L.& G. Reid #77
Lot 138 Killigrew Rd.
TAMBORINE QLD.
4270

[illegible]UR COMMITTEE

[illegible]

Wilf[illegible] Wright
[illegible]

[illegible]
Ph. [illegible]0751

SECRETARY
[illegible]bert Anderson
[illegible] Close
[illegible]MO 2822
Ph. 662672
Vi[illegible]tel 4968[illegible]

TREASURER
Brian Rutherford
[illegible]
Ph. 49616[illegible]

[illegible]
[illegible] Lawson
[illegible] Bayview [illegible]
WARNERS BAY [illegible]
Ph. [illegible]

[illegible]
[illegible] Mulvany
[illegible]
[illegible] POINT [illegible]
Ph. [illegible]

EDITOR
Brian Woods
[illegible]
[illegible]MO 2822
Ph. 662307

COMMITTEE
Bob MacClure
75 Deborah St[illegible]
KOTARA SOUTH 2289
Ph. 437431

Gary Jones
53 Janet St[illegible]
JESMOND 229[illegible]
Ph. 573744

Tony McGove[illegible]
215 Grinsel[illegible]
KOTARA 2289
Ph. 523162

Peter [illegible]
25 Reserve R[illegible]
WANGI WANGI 2267
Ph. 751930

CONTRIBUTIONS

[illegible]

DISCLAIMER

The [illegible]
newsletter [illegible] the [illegible]
NINETY [illegible] GROUP [illegible]
effort [illegible]
correct[illegible]
information [illegible]
[illegible]
TEXAS [illegible]
and [illegible]
INST[illegible]
[illegible]
[illegible]
may [illegible]

```
1370 FOR I=1 TO 3                    1890 CALL HCHAR(19,22+I,ASC(
1380 CALL SOUND(200,550,2)           SEG$(STR$(TOTAL),I,1)))
1390 CALL SOUND
1400 NEXT I                          TO LEN(STR$(PLA
1410 CALL HCHAR
1420 CALL HCHAR                      HAR(21,22+I,ASC(
1430 GOTO 1870                       AYS),I,1)))
1440 TOTAL=TOTAL
145
146
147
```

EXPLORING BASIC WITH THE H.V. 99ERS BASIC GROUP

Hello its fill a page time again and by the time you read this only about sixty days to Christmas.

We certainly hope you have gained some benefit from these articles, because we are sure that we have through writing them. There has also been a few absent from our classes lately so we hope with the weather warming up they will be able to find there way along to the meetings again.

Last month we gave our solution to problem 2-1 but had to carry over problem 2-2 as like this month there is just not enough days in it. So finaly for those of you who had a crack at it here is our solution to it.

SOLUTION 2-2

```
10 RANDOMIZE
20 FOR LOOP=1 TO 6
30 NM(LOOP)=INT(29*RND)+1
40 FLAG=0
50 IF LOOP=1 THEN 130
60 FOR INNER=1 TO (LOOP-1)
70 IF NM(LOOP)=NM(INNER)THEN 90
80 GOTO 110
90 INNER=LOOP-1
100 FLAG=1
110 NEXT INNER
120 IF FLAG=1 THEN 30
130 PRINT NM(LOOP)
140 NEXT LOOP
```

Now with that problem answered we can enter our solution to problem 3-1 and 3-2 which both deal with arrays. Arrays allow you to arrange your data so it may be stored and retrieved easily.

SOLUTION 3-1

```
100 DIM SCORE(10)
110 CALL CLEAR
120 FOR A=1 TO 8
130 READ SCORE(A)
140 PRINT SCORE(A)
150 NEXT A
160 DATA 10,25,39,42,58,61,71,84
```

SOLUTION 3-2

```
100 DIM SCORE(10)
110 CALL CLEAR
120 FOR A=1 TO 8
130 READ SCORE(A)
140 PRINT SCORE(A)
150 SUM=SUM+SCORE(A)
160 AVER=SUM/8
170 NEXT A
180 A$="THE SUM EQUALS:"
190 B$="THE AVERAGE EQUALS:"
200 PRINT A$;SUM
210 PRINT B$;AVER
220 DATA 10,25,39,42,58,61,71,84
```

We have to explain; the DIM statement we put in our solutions is not necessary when storing ten or less pieces of data but must be used to set the depth and dimension of larger arrays. Sticking with arrays this weeks problems are to expand on what we have just done.

PROBLEM 4-1

Modify the results of problem 3-1 to allow ten numbers to be input not using data statement.

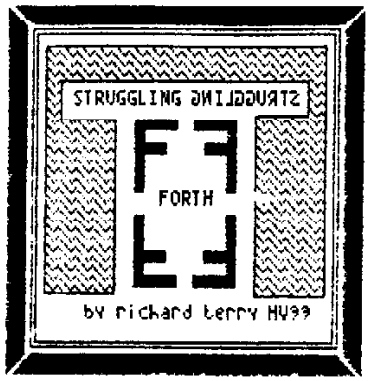
PROBLEM 4-2

Now modify the program again. Use a two dimensional array to enter a second set of ten numbers and then print result assuming the first set of numbers are exam results for student A the second set for student B.

As usual we would appreciate any contibutions or help with the artical please send input to this address:

ALAN FRANKS
C/O THE SECRETARY
HV99 USER GROUP
6 ARCOT CLOSE,
TARRO 2322

```
 0  ( OCTOBER/01)
 1  ------------------------------------------------------------
 2  -                                                          -
 3  -   STRUGGLING FORTH                                       -
 4  -                                                          -
 5  -   HV99ERS  OCTOBER ARTICLE                               -
 6  -                                                          -
 7  -   BRIEF NOTE ON HEXADECIMAL                             -
 8  -   POSTFIX MATHEMATICAL NOTATION                         -
 9  -   THE PARAMETER STACK                                    -
10  -   ADDITIONAL STACK OPERATORS                             -
11  -                                                          -
12  -                                                          -
13  -                                                          -
14  -                                                          -
15  ------------------------------------------------------------
```



This month I promised to talk about the stack and postfix mathematical notation. I'm going to keep the discussion as simple as I can at the risk of boring you. I'll try not to assume you know anything!

A note on Hex.

For those of you reading the assembly article last month you will have noticed Joe's recommendation to buy a Hex calculator. In Forth we use a lot of Hex in our programs, BUT, we have our own in-built calculator. Though we write our code to our SCREENS (see last week), we can at any stage exit using the function 9 key to IMMEDIATE mode. Say we wanted to convert a number from decimal to Hex, we can type BASE HEX (or simply HEX),and the number base will be hexadecimal. For example to convert the decimal number for the character <A> to its equivalent in hex we type:

```
65  (decimal ascii for A)
HEX (change base to HEX )
<enter>
press .(print the result)
```

and press enter. Forth will reply with:

```
41 ok.
```

Before we go on all further occurrences of press enter will be denotated <enter> or <e>.

POSTFIX NOTATION.

Everyone will be familiar with your old primary school days when you started your sums. Lets use the classic example:

```
2 + 2 = 4
```

This is called if my memory serves me correctly INFIX notation. Here we have the items to be added logically joined by the operator, in this case the + sign. Because of the way Forth words work, not worrying why now, the construction of this calculation is different. In Forth we would write:

```
2 (put a two on stack)
2 (put a two on stack)
* (multiply these two)
<enter>
press . (print the result )
```

To which Forth would reply:

```
4 ok.
```

There are about 30 arithmetic operators available to you, many of which you will probably never use, but they all work this way. I know it seems daunting, but after a while you will feel strange putting the plus sign between numbers! Try doing simple sums at your terminal now using other operators, for example:

```
using minus sign (-)
3 3 - <enter> .
0 ok
or
using divide     (/)
12 4 / <enter> .
3 ok
```

and so on. Experiment with your own sums. I remember when I started reading Brodie I got terribly confused on this simple concept, and

felt really inadequate. I couldn't do his examples properly. However once I used them in practice it became elementary.

I'd also like you to make a mental note that all these arithmetic operators are in themselves words in the Forth dictionary, ie they are definitions in the same way everything else is, but we will look at that in a later article.

By now you've either got bored and disappeared, or your curiosity is asking you where all these numbers disappear to when you type them at the terminal. The answer is that dreaded word: The stack.

THE DREADED STACK.

My recollections of my attitude to the stack bring back memories of confusion and fear. Fears that it was too complicated to master. What the hell was it. Where was it? How to conceptualise it?

Let me try and draw a couple of comparisons.

You are probably familiar with extended basic, either in practical usage or with scanning programs. Now think about an extended basic subprogram such as listed below:

```
1000 SUBCONTINUE(X)
1010 CALL KEY(0,K,S)::IF S=0
     THEN 1010
1020 IF X=1 THEN PRINT
     "INSERT NEXT DISK"
1030 SUBEND
```

In this hypothetical case, our subprogram SUBCONTINUE is an entity unto itself. It lives within your program, but the only communication it has with it is by any data passed back and forth to it by the variable X. Think of it like a prisoner behind bars. He is confined to the cell and can only interact with the outside world by passing things be they food/letters files etc through the bars of the cell. In doing so he can affect those outside however.

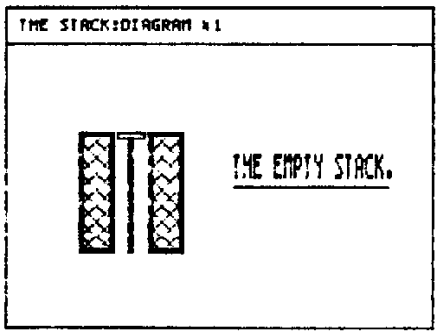
Comparing this subprogram to Forth we may have a word we call SUBCONTINUE. For it to operate it needs data. It is an island unto itself as well. We supply the necessary data by placing it on the stack in a manner we will see later.

