. Take a 6 hole length of plastic mains wiring barrier strip (terminal block), the kind where each wire can be brought into its own terminating hole, with a screw to hold it and make firm electrical

Connection to the wire coming in on the other side. Install this inside the box - it can float free and otherwise unsupported. Terminate all of the +5V wires into 2 of these, a couple more for all the earth returns, and one each for +12 and -12V lines. Tape up any unused lines safely out of the way (there may also be a -5V supply and a 5V supply ready line). The other side of this block is where you connect your external cable. Depending on the length of this cable you will probably want to use at least two 5V and two earth return lines up to where the power is needed to minimize voltage drops.

In the Amiga A500 application the plug at the computer end is hard to get hold of as Commodore refuse to sell it as a separate spare part, so I just cut the old one off fairly short. Before you make this drastic step check and carefully record what voltages appear on what pins. Repeat the exercise using barrier strip at this end, with doubled wiring in between on the heavy current lines. Only a short length of the Amiga cable should be left on the plug as it is very light gauge and the original A500 supply was run well above 5v to allow for voltage drops. The drive power lines will be used for external 5 inch drives which the original supply could never support. Do a final check that the right voltages are on the right pins before plugging into the computer.

In TI-99 applications where could this be useful? Mostly it would be for powering external disk drives, floppy or hard, or other experimental work, and it would be the disk drive connector lines that would be used for external drives. The main +5V, and +12 and -12V supplies are not of much relevance to the TI-99 PE-Box or its plug-in cards, as these use a higher voltage on the back-plane with separate local regulators on every card. Typical TI hardware intensive approach, but the TI-99 PE-Box is an industrial strength system that makes any IBM/clome type enclosure look like a delicate consumer item.

The problem with the PE-Box is that it comes with these intermediate supply rails running at very much higher voltages than the circuit diagrams indicate. I suspect this happened because it was designed about the time brown-outs were common in the USA and the TI designers decided to build in as much tolerance to mains fluctuations as the cards would stand in extra dissipation in their own regulators. This has caused a problem in later years in that some cards now available, particularly from Myarc, have much poorer thermal design than the originals. Switching supplies, at least if they are running well below full output, can be much more tolerant of mains variations and are more efficient, but are more complex and harder to fix. Commodore combined the worst of both worlds in the A500 supply.

Over and above this there is the problem in Newcastle and Australia in general, that while the PE-Box was designed for 110 or 220V AC mains supplies, the actual mains supply is at least a nominal 240V and in Kotara is over 250V much of the time. The transformer in the PE-Box seems to be adequate to handle both the higher voltage and 50 Hz rather than 60 Hz excitation but has no winding taps to allow compensation. This may be responsible for the excess leakage fields that have caused problems for AVPC and Geneve owners in 50 Hz countries who tried to sit their monitors on top of the PE-Box. This is something that had not been noticed very much if at all in USA on 60 Hz and the reason may very well be the rapid increase in leakage fields as transformer cores start to go into saturation when driven too hard. Our immediate concern is the high voltage on the PE-Box supply power rails and high dissipation in all the on-card regulators, which is not a good idea for long term reliability, let alone overheating problems in summer. The solution to this problem has been given in the Sydney TND, but we will talk about it in detail here anyway.

The fix is to use a transformer to reduce the AC supply voltage to the PE-Box. I have heard of people in the USA using a light dimmer but I would not recommend that way, as I suspect it would reduce the fan speed more than the internal heat generation and maybe even make things worse. The transformer does not need to be of full power rating if connected as an

autotransformer. Only the secondary winding has to be rated at the PE-Box current level, and a 1 Amp rating should be adequate for a TI system. I used a spare 240V to 30V transformer. My main system is 110V only and already uses a 240/120V transformer in a box (it came with the system when I originally bought Newtech's demo system after Orphan day) so I mounted this transformer in the same box. If starting with a 240V system you would have to provide your own electrically safe box. Ben Takach reported in the Sydney TND that the PE-Box would operate fine down to 185V. There is a trade-off here familiar to linear supply designers. The lower you make the supply voltage the less excess dissipation you have to put up with, but you have less headroom for mains voltage fluctuations before it all collapses. As it stands the TI PE-Box has an enormous margin, but I wouldn't want to push the supply down too far at the risk of losing the memory or corrupting disks when caught by a mains voltage drop.

How to make the connections? If you want to connect your console up to the reduced voltage also, use a small multiple extension strip with its plug removed for the output lead. Do not create a temptation to plug all sorts of things in, and install a fuse in the incoming active line if you do things this way, in order to protect the transformer. Firstly the green earth lead must be carried through the mounting box by a mechanically and electrically firm connection. Connect the transformer 240V primary across the brown (active) and blue (neutral) wires. Transformers to Australian standards will have a well insulated and separated primary winding. If your house wiring installation is correct there should be almost no voltage between the (blue) neutral and the (green) earth. After this is verified, connect the outgoing (blue) neutral wire to the incoming (blue) neutral. Connect one end of the transformer secondary (30V or thereabouts) winding to the end of the primary where the incoming (brown) active wire is connected. Do not connect the outgoing active lead yet. Now with the respect and care due to 240V AC, plug in the input lead to the mains, turn on the input voltage, and measure the voltage between the other end of the secondary and the common neutral. You have a 50-50 chance of getting it right first time with the secondary voltage subtracting from the primary voltage. If higher then turn off, unplug, and start again with the other end of the secondary. When you get it right, again turn off, unplug, and connect up the outgoing (brown) active lead to the free end of the secondary winding where you were measuring the voltage. Now make sure everything is mechanically and electrically sound, and stand back and admire your handiwork.

This seems to have worked very well for the PE-Box, but has brought a subsidiary problem. Those of you who had the original 110V TI external disk drive units will know that they always ran very hot. I suspect the original power supply in these was marginal even in the USA at 60 Hz. Ours have been gathering dust ever since we put the Chinons in the PE-Box, but recently I hauled one out to install a pair of second hand Panasonic 720K/80T 5 inch drives. On the reduced supply they run just fine though the transformer core still gets pretty hot. This is only true while the mains supply is at its normally very high level, but if it drops to even a more normal level the drives quit working on the reduced supply. If the transformer core stays only warm, there is not enough voltage to run regulators and drives. In other words given the real ratings of the transformer at 50 Hz, these PHP1850C units should never have been released in Australia without a revised power supply. So we are in a bind, and the solution may very well be an external drive supply. And I have had practice in doing that now! We have one of those Burroughs bank boxes but it has clapped out with a mysterious fault I have not been able to diagnose (mysterious to me anyway and I met my first low voltage switching power supply 20 years ago in the Electric Turkey). That Burroughs box was an utter waste of money and only ever had about one day of use. Seems to be the story of home computers - you end up with this trail of obsolete or junk equipment bought at vast expense not long before.

So there it is - power supply problems solved. Remember that any disasters are your own, and if you don't understand what it is about get someone who does to do the actual work. This particularly applies to the 240V side of things.

BEATING AROUND THE BUSH WITH Ron Klienschafer

TESTING AND DEBUGGING YOUR QUEST RAMDISK.

In the last twelve months since the Quest Ramdisk came into use there has been many reports of some of them not behaving as they should the main problem being a loss of the DSR in the DSR static Ram IC. Recent events with some hardware problems with the Clubs Quest (mysteriously posted to my address) and some interface problems with my own bucket of bolts has shed some light on a possible cause of some of these temprements, and out of the exercise came a cure for the DSR loss.

Up until now it was assumed that the cause of the DSR loss problem was from a spike in the power supply when the system is either switched on or off, this is not the real problem, the logic behind thinking this is that if power supply spikes were the problem then ALL of the Quests would give some trouble fairly often, but this is not the case. Since installing my own Quest it has performed flawlessly for nearly twelve months without corruption of the DSR or loss of data, but only if the system was powered down by switching off the console first then the PE box, but here a clue to the problem was noted, all would be OK ONLY if a pause was made between turning off the console and PE box.

We all know how stable the power supply system on the TI is, in my own situation I have had on occasions such a huge drop in power from the old generator that the TV used as a monitor would not even operate but when the power was returned to normal the old TI was still there with the cursor blinking merrily away still ready for work.

OK what is the cause?.

When the system is powered down, sometimes by some owners just turning off at the wall switch! Yea!, in which case the DSR will certainly go out to lunch, what happens is that the Quests power is dropped to the backup batteries lower voltage very quickly but the power supply in the PE box and console has large value electrolytic capacitors that are still discharging into the system, when this happens all the IC's in the console and PE box are doing some very fancy random switching and because the CE and WE control lines on the Quest are not kept high during this period the Quest's static Rams cop it, the Write Enable line cannot be held high enough by the batteries through the pull up resistor. Tests have shown that nearly ALL the Rams lose the first byte when the above happens, more so the 32K memory IC the 8K DSR ram and the first 32K IC that holds sector 0.