Lets think of another more complicated example. Your local council is having an election, and is looking for trustworthy people to count the votes, so naturally they pick members of the HV99ers. In the counting room there is a table in the middle, which will represent the STACK. Sitting at the table is El Presidente holding 3 cards. Against one wall there are three counting tables piled high with ballot papers clearly marked 1 2 3. You, the COUNTERS now become in our analogy the equivalent of a Forth Word. You are going to count a candidates total vote, but you will not know which one. As each of you approach the table El Presidente places a card on the table (the stack). You pick it up and proceed to the table which corresponds to your number. The table top is again empty(ie stack is empty). The next COUNTER steps forward and the process is repeated. IE you, the Forth word, have picked up data from the top of the table, leaving it empty, and used that data to choose what to do, in this case walk to a counting table. You now count the votes. Time passes... El Presidente knocks back a few more cans (there must be some perks), the table top (the stack) remains empty. Suddenly the votes are counted. You have no-idea whose votes you've counted, all you know is the total was say 550. You write this on a card in big black texta-coloured letters, walk back to the table and deposit the card in the middle of the table, along with the original card showing which table you came from, kick El Presidente in the shins to wake him from his inebriated slumber, and leave the room. He grunts a couple of times, takes the two cards from the table (the stack) and using the table card as a reference, places the vote total in a correctly numbered pigeon Hole. You have performed an important task taking data from the table (the stack) and later putting different data back on the table (the stack), without knowing who originally set up the system of counting, whose votes you counted etc. The others complete their task similarly. Now from out of the dimmer recesses of the room

comes the organiser, say Le Secretary. He knows the codes. From pigeon hole number 1 he takes the total vote count and places it on the table (the stack). On top of this he places a written card with the correct candidates name,and leaves. Another cog in the wheel we will call the BOARDMAN - whose only function is to pin the results to a notice board - picks the name off the table (the stack),leaving the vote count still on the table (the stack-is now not empty yet), and pins it on the noticeboard. He then returns to the table (the stack),removes the count (stack empty), and pins this result under the name.

This all took place without any verbal communication. Each player knew his own function and performed it sequentially as called. Their only communication was via numbers/names, left on the table. So it is in Forth. Our words perform their functions by taking data from the stack, going away and doing their business in the bowels of the computer in little rooms of their own, and return leaving their result on the stack to be used by another word, or sometimes not needing to leave any result at all.

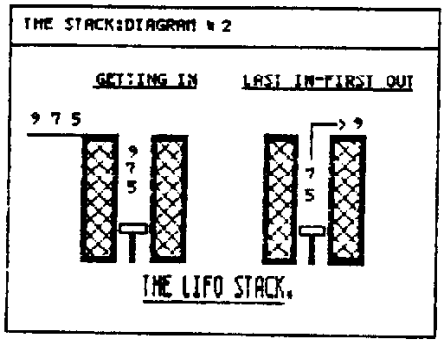
Lets look at another conception of the stack. a la Brodie with his hole with a spring:



Lets look at the properties of the stack. Firstly as a hole, its very deep. During your programming you should never receive a "stack full" message unless something has gone terribly wrong. You will find you will usually only need between 1->6 items on the stack at any one time, any more and its too hard to keep track of, and its a warning signal your programming has become sloppy and needs to be re-appraised. Lets put a few numbers on and see what happens. Do this at your terminal:

5 7 9 <enter>

Diagramatically this looks like:



Now lets remove them using the Forth word: . (pronounced print),this word when executed has the blindingly mindless task of walking up to the stack, taking off the top item and printing it onto the screen in front of you. Lets use it sequentially to see what happens when we remove items from the stack:

press: . <e> 9 ok . <e> 7 ok . <e> 5 ok

NOW lets do it all in one action:

5 7 9 . . . 9 7 5 ok.

ie they came out in reverse order that they went in. Hence the FORTH stack is called a LIFO stack ie Last In First Out (refer to Diagram #2) . I know this seems terribly elementary but its essential to all stack programming to grasp the concept. Try experimenting with numbers on your stack.

STACK OPERATORS.

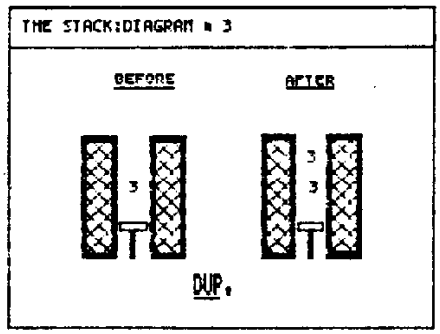
Sometimes having one copy of a number on the stack for use by the following forth word is not enough, or the numbers may be in the wrong order. Forth provides for this by forth words which operate solely on the stack.

Lets extend our Ballot Room example above. Outside our tally room we actually have crowds of people waiting for the result, so many that they are hanging around a notice board in the front, and another at the back of the building. When our

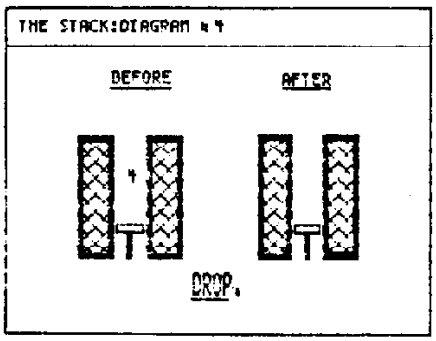
BOARDMAN arrived at the table, he was faced with a dilemma. He vaguely knew he needed two copies of the name, and later the vote count result, one each for the front and back notice boards. Being a bit of a dim-wit (we think he was a visiting Tishug member), he calls another HV99er, The multi talented DATA-DUPLICATOR. At a glance this genius realises the solution. He will have to duplicate the name card on top, which diligently he does with his texta pen. On the table we now have two copies of the name,underneath which is the count. The BOARDMAN takes the two copies of the name and pins each on a noticeboard, returns, and finds the DATA-DUPLICATOR has already operated on the remaining count card on the table and has DUPlicated it ready for use.

And so it is in Forth:

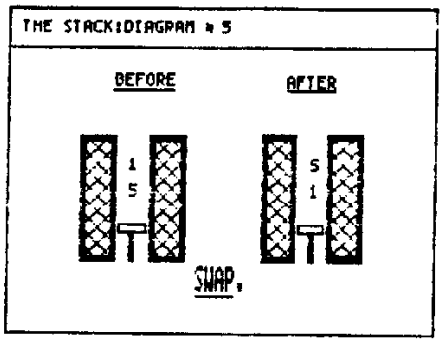
DUP duplicates the top number



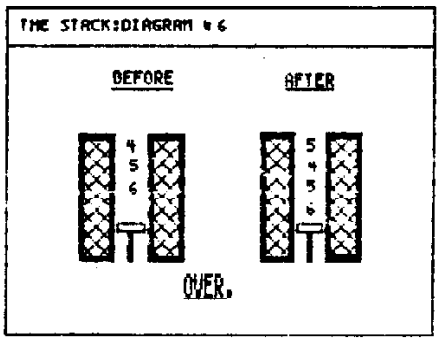
DROP discards top stack item



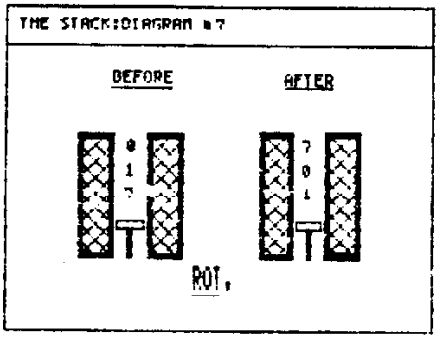
SWAP exchanges the top two items



ØVER makes copy of second onto top



ROT rotates third item to the top.



There are others definable, and for the curious I include some additional stack operators on SCR #152, with a description of their action on SCR #153. I would advise against indescriminate use of PICK and ROLL, as my own feeling on it is though useful at times it encourages sloppy thinking - why worry about stack orders if you can pick what you want and put it on top.

TACK DIAGRAM CONVENIIONS.

convention there is a shorthand
y of describing what the stack
pects, and what is left after the
rth word operates:

(before ----- after)

(n1 n2 n3 ----- n4)

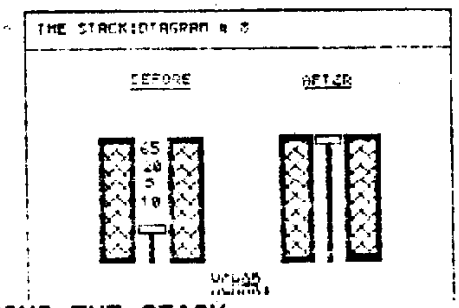
e most left hand number is deepest
the stack, the most right hand
e is the most superficial ie here
is on top of the stack before our
pothetical word needing this
nfiguration is executed. To give
u a practical example refer to the
HAR word on page 37 of Appendix D
the Glossary of the Ti-Forth
nual. This works the same as the
HAR of extended basic. In Forth
must provide it with:

(c r cnt ch ---)

column r=row cnt=number to print
=character to print.
leaves nothing on the stack. For
ample to print a row of A's
arting at column 10, row 5 we
uld type:

5 20 65 HCHAR <enter>

oking at it vertically the stack
uld look like this:



ARING THE STACK.

ry so often you need to closely
mine the stack to de-bug
grams. When something has gone
ray there could potentially be
dreds of characters on the stack.
ting tapping the print command
repetitively would be a real
e. You could load the DUMP
ions to examine the stack
tents, but this is rarely
essary. To clean the stack back
its base layer simply type in
thing which is not already
ined eg: XX , to which Forth will
reply:XX? and in the process re-set the stack, which inspection with the Print(.) command will show:

Ø
N
11766

THE RETURN STACK.

Just when you thought life was becoming clearer up pops the return stack. I make mention of it here just to let you know it is another area where we can store data. We won't use it today until you get used to the ordinary parameter stack. Indescriminate use of the return stack can lead to system lock up, as the system uses it for its own data handling. However it can be a great place to hide numbers temporarily during a definition as will become apparent to you later.

Thats all for This month, next time, I'll try and examine what the Dictionary is in Forth, and reply to the contents of Joe's article.

FOR MORE ADVANCED USERS

For HV99ers, I have codings available for such diverse things as:

-Assembler optimization of Floating point routines
-Sound chip routines
-Forth De-compiler
-a couple of different clock programs -codes for double density: I know I've denigrated the latter as a disruptive force, but since then I've discovered easy ways to convert between the two, so if you have double density capacity I'll give you a copy of the screens necessary to convert, or if you like a binary image of the disk with the auto-repeat editor, but in double density.
-Source code of the Kernel
-flow diagrams for all Forth Kernel words
-Forth Font program
-3 pass diskcopy
-plus much more.