So how to test a Quest and what to do to fix the DSR problem?, we will start at worst case conditions where the Ramdisk wont work (assuming that it has been operating OK for some time), most of the common faults occur as follows but you must realise that other components can fail!, we will then go on to a method of curing the DSR loss problem.

TESTING:

The first thing to do is to remove (or disconnect) any other 32K memory expansion system, fit the Quest with a KNOWN GOOD 32K memory expansion IC and put the jumper on to enable the 32K memory expansion on the Quest, this is important to make the line drivers IC's on the Quest work flat out.

One weak link in our system is the line driver IC's, these bearing the numbers 74LS245 for the data lines and 74LS244 for the address lines, past experience with these IC's have shown that they can behave very strange indeed, they are prone to variations in operation with temperature and can be severely randomly temperamental, more so because the Quest is not in a case and getting plenty of cooling air. Just suppose you have an errant line driver IC that does unspeakable things on power up, this could cause havoc with the static Rams on the Quest, and these line drivers have been known to have opposite faults IE: they give instant trouble when cold and the reverse, they really stir the pot after warming up.

This type of fault first came to my notice when on one occasion I couldn't access my Disk drives until I had switched the console on and off several times, this turned out to be a faulty address line driver IC and the address contention was all out of whack, at first it was suspected that maybe the Disk drive heads needed cleaning but the culprit was that the CPU could not service the Disk controller card correctly, a new IC fixed the problem.

CHECKING THE QUEST.

If on power up the system locks up replace the 74LS245, if lock up still occurs start replacing one of the three 74LS244's ONE AT A TIME until power up can be achieved.

Install a QED module and load the QED files, Funnelweb, Quest Utility etc. (Version 4.4), when loaded do a 32K memory test from the Grom menu header, if a fault in the 32K is displayed replace another of the 74LS244's until the 32K is reported OK.

Now load and run the Quest utility program, do a Ramdisk memory test, this tests ALL of the memory on the Quest, if the DSR ram is shown as faulty AT THE FIRST PASS, before taking any other action replace any other 74LS244 and reload and retest, if a fault is still reported remove and reinsert the DSR ram IC (6264), although the sockets are of top quality bad connections can still occur, if still faulty replace the DSR Ram, sometimes it is best when testing the Quests Ram to disconnect the batteries and remove all data from the Quest, the test procedure works best on the first pass.

If during testing a fault is reported in one of the Quests 32K sector IC's swap it for another on the board and do a retest, if a fault is still reported at the SAME IC number then one of the 74LS244's is definitely faulty, this can still occur even though the 32K memory test shows OK, naturally if the IC number shown as faulty is the one where you swapped the first suspect then the 32K IC is most likely faulty.

When you are satisfied, load the DSR, format the Ramdisk and load some program files and give them a run.

A suggestion is that if any of the 74LS IC's are to be permanently replaced it may be a good idea to use the 74HC series, these being 74HC245 and 74HC244 respectively, although the HC series are slower than the bipolar LS series they wont cause any problems for old tortoise TI, the HC series may be a lot slower but they can sink more current and thus may give less trouble especially if the card is removed from the PE box before the recommended two minute waiting period for such action. So if you are experiencing problems the HC series may be worth considering.

I think that I said in an earlier article that the 5 volt regulator heat sink was a little inadequate, these regulators can be real beasts sometimes, if there is a fault in one it will become apparent after a period of time when it warms up and the fault is easily noted in that odd characters will be displayed when a Dis/Var 80 text file is loaded from the Ramdisk and you know those odd chars should not be there, a solution to that problem is to fit a larger heatsink, I made one up out of scrap aluminium that bolts under the regulator and curves up past the batteries and has a large heat dispersal area.

CURING THE DSR LOSS.

First attempts to cure the problem of losing the DSR was to try to hold the write enable line (WE) on the Quest high during power down, this met with some success but proved to be a nightmare of transistors, resistors and other bits and pieces, a real problem for an easy fix!, and beyond the scope of explaining in this newsletter.

A simple solution found is to fit a large VALUE electrolytic capacitor to the Quests power rails AFTER the voltage regulator and VCC control diode. Depending on the number of Ram IC's that are on the board sets the criteria for the value required for the capacitor. With only four 32K IC's plus the 6264 DSR Ram a value of 2200 uf seems to be adequate but with a full board of 16 Rams plus 32K and the 6264 a value of at least or greater than 4000 uf will be needed, so a suggestion is to use a 10,000 uf, 10 volt electrolytic, on any board.

What happens after this modification is that at the time of power down all the Quest main control lines, CE, WE etc are held high long enough until the console and PE box IC's have finished their funny business, the large capacitor then slowly discharges down to a low voltage ready for use the next time.

Fitting such a capacitor may prove to be a little difficult. If there are not too many cards in the PE box it would be possible to stick the capacitor on the back of the card with some double sided tape then run a couple of wires to the relevant tracks. If space is a premium in the PE box then a suggestion is to fit a MOLEX 2 pin socket to the board, attach a couple of lengths of wire to the Molex plug and run them out of the rear of the PE box and attach them to the capacitor which could be mounted to the rear of the box with a suitable clamp. INSULATE all joints or exposed wire.

On the Quest is two spare holes on tracks on the front centre COMPONENT side of the board, these are between two little blue monolith capacitors and the yellow LED (light), the OUTSIDE hole must be connected to the NEGATIVE terminal of the capacitor and the hole next to it (INSIDE TRACK) must be connected to the POSITIVE terminal of the capacitor.

Tests after this modification has proved most successful, with everything powered up and a program on screen the wall switch has been turned off and on many times to test the modification and resulted without any loss of Ramdisk data.

There are a great many places that the capacitor can be connected but to describe where in this article would become very confusing, anyone in doubt as to what to do SEEK HELP, don't try it if you are not sure.

Dick smith sells a 10,000 uf 10 volt Electrolytic capacitor (Cat. No. R-4480 approx \$4.50) and a two pin Molex plug and socket (Cat. No. P-5103, approx \$1.95).

Now may all your Quest problems be gone and a Merry Xmas to everyone.

Ron Kleinschafer.

MORE MOANING!!!

You might have noticed that I have been using more and more full page formatt in stead of the double columns. Well there is a reason! My employer who is wise and mighty and also hands out the money each fortnight, has decided that I have had enough of shift work, I am now on normal 5 day dayshift for at least the next 10 months. So! I have less time to get the newsletter to gether each month. Using full page instead of double column dramatically cut the time need to physically put the newsletter together. So if you are intending contibuting this year then please use the full page formatt from now on.

Joe W.

BROKEN HILL CALLING Bev Warren

After many promises to write an article, I have finally put pen to paper. Yes, I am handwriting this. I know I should be using old faithful, but if truth be known, the kids are using it so if I want to get an early night, you hand write your urgent correspondence.

As I am not a wizz on computer pros and cons, like some of our great members, I have decided to list all the reasons why our city cousins should come to the "Country".

Broken Hill in Western N.S.W., or is that "out back" N.S.W., is a really great place. We have ideal weather except mid summer (40-45) and lots to see.

In town you can feast your eyes on all sorts of art, in approximately 17 art galleries. You can go down the mine and see what made the Hill.

You have museums, wonderful buildings, one whole block is under the heritage council! You have the Sundown Trail, which at present has the Sturt Desert Pea out in full bloom.

That brings me to the wildflowers. They are abundant at present, but even in dry years still plenty to see.

Not to be forgotten is the Flying Doctor base, and of course the school of the air, where my children, Vanessa and Megan did their primary education by HF radio.

Approximately 100km south is the Menindee Lake system, seven times larger than Sydney harbour, and Kinchega National Park where the old Kinchega Woolshed still stands and the ruins of the paddlesteamer "Providence" adorn the banks of the Darling River.

Of course fishermen are not forgotten, many good catches are to be caught to put on that fire in the morning as you watch the wildlife along the river's edge.

There are campsites along the riverbank. Copi Hollow is where those who wish to ski on the lakes head and of course for our sailing folk there is an area for you.

Menindee township has a modern motel and of course the famous "Maiden's Hotel", also the sight where Burke and Wills' camel drivers are buried, to mention just a few.

Tandou, our own dairy, runs the biggest irrigation system in the southern hemisphere, and of course the orchards along the river.