ADDRESS FOR CORRESPONDENCE

RICHARD TERRY
141 DUDLEY RD
WHITEBRIDGE 2290
NSW AUSTRALIA

```
SCR #152
  0 ( OCTOBER/02:ADDITIONAL STACK WORDS     )
  1
  2 : PICK          2 * SP@ + @ ;
  3 : ROLL          DUP 1 =
  4                 IF DROP ELSE DUP 1
  5                 DO SWAP R> R> ROT >R >R >R  LOOP
  6                 1 DO R> R> R> ROT ROT >R >R SWAP LOOP THEN ;
  7 : 2DUP          OVER OVER      ;
  8 : 2SWAP         ROT >R ROT R> ;
  9 : 3DUP          OVER OVER >R >R >R >R DUP  R> R> ROT R> R>  ;
 10
 11
 12
 13
 14
 15

SCR #153
  0
  1
  2 PICK :Stack expects the Nth number you wish to copy and places
  3       a copy on the top of the stack
  4 ROLL :Stack expects the Nth number you wish to move, and "rolls"
  5       this to the top of the stack,ie moves it from where it was
  6 2DUP :Duplicates the top two stack numbers
  7 2SWAP:swaps around the top two pairs of numbers
  8 3DUP :duplicates the top three numbers on the stack
  9
 10
 11
 12
 13
 14
 15
```

HOME COMPUTER MAGAZINE

This article is re-printed from 'BITS, BYTES & PIXELS', the magazine of the LIMA 99/4A Users Group of U.S.A.

If you think your long term subscription to 99er MAGAZINE, or to HOME COMPUTER MAGAZINE was improperly terminated when the magazine changed its name recently, YOU MAY BE ABLE TO DO SOMETHING ABOUT IT.

Send a letter of complaint indicating how many issues are left in your subscription and how much money you paid. Send your letter to:

District Attorney's Office,
Consumer Affairs,
400 Lane County Court House
EUGENE OR 97401
U.S.A.

The D.A.'s office will send a copy of your letter to the folks at what is now called HOME COMPUTING JOURNAL and ask them to tell the DA's office how each case is going to be resolved. In an AUG 6 phone conversation, a spokesperson at the Consumer Affairs section said that they knew of some people who had received a settlement from H.C.J.

DATA, READ, RESTORE

This article originally peared in the August 1985 issue of T-BITS, the magazine of the T.I. ers Group of Perth. The author's me is not known.

The READ-DATA-RESTORE statement plemented from both versions of SIC are powerful tools if used rrectly. If not they can be a in in the **%#######.

Many lost hours have been used by a simple, silly mistake in ing these statements.

DATA statements can save a rge amount of programming effort d memory space. Consider the llowing programmes:

```
 A=2
 B=3
 C=4
 D=5
 PRINT "X = ";A*B*C*D
 END
```

```
 READ A,B,C,D
 PRINT "X = ";A*B*C*D
 DATA 2,3,4,5
 END
```

Both programmes serve exactly same purpose but the second gramme is 2 lines shorter. If don't believe it, try them out. , they both do exactly the same ng.

How did the second programme k:-

The keyword READ tells the puter that some variables follow ch don't have any value yet. To d their values, the computer rches for a DATA statement, where values are located.

In our example, at line 1ØØ, the computer "sees" the keyword READ, and then A; it searches for a DATA statement, finds it and stores the first value in the DATA statement in location A.

```
1ØØ READ A,.....
11Ø
12Ø DATA 2,.....
```

Values for B,C & D are found in the same way.

When it has finished with line 1ØØ the computer has given A the value of 2, B the value of 3, C the value of 4 and D the value of 5. At line 11Ø the value of X is calculated and is printed on the screen.

There are several very interesting variations possible with the READ-DATA statements.

1) We can have more than one READ statement for one DATA statement. The various READ statements use the DATA statements one by one. When a value has been used it CANNOT be used again, unless something special is done. This will be explained later.
For example:-

```
1ØØ READ A,B
11Ø PRINT A+B
12Ø READ C,D
13Ø PRINT C+D
14Ø DATA 5,1Ø,15,2Ø
15Ø END
```

Here's what happens; the READ A,B statement in 1ØØ looks for the first DATA statement and gives A the value of 5 and B the value of 1Ø. The READ C,D statement looks for the first DATA statement and recognizes that the first two values have already been used and uses the next two values, 15 and 2Ø for C and D respectively.

2) We can have several DATA statements. It does not matter to the computer where the DATA statements are located in the programme. It is better though if all the DATA statements are together. Nor does it matter how many DATA statements are used. The computer combines all the DATA statements into one big list of values, which will be used one by

one by the READ statement. So:-

```
100 DATA 2,3,4,5
```

is the same as:-

```
100 DATA 2
110 DATA 3,4
120 DATA 5
```

It is a wise move not to have more than three programme lines for each DATA statement and to keep the same number of data variables in each DATA statement.

Example:-

```
100 DATA 1,2,3,4,5,6,7,8,9,0
110  DATA  10,11,12,13,14,15,16,17,
18,19
```

Each of the above has an identical number of variables in each statement, ten to be exact. This makes it easier to keep track of how many DATA variables have been used. The importance of this will be seen later.

3) Two other possibilities can occur:-

a) One is that there are fewer variables in the READ statement than there is in the DATA statement. In this case, only the values in the DATA statement needed by the READ statement are used.

```
100 READ A,B
110 READ C,D
120 DATA 1,4,5,10,20,40,80
```

In this example the DATA variables 20,40 and 80 are not used.

b) One the other hand, there may be fewer values in the DATA statement than the number of variables in the READ statement. If the computer finds it needs more values than are provided it HALTS the running of the programme and issues a DATA ERROR IN XX, where XX is the line number of the READ statement in which the error occurred.

```
100 READ A,B
110 READ C,D
120 DATA 1,5,20
```

This programme would cause a DATA ERROR IN 110 to be printed on the screen.

The moral of this story is that you must ensure that the variables and the DATA match, if that is what you want.

It is important to note that both numeric and string (word type) variables can be used in the READ-DATA statements, but there are some rules that must be applied. For example:-

```
100 CALL CLEAR
110 READ A,B$,C
120 PRINT A;B$;C
130 DATA HI,THERE,5
```

This programme will cause an error as the first DATA statement is a string, and the computer was expecting a number. A string variable is denoted by the $ sign after the variable named in the READ statement. This can be corrected by changing line 100 to read:-

```
100 READ A$,B$,C
```

To what use can the READ-DATA statement be put? The scope for its application is unlimited, a few examples are listed below. Try them out for yourself and see the results.

#1

```
100 CALL CLEAR
110 FOR N=1 TO 28
120 READ C
130 CALL HCHAR (10,N+2,C)
140 NEXT N
150 DATA  72,105,32,73,32,97,109,32,
97,32
160  DATA  68,97,116,97,32,100,101,
109,111,110
170 DATA  115,116,114,97,114,97,116,
105,111,11 0
```

This could have been implemented in the following way as well:-

```
100 CALL CLEAR
110 FOR PRNT=1 TO 6
120 READ A$
130 PRINT A$;
140 NEXT PRNT
150 DATA This is an example
160 DATA of a DATA statement except
170 DATA it requires only one PRINT
180 DATA statement to serve the same
190 DATA purpose as multiple PRINT
200 DATA statements.
210 END
```

The first programme acts as a RINT statement. Where the nformation is stored in a "hidden" ormat. The DATA statements are ctually the ASCII codes for the haracters that appear on the creen.

See if you can have the ollowing printed out using the same ormat:-

EXAS IS MY HOME

INT! The ASCII codes are listed on age III-1 of the USER REFERENCE UIDE.

```
0 CALL CLEAR
0 FOR NOTE=1 TO 10
0 READ A,B,C
0 CALL SOUND(A,B,C,)
0 DATA 75,294,0,75,294,0,75,295,
75,294,0
0 DATA 150,294,0,74,294,
75,294,0,150,392, 0
0 DATA 150,440,0,150,494,0
0 NEXT NOTE
0 END
```

STORE

As was stated earlier, as each TA statement is used it may not be ed again UNLESS a RESTORE tement is used. This may take e of two forms:-

```
0 RESTORE

0 RESTORE line number
```

If the first statement is used n the next READ statement will e again start at the first, or est line numbered DATA line and l then recycle. If the second tement is used the READ statement l take the data that is stored at t line number, or the next DATA tement after that line number, as first DATA statement that is to re-read and then recycle.

MARY

For giving many variables ues, READ-DATA statements are h more efficient than INPUT or statements, especially if the gramme is to be run many times.

The READ sta'ement names the variables in which the values are to be stored.

The DATA statement contains the values which will be stored in the variables.

It's the programmers resposibility to ensure that the variables in the READ statement match the values in the DATA statement.

EDITORIAL

October already! This year has really flown! My mother always said that time really passes you by as you get older, but I thought she was only kidding!

Again this month we have a fairly well balanced magazine, thanks to our regular contributors, as well as reprinted material from other sources. There are articles to suit all levels of TI know-how, FORTH, XB and ASSEMBLY, as well as our regular Adventurers Corner, Librarians column and our recent addition, The Hunter Valley Software Library Guide (thanks Paul). Again Ron K. comes good with an article on Hardware.

As many would not be aware, our resident Adventurer, Rodney Gainsford, has TRADED HIS TI!!! in for an APPLE, but is more than happy to continue writing his Adventurers Corner. Thanks indeed Rodney. It just goes to show you can't get the TI out of your blood no matter what other computer you may decide to buy.

Don't forget that the Newsletter is not published in January (even the Editor needs a break!), but our December issue we hope will be a whopper!! To help achieve this please, please contribute something. I am giving such early notice so that all the 'GUNNERS' will have time to make up their excuses.

Well, that's enough naging from me, enjoy the magazine, and please try to contribute something soon.

Brian Woods,
Editor

HARDWARE HACKERS CORNER

FROM CHAOS MANOR.

Ron Kleinshafer HV/99.