Silverton is close by. A historic township in its own right and also the sight where several films have been made, Breaker Morant, A Town Like Alice, Mad Max to mention just a few. Also can be found Stephens Creek dam and a lovely park which was built by the miners for recreational purposes.

Mootwingee is approx. 130km north, and has Aboriginal paintings, etchings, spectacular gorges and rockholes and a lot of wildlife. It is also a National Park and well worth a visit. Bring your walking shoes as there is lots to see. Also "Wright's Cave" where Wright's initials are there to remind us of years past.

On to White Cliffs, our very own opal country, quite different from Lightning Ridge. There is the Solar Power station and plenty of dugouts, lots of opals to see or buy but best of all there is the "Dugout Motel". Only opened this year, it is a great experience. The rooms are dugout of the hills and you will really be staying underground. It has a swimming pool and wonderful meals and the hospitality is great. Leon and Marge Hornby will make your stay one to remember. I stayed last weekend and would recommend anyone to include a stopover at the "Dugout". *

hoping this has wet your appetite for an out back experience, so come on all you Aussies and indeed our overseas members, we would love to see you over this way to show you the other side of life.

A really great place for a TI meeting.

Ben Warren.

* Broken Hill is the gateway to the Inland. You can go to the centre and see Alice Springs and Ayers Rock, or further north to Darwin.

Or west to the Flinders Ranges or Adelaide. Perhaps south to Mildura and Melbourne, are east to Sydney or Brisbane. A shorter trip perhaps is Tibooburra, Sturt National Park and Corner Country, approx. 5 hours by road

D I Y SEASONS GREETINGS

by a Basic Battler

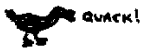
Well, the marketing hype is underway for the retail rip-off season, which is still referred to as Christmas - probably for sentimental reasons. And so it came about that, although I think I've finally managed to kick the habit of writing articles for the newsletter, I decided that some form of well-wishing wind-up to the year would not be untoward.

The initial thought was - something along the lines of a brief article, perhaps with a small program (why not?) containing an appropriate and meaningful MSG\$, which, as you are no doubt aware, is computerese for message.

Now let's see - the MSG\$ should be no great problem - intake of a suitable quantity of inspiration should cover that angle: and I had a subby around somewhere that would 'display' it on the screen without chopping it into chunks and CALLing HCHARs. All that was needed was some sort of background for the screen.

So I hexadecimalled a CHARACTER on a bit of graph paper and came up with:

```
10 CALL CLEAR
20 G$="B166661818666681"
30 CALL COLOR(1,10,16)
40 CALL CHAR(32,G$)
50 CALL CLEAR
60 M$="something meaningful
70 etc etc
```

About the only good feature with this effort was that it would not over-excite anyone seeing it. 'sides which it was a tad too mundane for this 

I was still pondering the problem during the inspiration process when I happened to recall a triangular shape I'd seen some years ago - one of those 'designs' in two dimensions that cannot exist in three. Probably the most common is the 2/or is it 3 chimneys type of thing.

Managing to contain myself from a Eureka or three, I thought - such a background design would not only be somewhat different. I could do with a good CHARing workout.

I should have realised at the time that once you stray from the straight HCHAR and narrow VCHAR paths of TI Basic, life wasn't meant to be easy. Looks like I'll have to change my brand of inspiration if it produces side effects like this.

Some (considerable) time later I had the triangular shape on screen - and it took only 70 program lines. So I was in somewhat of a quandary. Who'd be bothered tapping in 70 program lines for a static screen display? Rather than just bin it, I decided to include it even though it has no particular merit or relevance to my original idea. An example of character building - or just a typing exercise for a rainy day? Here 'tis

```
10 CALL CLEAR
20 A$="000101030307070F"
30 B$="0F1F1F3F3F7F7FFF"
40 C$="F0E0E0C0C08"
50 D$="FFFEFECFCF8F8F"
60 P$="00000C0C0E0E0F"
70 Q$="F0F8F8FCFCFEFEFF"
80 R$="0F070703030101"
90 S$="FF7F7F3F3F1F1F0F"
100 X$="FFFFFFFFFFFFFFF"
110 CALL COLOR(9,12,1)
120 CALL COLOR(10,7,5)
130 CALL COLOR(11,7,1)
140 CALL COLOR(12,5,1)
150 CALL COLOR(13,7,12)
160 CALL COLOR(14,5,12)
170 CALL CHAR(97,A$)
180 CALL CHAR(98,B$)
190 CALL CHAR(99,X$)
200 CALL CHAR(104,C$)
210 CALL CHAR(105,D$)
220 CALL CHAR(106,S$)
230 CALL CHAR(107,R$)
240 CALL CHAR(112,P$)
250 CALL CHAR(113,Q$)
260 CALL CHAR(114,R$)
270 CALL CHAR(115,S$)
280 CALL CHAR(116,C$)
290 CALL CHAR(117,D$)
300 CALL CHAR(118,X$)
310 CALL CHAR(120,X$)
320 CALL CHAR(121,R$)
330 CALL CHAR(122,S$)
340 CALL CHAR(128,A$)
350 CALL CHAR(129,B$)
360 CALL CHAR(136,R$)
370 CALL CHAR(137,S$)
```

```

380 A1$=CHR$(97)&CHR$(99)
390 B1$=CHR$(98)&CHR$(99)
400 P1$=CHR$(118)&CHR$(112)
410 Q1$=CHR$(118)&CHR$(113)
420 R1$=CHR$(121)&CHR$(120)
430 S1$=CHR$(122)&CHR$(120)
440 SP$=CHR$(106)&P1$
450 RQ$=CHR$(107)&Q1$
460 A3$=A1$&CHR$(128)&CHR$(118)
470 B3$=B1$&CHR$(129)&CHR$(118)
480 A4$=A3$&CHR$(117)
490 B4$=B3$&CHR$(116)
500 S3$=S1$&SP$
510 R3$=R1$&RQ$
520 Z$=CHR$(32)&CHR$(32)
530 FOR K=2 TO 10 STEP 2
540 K$=K$&Z$
550 Y$(K)=K$
560 NEXT K
570 FOR L=1 TO 15
580 X15$=X15$&CHR$(99)
590 NEXT L
600 FOR L=1 TO 18
610 X18$=X18$&CHR$(120)
620 NEXT L
630 PRINT TAB(15);A1$&CHR$(1
28)&CHR$(112);TAB(15);B1$&CH
R$(129)&CHR$(113);TAB(14);A3
$&P1$;TAB(14);B3$&Q1$
640 PRINT TAB(13);A3$&CHR$(1
05)&SP$;TAB(13);B3$&CHR$(104
)&RQ$;TAB(12);A4$&S1$&SP$;TA
B(12);B4$&R1$&RQ$
650 PRINT TAB(11);A4$&Y$(2)&
S3$;TAB(11);B4$&Y$(2)&R3$;TA
B(10);A4$&Y$(4)&S3$;TAB(10);
B4$&Y$(4)&R3$
660 PRINT TAB(9);A4$&Y$(6)&S
3$;TAB(9);B4$&Y$(6)&R3$;TAB(
8);A4$&Y$(8)&S3$;TAB(8);B4$&
Y$(8)&R3$
670 PRINT TAB(7);A4$&Y$(10)&
S3$;TAB(7);B4$&Y$(10)&R3$;TA
B(6);A1$&X15$&CHR$(137)&CHR$(
120)&SP$
680 PRINT TAB(6);B1$&X15$&CH
R$(136)&CHR$(120)&RQ$;TAB(6)
;S1$&X18$&CHR$(106)&CHR$(117
)
690 PRINT TAB(6);R1$&X18$&CH
R$(107)&CHR$(116);
700 CALL KEY(Ø,R,S)
710 IF R=-1 THEN 700

```

Bit of a doozy, eh? When I first tapped it in for a trial RUN, it had a glitch in the apex area (which can be quite painful!). So I unknotted a couple of strings to allow for a spot of experimentation and gave it another whirl - no problems! - the line was 640. Decided to put it all back as it was because my head was hurting after so much concatenating and deliberating about alternative brands of inspiration.

The CALL KEY is merely to hold it for an indefinite time - in case there's an OOH or AH forthcoming - just don't peer too closely at some of the colour contrasts along the edges. FCTN 4 out of it.

Being a non-typist I found the PRINT lines rough going on 4A's keyboard which is anything but user friendly with so much SHIFT and FCTN keywork involved.

I suppose it's understandable in a way that computer keyboards initially followed the general lines of teleprinter/typewriter keyboards and existing symbols were adapted for specific computing requirements.

But I digress. And, with all that waffling, I've blown the brief article bit and my original idea is well and truly down the gurgler. And I'm left with the subby I was going to use to 'display' the MSG\$.

Even though I've had it around for quite a while now (about the time I was weaned from URG and BB), it has not seen much use as I don't suffer from meaningful MSG\$ syndrome. It came about when I found that

```

WHEN YOU HAVE TO PRINT LENGT
HY TEXT TO SCREEN WITH 28 CO
LUMN BASIC, SPACES SEEM TO A
VOID THE 28TH & 29TH COLUMNS

```

Such dismembered words are difficult to read, especially when you have to move your lips at the same time.

Of course there's the option of slicing the MSG\$ into <=28 character chunks. And one pernickety programmer I know of has even gone to the extent of altering words and rephrasing a MSG\$ to assist in this process.

Even if you leave text as is, it can be a tedious exercise counting the <=28 char. chunks. I used to use the standard rule for numbers up to 28 - fingers plus toes (socks off) plus the other (visible) bits-that-there-are-two-of ie arms, legs, eyes and ears. Which sorta explains how the subby came about. It could do with a wash and polish (be my guest). However I'm fairly sure that there's a better version around that does much the same thing.* (probably more than one). Anyway - here 'tis


```

10 CALL CLEAR
20 M$=""
30 LN=LEN(M$)
40 IF LN<=28 THEN 410
50 M1$=SEG$(M$,1,28)
60 IF SEG$(M$,29,1)=CHR$(32)
THEN 110
70 P$=M1$
80 GOSUB 500
90 U=T-1
100 GOTO 120
110 U=28
120 M1$=SEG$(M$,1,U)
130 M2$=SEG$(M$,U+2,28)
140 IF SEG$(M$,U+30,1)=CHR$
(32) THEN 190
150 P$=M2$
160 GOSUB 500
170 V=T-1
180 GOTO 200
190 V=28
200 M2$=SEG$(M$,U+2,V)
210 IF LN>U+V+2 THEN 220
ELSE 420
220 M3$=SEG$(M$,U+V+3,28)
230 IF SEG$(M$,U+V+31,1)=
CHR$(32) THEN 280
240 P$=M3$
250 GOSUB 500
260 W=T-1
270 GOTO 290
280 W=28
290 M3$=SEG$(M$,U+V+3,W)
300 IF LN>U+V+W+3 THEN 310
ELSE 420
310 LS=LN-U-V-W-3
320 IF LS<=28 THEN 330
ELSE 350
330 M4$=SEG$(M$,U+V+W+4,LS)
340 GOTO 420
350 M4$=SEG$(M$,U+V+W+4,28)
360 P$=M4$
370 GOSUB 500
380 X=T-1
390 M4$=SEG$(M$,U+V+W+4,X)
400 M5$=SEG$(M$,U+V+W+X+5,
LS-1-X)
410 M$=M1$
420 PRINT TAB(INT((29-LEN(
M1$))/2));M1$:
430 PRINT TAB(INT((29-LEN(
M2$))/2));M2$:
440 PRINT TAB(INT((29-LEN(
M3$))/2));M3$:
450 PRINT TAB(INT((29-LEN(
M4$))/2));M4$:
460 PRINT TAB(INT((29-LEN(
M5$))/2));M5$:
470 STOP
500 SP=POS(P$," ",SP+1)
510 IF SP=0 THEN 540
520 T=SP
530 IF SP<=28 THEN 500
540 RETURN

```

As I dusted it off with the original intention of giving it a light (MSG\$) snack, I realised that it catered only for an up-to-4 row Basic line MSG\$ and that it could be improved (that's for sure) and expanded. It would merely be a matter of developing a suitable algorithm* (I'm not exactly sure what it means - but I LIKE that word!) that would provide for user-selected number of rows and chars. per row.

If I ever get around to trying the algorithmic method (now there's an interesting thought), I might just give it a go. It will probably depend on how efficacious the new brand of inspiration is.

At present, however, I still use the diagonal thinking approach to programming problems. I remember that, when the idea for this subby first surfaced, I wondered how to go about detecting the last space before the 28th/29th char. I recalled the programette in URG which backwards a string, so I took a 28-char SEG\$ of the MSG\$, reversed it and POSed for the first space, which therefore located it at 29-POS in the original.

Sound complicated? - well it surely was. There was a bit more to it than that, of course. Strangely enough it worked. The main problem was that it was rather - well, complex is putting it mildly, and, as far as program flow was concerned, the result was barely a trickle. I don't know whether this version is any better, but there you go.

As mentioned, I've used it only once or twice, so I guess (hope) that the mathematics are OK. For this airing I decided to check it on a couple of sample meaningless MSG\$s and they came out OK. There's a lots of smallish words and CHR\$(32) version

```

20 M$="NOW IS THE TIME FOR A
LL GOOD MEN TO COME TO THE A
ID OF THE PARTY AND THE QUIC
K BROWN FOX JUMPS OVER THEM"

```

and a big word version

```

20 M$="COMMUNICATIONS AND MI
CROPROCESSOR AND EXPONENTIAT
ION AND MICROCOMPUTERS AND A
LPHANUMERICAL AND COMPUTERS"

```

Oh - a final thought - the quotes in line 500 enclose a space (of course you knew that). Doing a trial run I typed it as a null string.

Now let's see - something still seems to be missing. Of course - the meaningful MSG\$. Well, if you think I'm going tie all these bits together at this stage - WRONG! If you feel so inclined you can generate your own personal MSG\$, use the subby to display it on the screen against either of the backgrounds or, better still, give it all a miss and write a meaningful article for the newsletter instead. What else could you expect from a DIY article anyhow?

So, to round off the article, here is the MSG\$ bit - the result of some new inspiration. As it now doesn't have to go through the subby, the printed page version has been slightly expanded.

I'd just like to wish a merry Christmas and a merry New Year and happy computing to all 4A foster parents, including Fosters fanciers and imbibers of better brews and other fine fermentations. And, of course, Genevers too! (No, you don't detect a trace of envy there)** And let's hope that our 99** chipped computers - and our own fish&chipped selves - don't become endangered species in the 90s.

* In my notebook of odds & sods there's a subby for automatic word wrap a la Pittsburgh UG which I'm fairly sure came to me via the HV99 newsletter. Even when I added a missing closing parenthesis to the original I couldn't get it to work.

It's far more organised and concise than my effort. I was going to try to analyse it to see if I could get it firing for this article but my head still hurts (I wouldn't know what I was doing anyway!). If someone out there knows the correct version I, and anyone else with a counting to 28 probles would appreciate an article about it. The editor could use it to fill one of the NOTHING spots!

** I hope Santy got my letter in time!

FAR OUT

with

DICK SCHAYDEL

After Ron Kleinschafer sent me some copies of the HV99ers newsletter to read he then suggested that I attempt to write a bit, What me?, for a computer newsletter, I was very surpriced. The more that I considered the idea, the more it appealed to me. After all, there isn't much to do here in the bush after dark. Before I start something I may regret, let me introduce myself.

My name, of course is Dick (Richard) Schaydel. I'm over 40 and a confirmed bachelor. I have a small farm about halfway between Weilmoringle and Brenda on the Culgoa River (about 170 Km NW of Lightning Ridge). The river supplies water for a self-powered irrigation system powered by a hydro-ram. The farm supplies some irrigated crops (milo) other grain and a bit of meat. Right now, I've a pig breeding program going but my first love is Opal mining. That's where Ron and I first met, I was working a small claim in the Grawin Opal field and met Ron one day quite by accident in Lightning Ridge. The German extraction of our names gave us something in common besides digging around like a couple of flamin' wombats. We've been corresponding (thats like writing mate!) ever since. Ron always got the better of it --- he got the colour and I usually ended up with only potch. I did manage to create one of the biggest mullock heaps in New South Wales! I still can't resist the allure of "black fire" and make a couple of trips to the Grawin field each year. The fact is that the trips get spaced farther apart each year and the nobbies are fewer and fewer.

I got my first TI out of curiosity about what the flamin' things did, it (the TI ended up helping to keep up with my farm expenses but I have since moved on to one of those "other brands" to record my pig breeding program. The TI will always remain my favorite and I still use it as much as I can. So, when Ron suggested that I write an article for the Hunter Valley newsletter, I reluctantly agreed, so lets give it a go!.