MAKING PRINTED CIRCUIT BOARDS

There are three ways to get a printed circuit board for that "YOU BEAUT GIZZMO" you want to build or try out. One is to buy it, if available !!, the second is for someone to give it to you and the third is to make it yourself, the third is the only area that perhaps I can assist you in. By making them yourself you can use many and varied methods, some of these being.

PHOTOGRAPHIC.

This involves getting the necessary artwork, positives or negatives, using precoated sensitised laminates (or coating your own then baking), exposure, development, then if all is OK. etching, then finally drilling. A long, messy and costly process for one off boards.

DIRECT ETCH TRANSFERS.

This method has advantages for one off boards, whereby the pads, tracks etc. are on transparent sheets and by laying the required symbol over the laminate then burnishing with a pencil, the symbol is stuck down and will resist the etchant. The one big disadvantage is getting accurate component placement and almost impossible to make double sided boards if required.

VERO OR TRACKBOARD.

Handy for very small boards or for testing circuit designs but if the board component density gets to any reasonable size the whole thing starts to look like a rats nest and very amateurish. There is another way!!.

>>> ROLL YOUR OWN <<<

This method is simple, inexpensive, accurate with either single or double sided boards, you can use published designs (full size) or design your own, they can be small or very complex. The only limitation is that the tracks cannot be narrower than .5mm, but not many hardware hackers want tracks down to .005" (15 thou.) as can be accomplished with photographic methods. This method may not be new but I have not seen anything published about it, I call it.

DRILL FIRST ETCH LAST.

MATERIALS REQUIRED.

DALO PCB resist pen
1 roll .5mm precision slit artwork tape (BISHOP GRAPHICS)
(For signal lines etc.)
1 roll 1mm precision slit artwork tape (BISHOP GRAPHICS)
(For power etches etc)
1 small roll automotive type masking tape
Artwork knife (EXACTO or similar)
1 ANCOL DIN 250 spirograph
(available from newsagents)
2" nail ??
PCB drill (your own or your neighbours)
Etchant (Ferric Chloride or your preference)

OPTIONAL.

1 pack GEOTYPE transfer lettering (small size)

There is enough tape etc to produce, on average size, approx 10 to 20 PCB's depending on track density.

Obviously the first requirement is the artwork or drawing of the required finished circuit, full size and accurate. These can be from electronic magazines or other published articles, or if you design your own then the drawing has to be done on one tenth inch grid graph paper or film. One tenth inch grid is the standard for almost every electronic component (1/10 = 2.5mm).

Armed with the artwork cut to size (if you dont want to cut up your magazine photostat it and use

hat). Obtain a piece of copper aminate cut approx 10mm larger in idth and length than needed, (dont other to clean it yet) now take the rtwork and centre it on the aminate then tape it down with lear cellotape, keeping it very lat and taut, tape right round all dges. File or grind the 2" nail to round sharp point of approx 30', ow go over the artwork and centre unch every hole ACCURATELY, practice n a piece of scrap laminate first o get just a nice indentation to tart the drill, as you work mark our artwork lightly with a pencil o make sure you get every hole, hen finished remove the cellotape nd artwork then drill every hole th a Ø.8mm drill only.

Now clean the board by polishing th very fine (1200 grit) wet or y silicon grit paper, this removes e drill swarf and leaves a neat ean hole, do NOT use steel wool it s grease on it and the tape etc ll not seal properly, if further eaning is required then scrub with powdered kitchen cleaner, wash roughly and dry with paper towel oil free cloth.

Once the board is dry take up e DALO pen and holding it at 90' the board put the tip on a hole d turn it around several times, is leaves a neat small round pad ound the hole, do not press down the pen or the resist will flood, this happens dont attempt to wipe off, let it dry then it is easily moved with the tip of the EXATO ife then redo it. After all holes ve been done let the resist dry. ere any larger pads are required e the spirograph by centering the propriate size hole over the illed hole and run the DALO pen ound and fill in to the centre, er these have been done again let resist dry (a fast method is to the board in a moderate oven for ew minutes).

Now you play join the dots!!. ng the artwork tape start at a e and lay down the track to the t hole as per your artwork, press n lightly as you go and do not etch the tape, hold the knife n where you want to finish and l up on the tape, it will cut an but the knife may mark the ist, if so just touch up with the of the pen. If any sharp turns required cut the tape at the n then at the end of the tape just put a little resist with the pen then continue in the direction you wish to go, this also applies to "T" joins, just put a little resist alongside the track you want to join to then run your next track off the top of that. The masking tape is for very large tracks or ground etch fill ins and is a very good etchant resist. Repeat everything for the other side if it is double sided.

Once the resist, tracks and pads are laid down recheck that you never missed anything then get a sheet of paper, lay it over the board and rub all the tape down firmly with your finger, If you wish, important pads or pin numbers may be identified with the GEOTYPE lettering, when transferring these to the board try to only rub down the letter or number with the tip of a pencil because these use wax for adherence and widespread rubbing will not leave a very clean symbol. When satisfied etch in the normal manner. When etching is completed soak the board in petrol, rinse off and polish clean with a steel wool soap pad, this will prevent the copper tracks from tarnishing very quickly. THEN drill any larger holes to size that are required.

Double sided boards where IC sockets are used and the pins are soldered both sides presents some difficulty but by making the track from the pad under the socket slightly wider (1mm) for a short distance out then push a fine wire through the hole and bend it over onto the track, then solder the wire to the track, fit the IC socket and solder the bottom in the usual manner, check for continuity.

Edge connectors can be done with direct etch transfers but these must be plated and can be done with ENPLATE 421-ELECTROLESS IMMERSION TIN PLATING PROCESS, details are available with the product. (Available from CIRCUIT COMPONENTS (A/ASIA) PTY LTD.). After the board is completed and assembled clean off resin with metholated spirits then a good corrosion protection is to spray with DULUX metal finish clear, this can be washed off anytime with metholated spirits, for repairs or changes, then respray again.

Using this method produces very good PCB's and you will soon develop your own "style", it is surprisingly quick once mastered.

R.K.

ASSEMBLY LANGUAGE FOR THE LAYMAN

Hello! Well another month is here and it is probably timely to remind all ourselves that good ole' SANTA is fast approaching. Another year, my it seems to go so quickly, almost as quickly as TMS 9900 machine code runs!. I would like to remind people following these articles that I am always open for requests for special information.

OTHER WAYS!
****

As I mentioned last month there are as many methods of getting the character A DATA ready for the VSBW routine as there are programmers. No doubt each one sure that his or her method is the greatest. The two SOURCE CODE LISTINGS shown below illustrate two methods different to the original in the first article. Often, you will find that when you are writting a programme. The way in which you have constructed your programme dictates the method of collecting the data ready for use by a utility or routine.

SOURCE LISTING 2
****

```
0001 * SECOND TUTORIAL PROGRAMME
0002 * 14-9-86  FILENAME SOURCE=EA/PROG/2
0003 *                  OBJECT=EA/P/2OBJ
0004        DEF  START       PROGRAMME NAME
0005       REF  VSBW        REF TO VSBW
0006 RETURN DATA >0000       SAVE RETURN ADDRESS
0007 CHARA  DATA >4100       CHARACTER CODE FOR A
0008 WSREG  BSS  >20         SET ASIDE BLOCK FOR W/SPACE
0009 START  MOV  R11,@RETURN SAVE RETURN ADDRESS
0010        LWPI WSREG       CREATE MY W/SPACE
0011        LI   R0,>2FF     ADDRESS LAST SCN POSIT.
0012        MOV  @CHARA,R1   CHAR CODE INTO R1
0013 LOOP   BLWP @VSBW       BRANCH TO SCREEN WRITE
0014        DEC  R0          DECREASE R0 BY ONE
0015        JNE  LOOP        EQUAL TO ZERO? GOTO LOOP.
0016        MOV  @RETURN,R11 RETURN ADDR INTO MY R11.
0017       RT
0018        END
```

The first change here is in LINE 0007. The character code for the character A (>41) has been moved. The code has been placed in the most significant (left) byte of CHARA. Since the information we want is in the most significant byte, after MOV on LINE 12 there now is no need to swap the byte. So the SWPB instruction on LINE 13 is removed. Now the MOV instruction shifts >4100 from CHARA into REGISTER 1. Remembering of course that VSBW expects the code to be written to VPD RAM to be in the most significant byte of R1.

SOURCE LISTING 2A
****

```
0001 * SECOND TUTORIAL PROGRAMME
0002 * 14-9-86  FILENAME SOURCE=EA/PROG/2A
0003 *                  OBJECT=EA/P/2AOBJ
0004        DEF  START       PROGRAMME NAME
0005        REF  VSBW        REF TO VSBW
0006 RETURN DATA >0000       SAVE RETURN ADDRESS
0007 CHARA  DATA >4100       CHARACTER CODE FOR A
0008 WSREG  BSS  >20         SET ASIDE BLOCK FOR W/SPACE
0009 START  MOV  R11,@RETURN SAVE RETURN ADDRESS
0010        LWPI WSREG       CREATE MY W/SPACE
0011        LI   R0,>2FF     ADDRESS LAST SCN POSIT.
0012        MOVB @CHARA,R1   CHAR CODE INTO R1
0013 LOOP   BLWP @VSBW       BRANCH TO SCREEN WRITE
0014        DEC  R0          DECREASE R0 BY ONE
0015        JNE  LOOP        EQUAL TO ZERO? GOTO LOOP.
0016        MOV  @RETURN,R11 RETURN ADDR INTO MY R11.
0017        RT
0018       END
```

In this LISTING the change is simple enough. The MOV instruction on LINE 12 is changed to MOVB. This time only the most significant byte of CHARA is shifted into the most significant byte of R1. Take note that any DATA existing in the least significant or right byte of R1 remains intact.

I hope that both of these methods had already been tried by you.

A QUICK PATCH.
****

After Assembly of your programme

inor patches can be made to the BJECT CODE. If the alterations to e made are extensive then I would uggest that the alterations be made o the SOURCE CODE and not alter the BJECT CODE. Alterations can not be ade to compressed OBJECT CODE.

BJECT CODE is saved to disc as a IS/FIX 80 file. As a result atching of OBJECT CODE can only be one on the EDITOR in the EDITOR SSEMBLER package or with BEAX. The .I. writer or FUNNEL WRITER DITORS only save in DIS/VAR 80.

elect EDIT then LOAD and when rompted enter the filename;
EA/P/1OBJ"

nter 2 to EDIT the file;

nce the column width in our agazine is 35 characters I have had o show each line of code over veral magazine line in this rticle.

he first line is as below;

0044 (8 BLANKS)
0000B0000B0041A0004A0024BC80BC0000B
2E0C0004B02007F365F 0001

f you would like to take the time nd count the characters above you ill find that there are in fact 80 n total including blanks ie DIS/FIX 0.

programme consists of variable ize records. Each record is a umber of character tags followed by o to two fields. Pages 238/241 of he E/A manual describe these haracter tags fully.

0044 The first Character, 0 ndicates the beginning of the rogramme. The next 4 characters re the programme length. These our characters are the first field ssociated with the TAG 0. The econd field of the TAG 0 is 8 lanks. If we had wanted to give he programme a name the DIRECTIVE DT could have been used. The name iven would appear in the space here the 8 blanks are.

he next TAG is A followed by 0000. tells the loader that the llowing DATA is to be loaded at ddress 0000. The next tag B is used to indicate absolute DATA, this is the DATA loaded at address 0000. Check this on the SOURCE LISTING. The next TAG B again indicates absolute DATA 0041, this is loaded at the address following the previous data since no new address was specified. (This is the data for character code of A).

The next TAG A it's associated value 0004,the next A and it's value 0024 sets aside the contiguous block of memory for our WORKSPACE.
A0024 tells the loader to place the data following the next tag B at address 0024. This leaves the >20 or decimal 32 byte block from A004 to A024 free for the workspace.

By referring to the SOURCE LISTING and reading in the E/A manual what each TAG indicates the programme can be followed. When the TAG 8 is encountered. In this line it occurs after B0200. The loader then looks for the next four characters which are the check sum. This check sum is compared against the programme data entered. If the check sum is correct then loading continues. If the check sum is incorrect loading stops and an error indicated. How the check sum is evaluated at this stage I do not know, if you do then please drop a line to the EDITOR.
The check sum is followed by the character TAG F which indicates the end of a record, this TAG requires no field.
The remainder of the record is padded out with blanks then a sequence number is added. The sequence number above is 0001.

The remainder of the programme will not be discussed here but I suggest that you work through the programme referring to the E/A manual to see what the TAGS indicate.

Now back to the task at hand, altering the code. Change the value 0041 in the first line of code to 0042. That is, alter the character code data from A to B. Use function back to get to the EDIT menu, select OPTION 3 SAVE. When the prompt DIS/VAR 80 Y/N is shown then select N. Then save the CODE back to disc. When completed return to the E/A main menu.
Select OPTION 3 LOAD AND RUN.
Enter the Filename just used to save

the CODE and press enter. Loading will commence and then halt with a CHECK SUM ERROR.

Return to the E/A main menu, select EDIT then LOAD from the EDIT menu and reload the code. Now on line 0001 shown above change the TAG 8 to a 7. Resave and then LOAD and RUN. This time the screen fills with the character B.

Try some more changes, one which is simple is to change the name START to some other name. This occurs on the third line of code. After loading the programme alter START to APART. Don't forget to change the 7 in 7FAA0F to an 8. Resave and then LOAD and RUN the program , of course the start name is now APART.

TEXT AND KEY SCAN.
****
The following programme shows how to get TEXT onto screen. So that the TEXT can be read I have included a "PRESS ANY KEY TO CONTINUE" routine. After the first KEY PRESS the text is cleared from the screen and re-appears. The second KEY PRESS returns control to the E/A module. The second new feature is the use of BRANCH and LINK BL. This allows us to us small sub programmes at will without the need to retype them or consume valuable memory spare. KSCAN the keyboard UTILITY is show here used in the simplest way. It is used here similar to the CALL KEY is used to allow programme flow to continue after ANY key is pressed. Also included is a routine to clear the screen. Both of these routines will be used over and over again as we develop more complex programmes. As a matter of fact it isn't a bad idea one night when you are far removed from your computer to jot down all the things you can write as short routines. Then later write them and save to disc. When programming in FORTH it is essential to keep the STACK as clean as possible when leaving a WORD. This philosophy I try to maintain in ASSEMBLY. When writting a sub routine try not to have too much garbage left when you exit the routine.

This programme I hope will lead us to our cursor routine, then onto validating the data keyed in.

SEE PROG. 1 NEXT PAGE

This next programme is an extension of the previous programme. This programme shows how data in REGISTERS can be manipulated to get the data required for VMBW ready.

SEE PROG. 2 NEXT PAGE

CONCLUSION.
****
That is all there is time for until next month. In the mean time I strongly suggest that you read chapter 8 of Molesworth. If you have difficulty with the number crunching on page 69 don't worry to much. I will deal with that next month. Also from the E/A Manual;
ADD IMMEDIATE PAGE 85
BRANCH AND LINK PAGE 108
TAGS PAGES 307/309
VMBW PAGE 248

Until next month

TI-WRITER TIP

From the *NORTHWEST OHIO USERS GROUP* comes this tip. To avoid a BUFFER FULL notice using TI-WRITER (or FUNLWRITER), you should save the file and then use the SD (Show Directory) command to see the file size. Since the BUFFER is full at 92 sectors, you can see right away where you are. Thanks NW O UG!

OH OH OH OH

Errors can creep into any magazine, and that includes ours! In last month's issue, in the article by Allen Wright *HINTS AND TIPS*, line 5120 of the subprogramme should be:

```
5120 IF EOF(2)=0 THEN 5030
5125 CLOSE #2
```

PROGRAM 1

```
0001 * THIRD TUTORIAL PROGRAMME
0002 * 7-10-86 FILENAME SOURCE=EA/PROG/3
0003 *                 OBJECT=EA/P/3OBJ
0004        DEF  START                    PROGRAMME NAME
0005        REF  VSBW,VMBW,KSCAN          REFERENCES
0006 MESS1  TEXT 'HUNTER VALLEY 99'ERS'   FIRST MESSAGE
0007 MESS2  TEXT 'PRESS ANY KEY TO CONT.' SECOND MESSAGE
0008 STATUS EQU  >837C                    STATUS REG EQUATE
0009 BDSTAT EQU  >8374                    KEYBOARD DEVICE
0010 KEYVAL EQU  >8375                    KEY VALUE RETURNED
0011 ANYKEY BYTE >20                      ANY KEY PRESS MASK
0012 RETURN DATA >0000                    SPACE TO SAVE R11
0013 FINPOS DATA >02DF                    POS. TO FIN BLANKS
0014 WSRES  BSS  >20                      SPACE FOR MY WORKSPACE
0015 START  MOV  R11,@RETURN              SAVE RETURN ADDRESS
0016        LWPI WSRES                    CREATE MY W/SPACE
0017        LI   R0,>C6                   SCREEN ADDRESS
0018        LI   R1,MESS1                 ADDRESS OF TEXT
0019        LI   R2,20                    NUMB CHARS TO SCREEN
0020        BLWP @VMBW                    WRITE THEM!
0021        BL   @CONT1                   BRANCH TO CONTINUE!
0022        BL   @CLEAR1                  BRANCH TO CLEAR SCN.
0023        LI   R0,>E6                   NEW SCN POS FOR MESS
0024        LI   R1,MESS1                 ADDRESS OF MESS1
0025        LI   R2,20                    NUMB CHAR TO SCREEN
0026        BLWP @VMBW                    WRITE THEM!
0027        BL   @CONT1                   BRANCH TO CONTINUE!
0028        MOV  @RETURN,R11              RETURN ADDR TO MY R11
0029        RT                            BACK TO EA.
0030 CONT1  LI   R0,>2E6                  SCREEN ADDRESS
0031        LI   R1,MESS2                 ADDRESS OF TEXT
0032        LI   R2,22                    NUMBER OF CHARS TO SCREEN
0033        BLWP @VMBW                    WRITE THEM!
0034        CLR  @STATUS                  CLEAR STATUS REGISTER
0035 CONT2  BLWP @KSCAN                   BRANCH TO KSCAN
0036        CB   @ANYKEY,@STATUS          BIT 2 SET?
0037        JNE  CONT2                    NO? GO BACK TO KSCAN
0038        LI   R0,>2FF                  SCREEN ADDR. TO START CLR
0039        LI   R1,>20                   BLANK DATA
0040 CONT3  BLWP @VSBW                    WRITE BLANK
0041        DEC  R0                       DECREMENT R0
0042        C    R0,@FINPOS               AT FINISH POSITION?
0043        JNE  CONT3                    NO? BACK TO CONT3
0044        CLR  @STATUS                  CLEAR STATUS AFTER USE
0045        RT                            BACK TO MAIN PROGRAMME
0046 CLEAR1 LI   R0,>2FF                  LAST SCREEN POSITION
0047        LI   R1,>20                   BLANK CHAR DATA
0048        LI   R2,0                     PUT ZERO INTO R2
0049 CLEAR2 BLWP @VSBW                    WRITE BLANK!
0050        DEC  R0                       DECREMENT SCREEN POS.
0051        C    R0,R2                    IS IT AT ZERO?
0052        JNE  CLEAR2                   NO? BACK TO CLEAR2
0052        RT                            RETURN TO MAIN PROGRAMME
0054        END
```