I thought I would write about "vintage" programs until I realised that just about everything I have could be considered "vintage". I've never been one to sit for hours in front of the monitor shooting aliens of various shapes but there are some games that are enjoyable. One of the better XB games is one call AARDVARK. An Aardvark is a creature with an emormously long tongue (I know some people built like that!) with an insatiable appetite for ants. That is the premise of this one! You are the ant scurrying about in an ant-hill while this here Aardvark thing lashes out with his tongue at you. Try to move the sugar cubes at your own peril. Variations of this program have appeared from time to time but it is still worth a bit of relaxation. The game itself is a bit simple but the grahics are good and the concept is beaut. METEOR RESCUE is another program worth looking at again. The object of this one is to land your landing craft and rescue space explorere one at a time. Easy enough! The problem is that you have to dodge meteors on both your descent and ascent and still make a soft landing in each place. The graphics are well-done(the men running is an excellent touch) and METEOR RESCUE is a bit more challenging than Aardvark. Besides, rather than destruction, you are trying to accomplish something constructive.

If you write your own XB programs, there are a couple of utilities that can be of great help. NEATLIST will help you keep track of all your variables and constants and CRUNCH will do much the same as well as compact a program to a portion of it's original size. The former was authored by Danny Micheal and the latter by Tim MacEachern. To be

sure, there are others that will do the job as well but might not have the same combination of options.

The sense of this article is to get a few who may read it to dig into that great mullock heap of a disk library that Ron tells me you have and take a second look at what has been accumulated there. You might come to realise that some of the "potch" you have stored has a "flash of fire" after all. Whew! This article wasn't that hard! We got acquainted and plucked a couple of programs out of the dust bin. Time for a cuppa and then out to the pig pen for a look at that lot.

MODEM MADNESS. *****

Peter Smith our ever alert President has a couple of spare modems, One of these is a Scubert PE Box modem from TISHUG. Peter, in a fit of madness decided to loan this one to me so he could send his Presidents message at the last minute each month for the newsletter.

This has been an interesting exersize in the theory of deminishing circles for me. Never had much interest in modems, I always seemed to find that writing letters and sending discs through the mail worked the best. It also seemed infinitely more reliable to me.

Once in three months has the gadget worked for me. On the other occasions Peter has given up after an hour of total frustration, firstly with my "not up to date" software, secondly with me having to read the docs with every move, and thirdly with me not being "all that interested" in his modems anyway.

My favourite trick is to manage to leave the modem connected in such a way that I can't receive any incomming calls. This makes Peter's words "hang up and I'll call you back" completely useless. I don't do it on purpose, I just make an honest mistake. The exersize ends with me picking up a disc from one of Peter's friends the next day.

So to the midnight modem set I can say with some degree of knowledge, YOUR ALL donkers the lot of you.

Joe W.

MORE FROM

BRIAN RUTHERFORD

Last month I published the little one liner below, that when added to the beginning of a program and RUN made the program unlistable.

```
I CALL PEEK(-31952,I,J,K,L)::  
I=I*256+J-65536::  
K=K*256+L-65536::  
FOR A=I TO K STEP 4::  
CALL PEEK(A+2,J,L)::  
CALL LOAD(J*256+L-65537,0)::  
NEXT A::STOP
```

Here is the quick explanation. You peek the start and finish addresses of the line number table, and use that to step through and replace the length byte for each line of code with >00.

If you understand that read no further, but if you are having trouble with the concept I will try to explain what is happening statement by statement.

```
CALL PEEK(-31952,I,J,K,L)
```

This is reading the values stored at four bytes starting at address -31952=>8330. So we are really looking at what is stored at >8330, >8331, >8332, >8333. Now the two byte value stored at >8330 and >8331 points to where the line number table starts, and the two byte value stored at >8332 and >8333 points to the end of the line number table.

```
I=I*256+J-65536.
```

All that is doing is converting the two values that were found at >8330 and >8331 to a decimal number that the XB loader can understand. Of course that number is the address the line number table starts at.

```
K=K*256+L-65536
```

Again converts the two byte values found at >8332 and >8333 to a decimal form, and that is the address of the end of the line number table.

A little aside here. Why TI did not leave all peeking and poking in hex I don't know, I think it would make life a lot easier if it was.

So now we have two variables I and K which point to the start and end of the line number table. A point to remember is that each entry in the line number table consists of

FOUR BYTES. The first two are the line numbers and the LAST TWO BYTES POINT TO THE STARTING ADDRESS OF THAT LINE OF CODE.

```
FOR A=I TO K STEP 4
```

Is setting up a loop to enable you to step through the line number table entry by entry.

```
CALL PEEK(A+2,J,L)
```

Here we are using the loop counter (the address of each entry in the line number table.) +2 to read the pointer address to the start of that line of code.

Before we go any further I had better explain about how the program lines are stored.

The FIRST BYTE is the length byte ie. it is the number of bytes long the lines is, NOT including the length byte itself but including the last byte marker.

The next byte is the start of the rest of that line of code in token format. THAT IS THE ADDRESS POINTED TO FROM THE LINE NUMBER TABLE.

Lastly comes the end of line marker >00

```
CALL LOAD(C*256+D-65537,0)
```

Lets look at the C*256+D-65537 part first. If that had been C*256+D-65536 it would have been the address pointed to from the line number table and the start of that line of code, but by subtracting 65537 we are actually stepping back one byte, (WHICH IS THE ADDRESS OF THE LENGTH BYTE) and are replacing the length byte with 0.

```
NEXT A:: STOP
```

Finish the loop and stop, to save the program running on and allowing you to delete that line etc.

So there you have it, when the XB editor tries to list the line of code it gets its self in a real pickle because the length byte has been changed to zero which is the same as the end of line marker. This does not worry the computer when it runs the program as it does not need to know how long the line is, only where the line of code starts.

random bytes

with
BOB CARMANY

"We wish you a Merry Christmas, we wish you a Merry Christmas, we wish you a Merry Christmas --and a Happy New Year!" Season's greetings from the Northern Hemisphere!

This month, let's look at some TI-Writer stuff. Actually, these tips will work equally well with Funnelweb. I'm sure that almost everyone knows by now that you don't have to go through the entire procedure outlined in the TI-Writer documentation to access the various functions. For example, you can bypass (F)iles and simply type in "LF" for (L)oad (F)ile or (S)ave (F)ile to directly access those functions. One thing that you may not know is that you can eliminate the need to use the arrow keys to go to the actual filename. The editor doesn't care if there are a couple of spaces after the "LF" or "SF" commands. So, if you type in:

```
LF <SPACE> <SPACE>
```

you will find that the cursor is positioned over the first character of the actual filename when it appears on the screen --you no longer have to tab over with the arrow keys before you can change the filename.

I found that when I worked with the original TI-Writer character set, it became a real chore sometimes to find the CR's, LF's, and other control characters. This was particularly annoying when it came time to edit a document. Besides, the characters themselves weren't the most pleasing to the eye.

The solution was provided by Lutz Winkler. He designed a character set that used inverse video for the control characters which made them stand out from the text as well. In addition, the 9-sector set was shortened to 5 sectors and the characters themselves are a bit more pleasing to the eye. By the time that you read this, the inverse video character set will be in the UG library. Just copy it as CHARA1 in place of the original and you are ready to go.

The next solution!! Let's say that one of the kids has a report to do for school. He has been assigned to write a 2,000 word essay on the virtues of behaving in class (must be someone else's son!). Anyway, how do you know when the limit is reached? Well, you could type the essay in with Funnelweb and then go back with a pencil and paper and tally up the number of words -- and hope that you didn't make a mistake in the count. Another solution is to use the latest version of WORDCOUNT. This little program will count the number of words for you a lot quicker than doing it by hand.

The program itself appeared in MICROpendium awhile back but it is certainly worth a second look. Like the CHARA1 file, by the time that you read this it will be in the UG library.

Now, for a brief digression before I close this month's column. A truly wondrous machine the TI-99/4A!! It has capabilities that stagger the imagination. For me, it has been a "window to the world" --- compressing 12,000 miles into almost nothing. Periodically, I open up a well-worn packet which contains letters that have traveled those 12,000 miles (that's right guys, I've saved ALL of the letters). Sometimes I'll read through the packet from Larry Reid and reflect on the kinship that has formed over the years. Then, it's Ron K and his outrageous humor or maybe the technical treatises that Tony McGovern sends from time to time. I guess the "bottom line" of all this is that in the time of year for Peace on Earth and Goodwill toward all Men, I feel very fortunate to have been introduced to such a fine bunch of true "mates" by an orphaned computer.

"Merry Christmas to all, and to all a good night!!"



DISK REVIEW BEYOND FUNNELWEB

XX

by Al Lawrence

Funellweb 4.20 never made it beyond the Beta cocoon owing to MURPHY and his laws.