PROGRAM 2

```
0001 * THIRD TUTORIAL PROGRAMME
0002 * 7-10-86 FILENAME SOURCE=EA/PROG/3A
0003 *                 OBJECT=EA/P/3AOBJ
0004        DEF  START                      PROGRAMME NAME
0005        REF  VSBW,VMBW,KSCAN            REFERENCES
0006 MESS1  TEXT 'HUNTER VALLEY 99'ERS'  FIRST MESSAGE
0007 MESS2  TEXT 'PRESS ANY KEY TO CONT.'SECOND MESSAGE
0008 STATUS EQU  >837C                      STATUS REG EQUATE
0009 BDSTAT EQU  >8374                      KEYBOARD DEVICE
0010 KEYVAL EQU  >8375                      KEY VALUE RETURNED
0012 ANYKEY BYTE >20                        ANY KEY PRESS MASK
0013 RETURN DATA >0000                      SPACE TO SAVE R11
0014 FINPOS DATA >02DF                      POS. TO FIN BLANKS
0015 WSREG  BSS  >20                        SPACE FOR MY WORKSPACE
0016 START  MOV  R11,@RETURN                SAVE RETURN ADDRESS
0017        LWPI WSREG                      CREATE MY W/SPACE
0018        LI   R4,>2C6                    LAST SCN POS 4 TEXT
0019        LI   R5,>06                     FIRST SCN POS 4 TEXT
0020 SCRN1  LI   R1,MESS1                   ADDRESS OF TEXT
0021        LI   R2,20                      NUMB CHARS TO SCREEN
0022        MOV  R5,R0                      LOAD R0 FOR VMBW
0023        BLWP @VMBW                      WRITE THEM!
0024        BL   @CONT1                     BRANCH TO CONTINUE!
0025        BL   @CLEAR1                    BRANCH TO CLEAR SCN.
0026        AI   R5,>0020                   NEW SCN POS FOR MESS
0027        C    R5,R4                      AT LAST POS YET?
0028        JNE  SCRN1                      NO! GO BACK WRITE AGAIN
0029        MOV  @RETURN,R11                RETURN ADDR TO MY R11
0030        RT                              BACK TO EA.
0031 CONT1  LI   R0,>2C6                    SCREEN ADDRESS
0032        LI   R1,MESS2                   ADDRESS OF TEXT
0033        LI   R2,22                      NUMBER OF CHARS TO SCREEN
0034        BLWP @VMBW                      WRITE THEM!
0035        CLR  @STATUS                    CLEAR STATUS REGISTER
0036 CONT2  BLWP @KSCAN                     BRANCH TO KSCAN
0037        CB   @ANYKEY,@STATUS            BIT 2 SET?
0038        JNE  CONT2                      NO? BACK TO KSCAN
0039        LI   R0,>2FF                    SCREEN ADDR. TO START CLR
0040        LI   R1,>20                     BLANK DATA
0042 CONT3  BLWP @VSBW                      WRITE BLANK
0043        DEC  R0                         DECREMENT R0
0044        C    R0,@FINPOS                 AT FINISH POSITION?
0045        JNE  CONT3                      NO? BACK TO CONT3
0046        CLR  @STATUS                    CLEAR STATUS AFTER USE
0047        RT                              BACK TO MAIN PROGRAMME
0048 CLEAR1 LI   R0,>2FF                    LAST SCREEN POSITION
0049        LI   R1,>20                     BLANK CHAR DATA
0050        LI   R2,0                       PUT ZERO INTO R2
0051 CLEAR2 BLWP @VSBW                      WRITE BLANK!
0052        DEC  R0                         DECREMENT SCREEN POS.
0053        C    R0,R2                      IS IT AT ZERO?
0054        JNE  CLEAR2                     NO? BACK TO CLEAR2
0055        RT                              RETURN TO MAIN PROGRAMME
0056        END
```

H.V. CURSOR

BY BRIAN RUTHERFORD

After using a programme with its own uniquely shaped cursor, I thought it would be nice to design one for the club, and as the people at the last meeting liked my design, I was asked why wasn't it in the magazine? The answer to that is if I had put it in the last magazine I would not have had any thing to put in this months! But waffling aside, here is the programme that does it for you. Memory expansion is needed unfortunately.

```
1 CALL INIT::CALL LOAD(8196,
63,248)::CALL LOAD(16376,67,
85,82,83,79,82,48,8)::CALL
LOAD(12288,36,60,36,36,129,6
6,36,24)

2 CALL LOAD(12296,2,0,3,240,
2,1,48,0,2,2,0,8,4,32,32,36,
4,91)::CALL LINK("CURSOR")
```

You may also wish to design your own cursor, and to do that you just change the eight values in the CALL LOAD(12288,.....) statement. The character pattern below shows the relationship of the pattern to the values in that CALL LOAD.

	HEX	DEC
¦ ¦ ¦x¦ ¦ ¦x¦ ¦ ¦	>24	36
¦ ¦ ¦x¦x¦x¦x¦ ¦ ¦	>3C	60
¦ ¦ ¦x¦ ¦ ¦x¦ ¦ ¦	>24	36
¦ ¦ ¦x¦ ¦ ¦x¦ ¦ ¦	>24	36
¦x¦ ¦ ¦ ¦ ¦ ¦ ¦x¦	>81	129
¦ ¦x¦ ¦ ¦ ¦ ¦x¦ ¦	>42	66
¦ ¦ ¦x¦ ¦ ¦x¦ ¦ ¦	>24	36
¦ ¦ ¦ ¦x¦x¦ ¦ ¦ ¦	>18	24

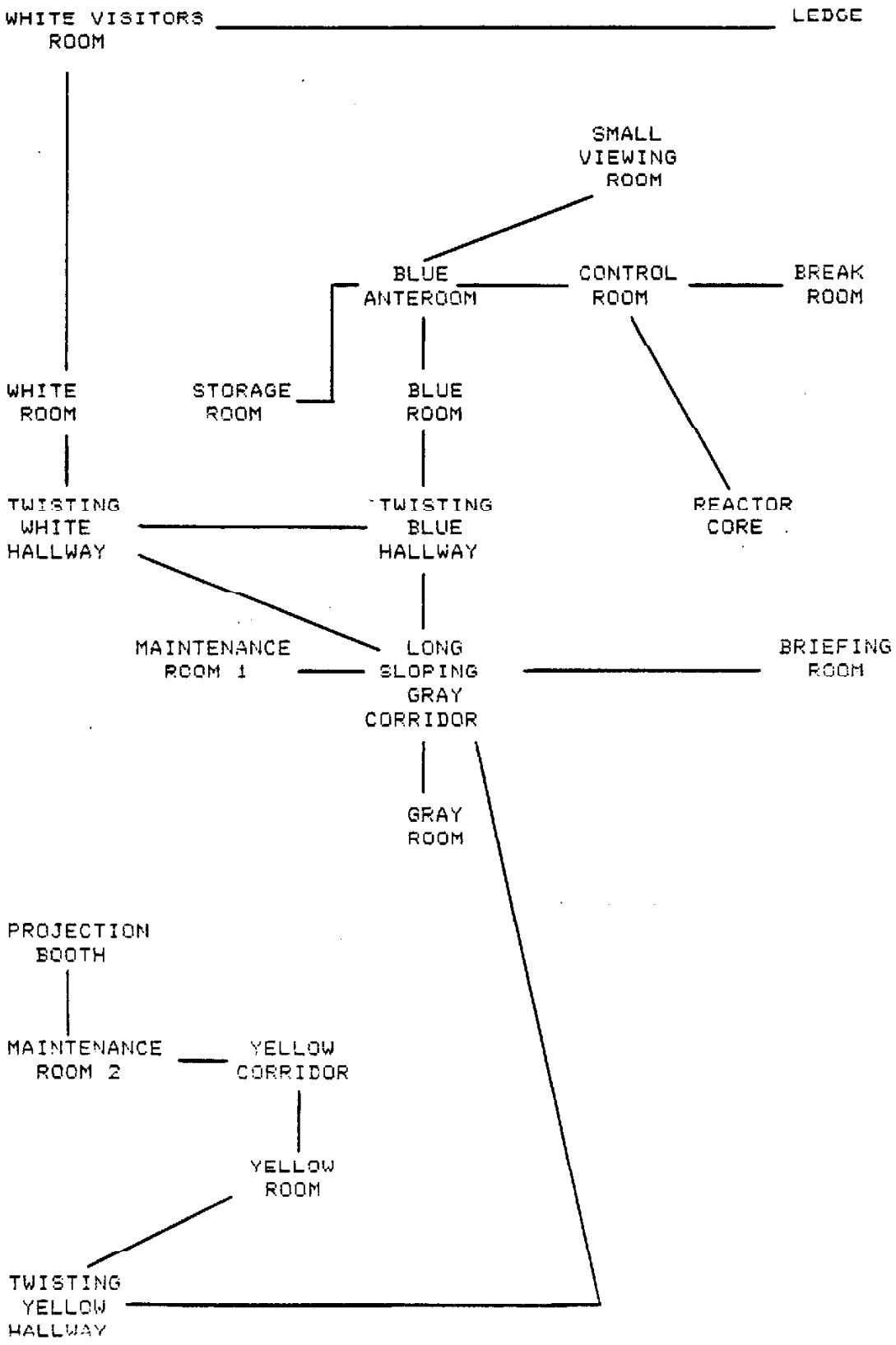
So to design your own cursor work out the HEX code for each line the same as for a XB or BASIC programme and change each two digit HEX number for each line to a decimal number.

Now for the people that are interested let's have a little look at how this programme works.

CALL LOAD(8196,63,248)

Decimal 8196 = >2004 and the value at that address points to the last free address in low memory. The value that is being loaded into that address is:- 63 = >3F and 248 = >F8, put them both together you get >3FF8. Which means that the value stored at address >2004 is >3FF8 and >3FF8 is part of the REF/DEF table.

MISSION IMPOSSIBLE



ADVENTURERS COLUMN

WITH RODNEY GAINSFORD

HI!. Weel at last it has been done! Two Members in the CLUB have SOLVED H.H.G.T.T.G. They are Geoffrey Gray and Yours truly. They both deserve a pat on the back. Unfortunately no hints are available as yet because of a time shortage. This month along with our map we have a point summary for ZORK plus! some hints on ZORK 111. Some general hints on solving adventures are also included.

Please send any maps or hints to;
Rodney G.
C/O. The Secretary.
H.V.99'ers.
6 ARCOT PLACE
TARRO. 2322

ZORK POINT SUMMARY.
****
These items have value for touch and for Trophy case, the number after the object is the combined score.