From Tony and Will McGovern down in the picturesque valley now known all over the world as Funellweb Farm. Comes :-

FUNELLWEB 4.21

This disk is now available from the HV99'ers Software Librarian.

Where to start ? The Web has spread beyond the original 2 Single Sided, Single Density disks. And is now issued on ONE Double Sided, Single Density Archived disk complete with Archiver 3.03 and YOU are REMINDED to respect the FAIRWARE principle to author Barry Boone as the latest version is supplied.

Files as supplied on the disk are:- AR 33 sectors

ED 33 "

EE 19 "

FW 33 "

FWDOCS/ARC 150 "

FWPRGS/ARC 286 "

LOAD 31 "

If you have only SS/SD drives or TI controller then you will have to request this when you order and the Librarian will make these up with the Archived Programs on one disk and the Archived Doc's plus AR, ED, EE, FW, LOAD on the other disk.

If you are one of the lucky TI'ers to have a V9938 chip for 80 column, a'la GENEVE or DIJIT AVPC system well there is a version for you. But please give full details of which Version you require to the Librarian when you place the order.

The full de-archived 40 col.version now requires 1001 sectors of disk space. There are some new doc's to read and extra programs to utilise in your quest to push the TI 99/4A to its limits, and if young Will's (now exploring MEGA TRACKER II in beta version ever survives the last blast off UNI exam stage) well you ain't seen anything yet !!!

All the program names now follow a 2 character pattern which as well as saving BYTES are more uniform. Thus CHARA1/2 becomes C1/C2 and UTIL1 is now FW.

CT8K/0 installs FW in the CACHE Module (E/A + 8KRAM)

Now supports Hard Disk systems and path names. Most of the 2 key press functions are dual eg :-

Ctrl A or Fct 6 is Proceed Ctrl C or Fct 9 is Back. .

Making one handed operation easy and are mostly standard, but all old key strokes are retained so do read each doc in case of some variations and also to get the full benefit as usual. Tony has put a lot of effort into the documentation and a lot of help full hints and tips. But I do believe in a broad review and not to cross each "t" and dot every "i" is more interesting. So make the user seek out more features and even to write some corrections or bugs not yet evident.

Version 4.21 is mainly compatible with prior Versions but it is recommended you Run your old saved SYSCON files to update with the latest version and resave them.

DS and UL also re edited by loading and making Reserve, edit and resave

Entry from XBasic unchanged and the screen is the familiar one the only new addition is :-

D 40 Col ED

This is only suggested and valid if you have the 80 col card and wish to use to 40 col Editor for some strange reason. It will of course be edited out as will the MYARC option, if these do not apply to your system. When you customise the LOAD program via CONFIG

CASSETTE and FORTH

Options for the above are only accessed from this XB screen.

FW Central Menu from E/A entry

1 TEXT EDIT 1 PROGRAM ED.
2 FORMATTER 2 ASSEMBLER
3 DISK 3 LOADERS
4 ARCHIVER 4 C-COMPILER
5 DATA-BASE 5 DISK-PATCH
6 DM-1000 6 LINEHUNTER
7 DSKU 7 RAG-LINKER
8 USER LIST 8 DISKREVIEW

Press > AID < from this menu and you are presented with screen :-

DSK ? > Ctrl= < Exit

Enter a drive number and mark file by pressing spacebar. Delete, set or unset protection. If you want to read another disk press enter and the above query is again shown.

When finished > Ctrl = < will exit you to the central menu screen

EXIT from the central menu is dual by pressing Ctrl.C or Fct9.

The screen will present :-

QUIT ? N (default)

Press ENTER and you have the option to reset all defaults.

Load Char-set

DSK 5 C1

Press enter if this is what you require.

NB. (disk #'s are the original one FW was loaded from)

Next default :-

PRINTER RS232.BA=4800
or PIO if installed permanently with GONFIG

Next default :-

From Drive TW/EA
55

Enter correct or change if required

If you do want to Quit, press Y and Enter and you exit to wherever you came from!!!

TEXT EDITOR modified to support 2 TAB setting > ST < will alternate between them. Both setting saved to disk with the file.

Ctrl A or Fct 4 page forward
" Q or " 6 page back
" E or " E line up
" X or " X line down
" S or " S cursor left
" D or " D cursor right
" Y L / R margin release

SD is now similar to the QD format and will also print a 2 column list of the disk directory. Auto Showing of Program Types is a bonus.

The Copy, Move and Delete functions are now LIGHTING FAST and the word wrap further fine tuned to the edge.

> Ctrl: <changes lower case letters to upper case and > Ctrl.< changes upper case to lower case. This is a little known feature but usefull for the Alpha Lock blues.

Program Editor now has a programmers cursor shaped like a star > Ctrl O < switches between the two. Feature now enables real lazy programming all in lower case and when the final comments are entered. The program line reformats to upper case and the comments remain in lower case as the cursor moves on to the next line.

Formatter unmodified but now loads as an ordinary PROGRAM file. This means you can if you prefer the R.A.Green Formatter or any other, substitute these files on your working disk.

Disk Utils is one suggested in the list of Utilities to use.

Loaders remain in the same format but Script loader simplified and supports PATH names. Sample SCRIPT is supplied as DEMI

Options 4 to 7 each side can be customised via the CONFIG and saved in the SYSCON file, so what is shown is ugested.

DM-1000 is supplied and has been modified to co-exist in the F'Web environment.

But at present it is NOT RELIABLE if your system uses a Cor-Comp or clone such as the P.Schubert AT Controller card when you are doing file copies to a RAMdisk.

Physical disks do not have this problem.

Note all versions or variants of DM-1000 have this problem. Tony published an in depth article about this in the MAY'88 HV99'ers Newsletter if you have any queries read this.

Disk-Patch is supplied and has been modified into a great utility.

Line Hunter is a programmers search utility, and was developed at the Farm by Will for their use, but is included as part of the package.

DISK REVIEW is totally new and developed at the Farm. (NOTE! A new Mascot in the form of a starved to death Funellweb Spider now rests atop Tony's Monitor)

The documentation is listed as :-

FWDOC/DREV for the 40 col.version,
FWDOC/DRAV for the 80 col.version.

Well worth reading but the program is User Friendly and should present little worries unless Murphy Law strikes again.

Entering Diskview the screen is divided into 5 well arranged lined sections with a pop up window in the centre.

In the window is request to :-

```
-----!  
!!  
! ENTER DRIVE : ? !  
!!  
! EXIT < ctrl= > !  
!!  
!-----!
```

Enter a drive number and the window disappears and in the larger right section appears all the usual file information.

Bottom right section shows the FWeb version number.

The section above contains :-

W ork file name
O bject file name
P rog. file name
X basic name appears if the pointer is on a basic or XB file.

Bottom left section shows

DSK # Name
Sectors used
Sectors avail.
File Count
Page 1 of #

Top left section is long and narrow showing :-

Mark -Spacebar
fct. 7/8
" E
" X
" View
UnProct.
Set Pr.
D el
P dir
O old
R un
f Copy
f Rename
CTRL H Header
CTRL A Procd

Yes you can now Rename or Copy files as well as View them. Renaming and you have the option to change the name as in DSKU a usefull feature.

When viewing a long file the buffer can fill but you no longer have the message BUFFERS FULL

It simply keeps on loading over the start until EOF is reached and the line numbers of the page show where you are.

Pressing Ctrl A when loading will load the whole of the file into memory, so you can flick backwards and forwards as often as required.

Ctrl @ zooms to SOF
" A " to EOF " E auto scrolls back
" X auto scrolls Forward
Hold Spacebar to read

any key to exit auto scroll

Ctrl = or Fct 9 to returns you to central menu.

There is so much more to say but the deadline is here for the Mag.

It is now getting more like the Disk Manager Tony has been going to do if he ever get's finished Tinkering and improving F'Web.
You can do a sector header read now.

I like it and so should all TI'ers

We'll hope I have wetted your desire to hurry to the Librarian and buy a copy soonest.

QED 4.6 / QUEST 6.10 UPDATES

by Al Lawrence

Out of the BLACK HOLE country from Ron Kleinschafer comes the latest version of the most useful programs for TI OWNERS with the Modules, E/A, QED or Super Space II and Horizion, or Quest RAMdisk Boards.

If you do not have a 32k module but have a RAMdisk and E/A module then Ron suggests the QUEST 6.10 disk is what you need to push the system to the limits.

Further modification of the DSR by Ron now completed and the updated disk is available from the HV99'ers Software Librarian.

Updates now mean on the board with CRU address >1000 selected, it is now possible to load and run an XB program by calling :-
RUN" DSK.NAME.filename" same as the Horizion will do.

New features, AUTO and AUIP are supplied on the disk QUEST 6.10 for anyone with only the E/A Module. But on request will be supplied for 32k modules system. PLEASE make full listing of your preference when ordering.

AUTO :- Shell program for the QUEST RAMdisk and with character set AUIP

Install both on the RAMdisk then from basic or Xbasic do a CALL AON
press enter, type BYE press enter.

Now each time you crank up the old 4A you have 3 pages, selectable to read or print a Directory, Files or run a program and to customise for all your programmes as required.
Alternating pages is by tapping the spacebar.

These will remain in memory each time you fire up, until you do a CALL AOF enter and type BYE which will return you to the normal title screen on start up. Which is pretty useless anyhow.

QED is also updated with an alternative to Super Bug if you do not use it. This being Archiver 3.03. the latest and greatest FREEWARE update from Barry Boone. (if you have not got it, GET it, use it and send a donation NOW).

Note...This is for QED modules only as it will not fit into the Super Space Module,

USE and ENJOY. Why not GET the disk, use it and pass it on, but PLEASE thank the Author with some COLOR or if you spell it COLOUR or \$\$ bread or somesuch THANK you RON for making our 99'4A much more usefull.

THANK YOU RON.

DEBUG YOUR QED MODULE

Ron Klienschafer

As most users of the 32K QED module knows it is a very handy addition to the TI's hardware, especially to users without a ramdisk.

Also as most owners know the most annoying aspect of the unit is it's habit of loosing the first byte in the current page in use if the console is switched off then on quickly, and especially if the whole powered up system is switched off at the wall socket. Users take heart, there is a simple cure, with the modification described below I have been unable to corrupt any data in it, that includes rapid operation of the console power switch, pulling the wall plug with everything powered up and even removing then reinserting the module in the console whilst powered up, and a programme on screen, in other words I just can't "blow it", and it has become a very, very stable unit.

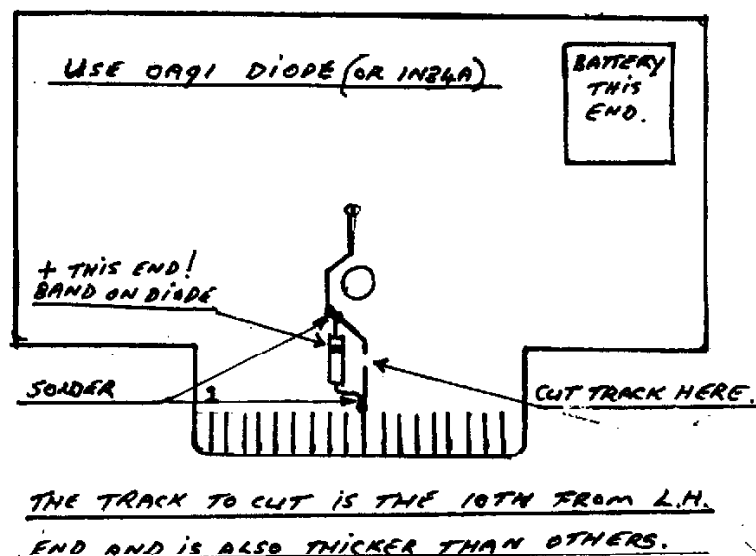
All the components needed is one germanium diode (approx 30c), type 0a91 or equivalent will do nicely.

Carefully cut the PCB track on the bottom of the module where indicated on the drawing, then solder on the diode as indicated, thats it!, simple huhh?.

I think that what happens is that the module tries to power some IC in the console when the console is powered down and the battery momentarily losses grip on the CE line on the Ram chip. Without tests I can't be sure what happens (anyone want to donate a 20 MHz scope???) but the modification works!.

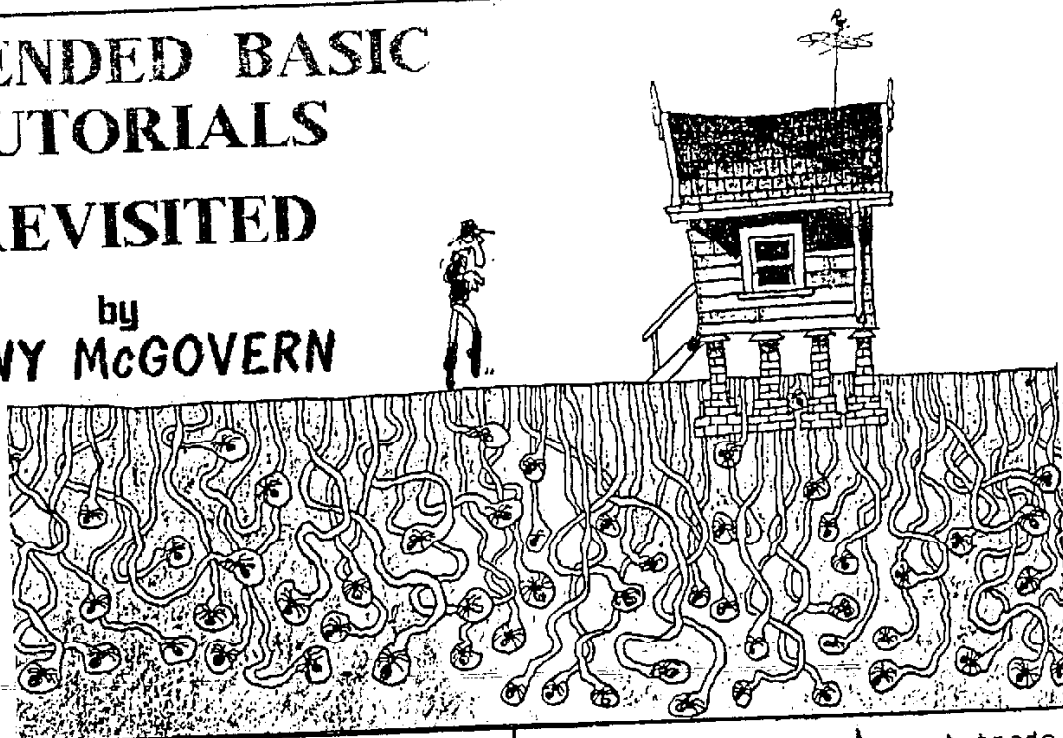
What about the superspace II module you ask?, well I don't know, I don't have one, but I have a feeling that one may mysteriously turn up in the post.

BOTTOM OF QED MODULE PCB.



EXTENDED BASIC TUTORIALS REVISITED

by
TONY MCGOVERN



VIII XB PROGRAM SCRUNCHING

Well, where do we go in the future? So far the series has had a detailed look at user SUBprograms and the ACCEPT AI statement, the two most powerful features of TI's Extended Basic, and also at the prescan switch commands lurking in the V110 manual addendum. For the next few sessions we will continue with topics which are of immediate relevance to console-only users, namely squeezing programs to fit in memory, and extracting maximum speed from XB. I can see no point, and have even less interest in writing about, say, SOUND or SPRITES from the very beginning, as these are fairly well documented in the manuals and the subject of many books and articles. That isn't to say that subtleties in using them won't come up from time to time.

Enough raving on for now and on to the real business. Let's now look at how to face up to that 'MEMORY FULL' message. This even comes up when you have memory expansion with a total of 48K of RAM in various guises. Programs always seem to end up needing more memory than is available! I do feel some unease in discussing this topic as many of the things that are done in compacting programs can only be regarded as poor program practice otherwise, as they make the code obscure and difficult to modify or develop

further. The other great trade off that must be considered when scrunching programs is speed of execution. Given an equal level of skill in program writing, coding for speed usually results in a longer program than would otherwise be written. Perhaps the easiest example to see is unrolling of a short loop which is repeated a fixed number of times. A FOR-NEXT loop gives compact code but carries a penalty of the loop overhead which could be avoided by writing out the contents of the loop the appropriate number of times. The subject of coding for speed will be taken up in detail in later Tutorials, and speed sacrifices with compacted code will only be noted in passing. The richer the language the more opportunities there are to optimize code one way or the other. Console Basic offers many fewer ways to do this than does XB and is much less fun.

At what stage should you bother trying to make your code compact? Remember that XB can only OLD or RUN one program at a time, so apart from loading time from cassette, or disk space, there is no reason at all to scrunch a program that runs in the smallest memory it is intended to run on. Most users with disks now have the 32K memory expansion, so this means the bare console. Minimum Basic programs to store in the module's RAM are the only ones you have real incentive to make

smaller still. Unless you know from the start that you are going to run short of space because of large arrays of numbers, or a need for maximum string storage room, be expansive -- document your program thoroughly with REMs, use lots of SUBprograms, use obvious explanatory names for variables, avoid reusing variables for unrelated uses and then you run out of room.