Beautiful Painting	10
Jewel encrusted egg	10
Ivory Torch	20
Gold Coffin	25
Egyptian sceptre	10
Pot of Gold	20
Crystal Skull	20
Jewelled Scarab	10
Large Emerald	15
Platinum Bar	15
Trunk of Jewels	20
Crystal Trident	15
Jade Figurine	10
Sapphire Bracelet	10
Huge Diamond	20
Bag of Coins	15
Silver Chalice	15
Clockwork Canary	10
Brass Bauble	2

There are also points for

Getting into the House	10
Getting into the cellar	25
getting past the troll	5
Getting to the torch room	13
Getting to the treasure room	25

TACKLING an ADVENTURE!!!!!
****
The first thing you do is try EVERYTHING and find out ALL the ways you can kill yourself. Once you have done this you map as much as you can. Then you start to play. When ever you solve a puzzle save the game and keep an accurate map nearby. When you find a puzzle you can't solve try things that are utterly useless to your mind, remember everything has a use.

ZORK 111.
****
* be trusting.
* Indicator 1 useless.
* 2 rub table get grue repellant.
* 3 Get torch rub table drop torch.
* 4 Save game Rub table.
* The chest will carry the lantern safely.
* Break the beam.
* Don't drop book in slot.
* Feed the old man.
* Save game. Take lamp into lake.
* Don't kill shadowy figure.

ZORK 11.
****
* You can't stop the wizard.
* In riddle room say "well".
* Orange cake - bang.
* Green cake - shrink.
* Red cake - ???.
* Blue cake - ????.
* Tell Demon to kill wizard.

ROD G.

HINTS AND TIPS

is article, by Ed York, iginally appeared in the CIN-DAY WSLETTER, magazine of the Users oup of the Cincinnati-Dayton ea.

NT #1
SPLAY AT, used in XB, will use the ion, comma & semi-colon just like PRINT statement. This line will e you some idea of how it works:

DISPLAY AT(5,1):"THIS":"WILL", IVE":"YOU","SOME":;:"IDEA OF ": W":"IT","WORKS."

the Reference Manual under PRINT details.

NT #2
Washington (D.C.) Users Group nted out that in XB, you can use statement RUN "CS1" instead of CS1 and RUN to load and run a gram (with a disk system RUN Kx.program name" is used and lained in the manual but RUN 1" is not mentioned).

NT #3
a way to save memory you can use non-numeric character in place of umeric character. The difference that a non-numeric character only s 2 bytes where a numeric racter uses 4 bytes. An example non-numeric character would be an whereas an example of a numeric racter would be a 1. The llowing example illustrates this nt:

```
CALL CLEAR
@=1
FOR A=@ TO 100
PRINT A
NEXT @
```

you have a long program where you d extra memory, then you can ain all that would be possible in s respect. Also, if you need to e memory, try shortening the iable names to a single letter h a maximum of two letters. ally if you still need more ory try removing all REM tements.

HINT #4
Have you ever wanted to round off numbers and did not know how to do it in a program? Well, it is easy once you know the secret. Here's how! The following example will illustrate the process:

```
100 CALL CLEAR
110 A=123.4567891
120 PRINT A
130 A=INT(A*100+.5)/100
140 PRINT A
```

When this program is RUN you should get the following results:

```
123.4567891
123.46
```

If you want to round off to three places then use 1000 instead of 100 and for four places use 10000 instead of 100. Try it!

HINT #5
Have you ever wanted to generate random numbers, say from -10 to 10? Well, it is also an easy matter to accomplish once you know the secret. Here is an example:

```
100 CALL CLEAR
110 FOR A=1 TO 15
120
B=(-1)^(INT(11*RND))*(INT(11*RND))
130 PRINT B;
140 NEXT A
```

When this is RUN, here is a sample of what you might expect:

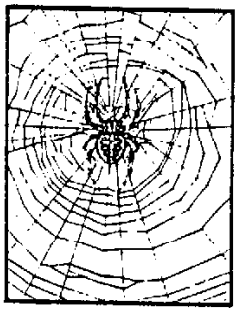
```
-4 6 0 5 3 -9 1 4 9 -6 -5 -10 7 -2
```

All you have to do to specify the range of random numbers is to change the 11 to any number you desire. To change the number of numbers that are to be generated, change the 15 to the amount you want generated.

HINT #6
Are there times when you would like to print something on the screen without having it scroll on the screen? Well you can and it is easier than you think! Here is how it can be done without XB. First decide where you want to print the message. You will need the row and column coordinates. Second, you will need the message or statement. Third, you will need the subroutine to accomplish this task. Here is an example:

```
100 CALL CLEAR
110 R=12
120 C=2
130   A$="TEXAS   INSTRUMENTS   HOME
COMPUTER"
140 GOSUB 160
150 GOTO 150
160 FOR A=1 TO LEN(A$)
170 B=ASC(SEG$(A$,A,1))
180 CALL HCHAR(R,C+A,B)
190 NEXT A
200 RETURN
```

This should clear the screen and print the message right in the middle of the screen. The subroutine can be used as many times as you wish. You may change the CALL HCHAR to a CALL VCHAR statement and change the value of R from 12 to 1 and the value of C from 2 to 16. Try experimenting and you can make it print backwards and diagonally. Have fun!



ENTOMOLOGY CORNER

Spring has sprung and the days are getting longer. Just as well as we have a bunch of rainforest tree seedlings to plant out. Have to keep the funnelwebs happy and feeling at home somehow. On the computing front from time to time minor changes and updates are made to Funlwriter. I'm going to have to think of a better name for that program now that it has outgrown its original bounds by so much. The latest version of DM1000 (Vn 3.3) was received from Bob Boone in Ottawa and promptly interfaced to F'Wr as part of the current issue. Bob tells me the next version of DM1000 is projected for release at the Chicago TI-Fair towards the end of the year. The complete DM1000 package is available from the club library in the usual way.

The C99B adapter file has also been revised so that it works with both the existing Vn 2.0 of c99 and the upcoming Vn 2.1 as well. There is also a C99PFI file modified by re-assembly to bypass unloading of the E/A utilities from GROM in all circumstances. This allows c99 program files to be run from F'Wr and E/A, which is how we usually use it. This may well be the last issue of the c99 compiler for the 99/4a as Clint Pulley is now working with a sample of the new Myarc Geneve computer. Clint sounds very happy with the new machine, with its faster operation of TI-Writer, and real single-key cursor operation on the IBM type keyboard. The prototype he has contains only 256K of CPU RAM and 64K of VDP RAM with more to fit in when a custom gate array chip is ready. Clint thinks it is a real alternative to the Amiga. I am wondering whether it will survive the cost increase in memory chips that has occurred in the USA now that the US industry has forced massive price increases on Japanese chip exports to the US. I'd love to have one but can't shake off that feeling that it isn't a realistic way to go. Of course I'd even swallow my distaste for Intel machine architecture if one of TI's PC or AT imitations were to fall off the back of a truck passing Funnelweb Farm, but when it comes to laying out the cash for an new system, we will have to think long and hard. Also I just recently saw an Intel ad for a new version of the 80286 processor that powers the IBM PC/AT boasting that this new one could do a context switch on interrupt in 20 microseconds. If they keep working at it they may come up with one that beats the old 9900 in that department.

Now that C99B has been generalized it seems safe to omit the C99C/D/E files from the distribution disk as the doubts about version dependence have been removed. The space freed up on the distribution disk will fill up with other utilities in time. The first of these is an alternative version

LOADA of the LOAD program. This is more adapted for use as a general purpose loader, going straight to the User's List screen which provides 18 choices, 16 of which are available for user entry. This will probably only be useful to people with DSDD drives or large RAMdisk cards to fit anywhere near the possible number of utility programs on. Talking of RAMdisks we hope in the relatively near future to have a Horizon RAMdisk installed. And thanks to Pete Smith we are all now fully aware of what NOT to do with this device. The most effective bit of teaching in years. Another no-no we were just painfully reminded of is turning off the power to a second or third external drive while it is still connected up to the computer. Attempts to access or even just catalog the first drive may then destroy disks. This happened to our first Newsletter Editor, Steve Taylor and I think it has just struck here too with disastrous results to the only copy of the LOADA source. As a later note a bug was discovered in the first issue of LOADA, and we had to reconstruct the whole thing.

There does appear to be one residual bug in F'Wr that I am aware of but have not yet been able to fix. This is that very same one I mentioned last time as reported by Woody Wilson from San Diego, the one that prompted several bug fixes and polishings. I never could reproduce the original, and assumed I had fixed it in the process. As you well know, program bugs just go into hiding unless explicitly squashed, and this one was no different. Brian Rutherford unearthed it again the other day, when he Listed an assembly to his printer and found line feeds weren't being sent to the printer. So where was it hiding ? In the PIO routine. Why didn't we find it before ? Because we use a TI-99 printer always set to RS232.BA=4800, and the bug only shows up with PIO. This is very strange because the only obvious relevant difference between F'Wr and E/A is that E/A opens the LIST device from GPL while F'Wr uses its normal DSRLNK routine. The trouble with that as an explanation is that F'Wr's DSRLNK is identical to TI-Writer's and that works perfectly with PIO, otherwise we would have heard the screams long ago. I am going to have to dig deeper. As assembly programmers already know from the E/A manual the RS232 card is an unruly beast which doesn't always follow the Tech Manual specs - and it runs deeper than just not preserving the GROM address. For the time being if you have a parallel printer, the work-around for Listing short to medium length assemblies is to list to disk and then use the Editor to print out the file. For long files it may be better to reset the printer to handle the absence of line feeds.

William has been busy too, mostly engaging in conversations with disk controller chips. His DISKHACKER program (part 1) was demonstrated at the last Club meeting and is now being sent out as a fairware release. For those who weren't there it analyses sector patterns on a disk, track by track, and presents the analysed information for your inspection. It starts off where MG's Advanced Diagnostics leaves off in a trail of deception, refusing to present perfectly good but non-TI-standard tracks, let alone present any of it in analysed form. The first release works with TI Controllers, but further programs in the series will work with Corcomp and Myarc controllers as well, all from the one program. It's not that he doesn't know how to make Corcomp and Myarc controllers jump throuh hoops already, but just that he can't face rewriting an existing working program when there are new worlds to conquer. It is an interesting exercise to inspect the details of various protection schemes used on commercial disks. I have been embargoed from even hinting at the details of the protection methods used on more recent commercial disks until after later parts of DISKHACKER are released. Suffice to say that DH'r is a sector analyser.