Now first of all a program has to be short enough to load. This is purely a function of program length. Next it has to be able to complete prescan when RUN. For prescan to succeed there must be enough room left over after the prescan for variable pointer and subprogram tables to be set up, and room set aside for numeric values, at 8 bytes per number. String variables are not assigned space until it is actually required, so it is possible for a program to crash later because it can't find enough room for strings. The well known hiccupping of long Basic programs occurs while Basic scratches around to reclaim string space when it has run out of new space. XB does it too, but it is a lot faster at 'garbage collection'. Now let's look at how to squeeze programs in, starting with things that affect the program length only.

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

Now it's time to look at what makes an XB program as long as it is. To

get started let's look at two very short programs to clear the screen.

```
100 CALL CLEAR
```

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

```
100 DISPLAY ERASE ALL
```

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

Now you don't have to take my word for this. If you have an expanded system you can write programs using CALL PEEK to explore stored programs, or better still use the E/A DEBUG (reassembled as uncompressed object code so the XB loader can handle it) for a quicker look. With console XB you can at best get an indirect insight by entering <CTRL+various keys> in a

REM statement and LISTing that. Be careful, you can crash the computer in ways wondrous to behold that way. Someone forgot to tell the computer not to try to turn token values back into reserved words when LISTing REMs. Ever notice when writing file specifications that keywords that do extra duty elsewhere LIST with the extra space, but the others do not. EASY-BUG in Minimem also allows you to look directly into VDP RAM or cartridge RAM to see Basic programs in their internal state.

In TI Basics, unlike those which store programs as ASCII files, the line number is always stored as a 2 byte integer, and it makes no difference to program length to use line #1 or line #10000. Try various line numbers in one of the examples above. If you are peeking around in the program, don't expect to find the line number at the start of its program line. It is in a separate table below the program, and each 4 byte entry has the line number followed by the location of the line itself. The line # table is sorted into order, but new or edited lines are always added to the lower address end of the program block. The program lines themselves are preceded by a length byte and terminated by a null (>00) byte. I won't go into it here but you can use this general information to interpret the various time delays when you edit a line or enter a new line.

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

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

and SIZE again. There is a saving of 5 bytes. The reserved word "::" has cost 1 byte, but 6 bytes have been saved by having one line fewer. Now if you scrunch a 500 line console Basic style of program into 200 XB multi-statement lines you have gained 1500 bytes. Of course you can't do this to every line because line numbers, as well as being line editor markers, are also where GOTOs and GDSUBs go, so you will usually end up with a few short lines you can't condense. FOR-NEXT loops work perfectly well within or

across multi-statement lines. The use of prescan switch commands is costly because you end up with !P+ and SUBEND on separate lines at the end of each subprogram so treated. Still, it's usually worth doing even though a long program may have several hundred bytes tied up in prescan switching. In desperation at the end you can always remove prescan switches starting with the shortest subprograms.

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

As I have noted in other connections in this series, TI in its self-defeating secretive way, never explicitly specified the details. TI Basic is most likely highly consistent in this from model to model, because any console can be called on to work with separate E/A or Minimem Basic support utilities such as NUMREF. On the other hand each XB module contains its own set of support utilities, and only has to be internally self consistent. There is information in TI's published data (XB, E/A, Technical manuals), giving details of VDP stack entries built by the E/A CALL LINK with some hints as to changes for the XB version. So to use XB LINKs at this level of detail you have to work by implication. Now it is done from time to time, but TI does not seem to have guaranteed explicitly to programmers that such procedures would work with all XB modules, or that the LINK stack entries are similar to internal table entries. Most likely they do and are. Only TI knows for sure.

hen again TI lost big while Apple and IBM make lots of money being more open about their machines, though Apple seems to be developing more secretive ways as it gets older and more arrogant.

Time for some little program experiments again. Enter the miniscule program

```
100 A=0
```

Before you do anything else work out how many bytes this uses. The answer is 11. In accepting the line the editor has already figured 'A' for a variable (because it starts with an allowable character) and not a reserved word and it is represented exactly as it occurs, no token involved. On the other hand it doesn't yet care that '0' is meant to be a number and treats it as an unquoted string. If it isn't an honest number, say 2N, it will only find out later when it RUNs and tries to convert it to a floating point number.

SIZE the program, then RUN it and SIZE again. XB does not reset everything until you have made an editing change, as you know from debugging efforts after BREAKing (fctn-4) program execution. At this stage you get more information from an expanded system, which will show

8 bytes of memory used and 9 bytes of stack. Now repeat the process with a longer variable name. The length is reflected both in the original program length and in the stack used. The stack usage is 2 bytes plus the variable's name length more than the minimum we figured out before. Most likely the 2 bytes are for a linked list structure to help table searching, and there is a symbol table entry of the variable name. Now turn off your expansion system and be like everybody else with console only, and repeat the above. Now you will find the increase over the program length is always the 14 bytes we figured earlier no matter how long the variable name is. Now try

```
100 A23456789012345,A2345678  
9012345=0
```

Still 14 bytes RUNtime overhead ! Change the second A to a B to make a distinct variable name, 15 bytes long. Only another 14 bytes overhead ! So how come ? Maybe it's

doing without the full word list link and squeezing things up a bit, but what about the symbol table ? The only consistent conclusion is that it doesn't have one as such, but points to the first location of the variable name in the program as located by the prescan. Read the Tutorial on prescans again. XB always searches in VDP RAM for variable names even if the program itself and the numeric values pointed to are stored in expansion memory.

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

The next Tutorial will continue with the principles of program scrunching, getting more into the program writing end of things.

THE LAST WORD.

This is it, last bit of space to fill. Looks easy enough, only a small section to fill. Problem is, no matter how small or large the space is to fill, it is hard when you can't get your brain into gear. (That is a task which I find difficult in good moment let alone now).

So what to say, for 1990 I have a big wish list. Firstly to all our members and their families, have a safe happy new year. To the peoples of the planet earth, hope 1990 brings more peace than 1989. To the planet earth, I hope that the peoples of the earth through 1990 learn to respect and look after you and all your plants and creatures.

So thats it, the last word, not so painful as I might have thought. Hope I get time to do a lot of programming this year. I've got heaps of ideas but little time to put them into practise.

Keep those articles comming in, won't you??

Joe Wright

THE INFORMATION PAGE

ATTENTION-----ATTENTION-----ATTENTION---

Bob Carmany will be visiting us in February. He will arrive in Australia on 26th February and leave Newcastle for Brisbane on 6th March. We have a tentative schedule for his visit.

The important dates so far fixed are; February General meeting on 27 th February. Bob will obviously be the main guest, he will be demonstration some of his programmes etc.

The following Saturday we will be having a family day in the Morpeth/East Maitland area including a bar-b-que lunch and visiting some of the historic sites in that area.

Then on Sunday 4th March a family day at the Pokolbin vineyards, wine tasting and a picnic lunch.

All our members are invited to these days. Times etc are yet to be fixed.

We are also planning some mid week activities for Bob, if you have some free time that week and would like to assist with entertaining Bob then please contact Pete Smith. One evening, preferably the Friday will be a night at one of the local clubs or pubs to Bob can sample the local culcha!

JANUARY MEETING.

The General meeting for January is on 23rd at Jesmond Community Centre, the main activities will be a demonstration of some not-so-well known software, which, however is really good.

COMMITTEE MEETING.

The first Committee Meeting for 1990 will be at the Boolaroo Ambulance Station at 7:00 pm on 13th February. We will be finalising details for Bob Carmany's visit. Please feel free to attend.

GENERAL MEETING FEBRUARY.

At the Jesmond Community Centre at 7:00 pm sharp on 27th. This will be a bumper meeting so be on time for a really good night of TI'ing.

SOCIAL CALENDER.

Some tentative social evenings are in the pipe line. If the Music Hall gets going after the earth tremor damage has been repaired, two nights are in the melting pot. The first is a Black and White minstrel night. The other is a Gilbert and Sutherland evening. Looks like at least the first one will include dinner at Sizzlers then the Theatre. Not too sure about the second as yet. More details as they get closer.

OUT OF TOWN MEETING.

Following on from Bev Warren's article and representations from R.K. we are kicking around the idea of chartering a (or a couple) of small planes to travel to some of the areas where our out of town members live and have a meeting. Have no idea what the cost would be, we would all have to pay our own fare of course. But if the costs come up ok the maybe we will spread our wings once or twice this year. We will sort out some costs over the next couple of months, keep an eye on this page for developments.