This is all a lesson in the futility of disk protection as practised. All it does is make the programs so treated inconvenient and inflexible in use, and puts the serious user to the bother of

removing the protection to make a backup. As an engineer quoted in a fascinating recent article in the IEEE Spectrum on protection methods said -- "I regard disk protection as a bug, and when I find a bug I fix it". A disk is a fragile enough form of archival storage, let alone in regular use where it may encounter a malfunctioning computer, or even just one with external disk power not switched on. Our personal policy is to refuse to buy any program on a protected disk. In practice it doesn't cause any hardship as none of the programs we have seen in this format are such that they can't be lived without. Let's take as a recent example the Miller's Graphics DISkASSEMBLER. From what we've seen it looks like a very good job has been done on it. Is the idea original ? Well, not really because it has been obvious for some while, and such programs are found on other systems. Will and I discussed the idea of writing a program to disassemble from disk files, recognising that it would have been convenient to have had earlier. With no pressing need apparent, and much work to be done on other things the idea was shelved. We weren't the only ones either, because there is a fairware program Universal Disassembler written in FORTH available from the Club library. Apart from initial inspection we haven't had occasion to use that one either, but it looks a good program too. DkA came our way as the protected disk with a request to make a backup for the owner. William can't resist a challenge like that, and it took him precisely one day to clone the disk using only programs he had written himself, apart from using the DkA on its own loader. Since then we have had reports that a cracked version is circulating in the USA. This news seems quite believable because it took Will only one day more to reduce DkA to E/A program files. And if a 16 year old high school kid in Newcastle can do it that quickly, then how many more must there be across the whole US of A who can do the same ? Now comes the silly side of the protectionist's paranoia. I decided that DkA looked like a worthwhile program to have and use, and that if we used it, it would be from a genuine original. So I wrote to Miller's requesting price quotation on an unprotected version of the program, and received a rather prissy reply to the effect that they didn't sell it unprotected. So there was one sale lost to protection, and I'm sure the refusal won't have helped curb piracy one little bit even if Miller's wouldn't trust us to respect their copyright, as the report of a pirate version came indirectly via Europe and it must already have been circulating at the time. So I haven't looked at it enough to be able to review it. I think it is also true now that many of the best and most distinctive programs for the TI are not protected, and come as fairware for that matter too. And outside the TI world you only have to look at the quality, value, and success of Turbo-Pascal along with Borland's later products.

All writers of disassemblers seem to have one thing in common. share holdings in suppliers of printer paper and ribbons. Only our trusty old DIS/ASS permits 4 column printout from a diskfile written as 28 column screen image, using the COLIST utility program. While the thought of protected software brings the fragility of disks to mind I should remind you of TI's advice in their Software Development Handbook --- always maintain 3 copies of important program or source files that you are working on. Why not just two ? If there are three you have no excuse for ever having all your copies in the machine at once, even for updating purposes. Also never update both backups at the same time without checking the validity of the initial copy first.

Now for some impressions of the TI-99 scene. Micropendium (that doesn't sound like a Texas size appendage) continues it's timely appearances. It carries a range of purely TI-99 relevant material of interest to all levels of TI-99 user. Just don't take too seriously the first letter to the editor on any particular technical topic until the later mail has come in. All in all well recommended for a subscription, and far superior to anything that existed

in the days when TI still produced the machine. Talking of that, see elsewhere in this issue for a short item on the fraudulent career of 99er magazine and its various aliases, and how best to go about getting your money back if they still owe you some.

The Smart Programmer has now reappeared with similar content to that of old, and promise of timelier publication schedules. As it has higher technical pretenses than Micropendium I will judge it accordingly. The first two of the new series are a mixed bag. Previous issues made a big deal of publishing details of the TI ROM code. In fact very little detail appeared of relevance to the assembly programmer in the various maps, that couldn't be found out very quickly, and it never really was much more use than the E/A and Technical Manuals. A handy reference on occasion but that's about all. If you want to see a real exegesis of the console internals see the TI-Intern book from Germany. Our copy came from Bernie Elsner in Perth. That will will show you how it should be done. I'm sure the Miller's stable have an equally thorough knowledge, but are not prepared to share it. That's understandable for commercial reasons but I do object to all the hype generated to convince people that the SP is really telling them something. This gap between between hype and reality is characteristic of all the Miller's output, at least on a hard-nosed engineer's judgment of the real utility of the products. Which is not to say that some really high class work hasn't gone into MG's program and gadget output.

The new series carries on the tradition. The second issue contains a dissection of the MINIMEM module which says nothing more than is in the manual possessed by every owner of that module. Big deal ! The other articles are of more interest. The one on disk/cassette load and transfer utilities for machine code program files presents useful programs. They are about on a par with Will's beginning efforts in this vein, and at least they inspired him to redo the job to Funnelweb Farm standards incorporating his own text-mode machine code file-name editor that he had written for DISKHACKER. It is now on issue as fairware. Another article on the TMS-9995 processor was of interest in the "what might and should have been" category (and may yet be if Myarc's machine gets off the ground). Somehow it missed talking about the most significant advances of the 9995 over the 9900 though.

The high point of the issue was a DSRLNK/GPLLNK that used the console routines so that it was shorter even if slower than the usual assembly routine. A quick glance shows 4 bytes could be removed from the GPLLNK, and there must be something I haven't quite cottoned onto in the DSRLNK which allows multiple varied use under error conditions. Not surprisingly it uses indexed addressing to support GROM paging, a fairly reprehensible omission from Funlwriter perpetrated for reasons of code squeezing, but then I don't have any interest in flogging Gramkrackers and always advise people to spend their money on an honest straightforward RAMdisk instead. The XML used (the eXecute Machine Language escape from the clutches of GPL) is the one I remarked on in an earlier article in this series, a freak accidental in a data table near the end of GROM #0. I never would have used it, because I could have had no absolute confidence that this table was precisely the same in all consoles, let alone residing at the same absolute address. The method of searching GROM #0 for a regular XML is sounder, in the absence of complete knowledge. Now what this article did do that very few individual owners are in a position to do is claim that the routines, which used absolute addresses in the console including this XML, would work on all models of 99/4a produced.

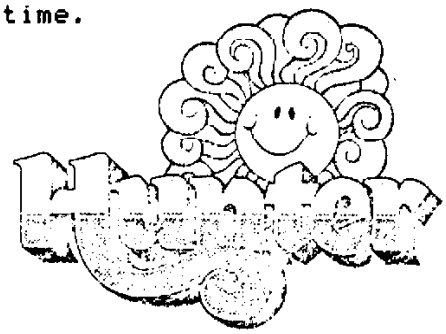
This little gem made the second issue very much better than the first which was mostly a plug for MG's hardware offering - the Gramkracker (a pun that doesn't translate from the American). I don't think I would lay out any

money for this gadget as it doesn't seem to do anything that we want to do that wouldn't be done better with Funlwriter and a RAMdisk. I have heard F'wr described as the "poor man's gramkracker". Combine that with the availability of programs to dump GROM only cartridges and run them from memory expansion with their own GPL interpreter in RAM as well, and there isn't much call for the MG device unless you have money to spare. My advice is to go for a Horizon or Myarc RAMdisk instead. The main item of interest in the first issue was a FORTH article, biased towards Wycove Forth. I have heard from several sources that the Wycove version is better than TI's, being smaller and faster. The Wycove is a commercial product and has the distinction of being one of the very few machine code programs available from a source other than TI before they orphaned the 99/4a, and probably the best too. I can't comment on the relative merits from personal experience because after a very short flirtation with TI-Forth I rapidly decided that the elegant TMS9900 assembler code was easier to use. When we get back to high level languages it will be with c99, and Pascal if I can ever get the p-system working on DSDD disks.

We have received from France a copy of the TI-Writer disk sold there by TI. This is comes up as Version 2.0 on the Editor's end of file message and the Formatter is dated 1983. The disk fully supports the foreign language capabilities. With the version sold here selecting one of the module's language entries would bring up the GROM resident parts (selection screen and SD) in the language selected, but the Editor and Formatter internally were still in English, or what passes for it in word processor prompt lines. In the continental version there are three extra sets of files, a character file for each language with true lower case CHARA1-CHARG1, and a set for each of the Editor and Formatter which contain the commands and prompts in each language. The bug in Recover Edit that was introduced with the *fix 1 update of Vn 1.0 issued to User Groups by TI has been repaired. I'm not sure what the copyright status of Vn 2.0 is. I have that feeling that if TI had ever imagined that something like F'Wr was possible, which they clearly didn't, that they might never have released *fix 1 of Vn 1.0 to the public domain. I suspect TI don't ever want to be reminded of the existence of the 99/4a but they are a large corporation, from the land of litigation, that can afford lots of lawyers.

There is an incompatibility between the Editor versions also in that tab records written by Vn 1.0 are not recognised as such by Vn 2.0 but print as a line of special characters. The character files have an extra set of entries above #128 for extra screen characters eg for French small characters with accent marks. I don't know whether they cater for Canadian French. In the absence of a manual I'm not sure how these are entered from the keyboard. That all can be lived with but the reverse problem is more serious - the tab record written by Vn 2.0 locks up Vn 1.0 on the way in. I temporarily lost this file until I went back to TI-Writer and Vn 2.0 and used PF to rewrite the file again without a tab record. As a curiosity the English language version is British rather than American. Seeing as we use # often and pounds sterling but rarely, the American version is more appropriate here in Australia.

Other recent arrivals in the fairware line include PRBASE Vn 2.0 which looks like becoming the database program of choice, and the RAG macro-assembler. Haven't had a chance to look at either of these in detail yet, but Al now has copies in the library. That's all the news, gossip, opinion, libel and so forth from Funnelweb Farm for this issue, so bye now until next time.



LIBRARIANS NOTES

HI 99'ers,

Did you know that a Monthly Disk is available and as far as possible has the same content as the Month's Cassette.
Some programs were for Disk but can be adapted easily for cassette and where one prog. runs another,you get into the programme and alter the line #xxx which may have -"RUN DSK1.pgm name." to "RUN CS1" then your Programme will Load and Run. XB. This only applies to XB and not Basic programmes .

We have Programmes Suitable for 32k Cassette for those who have no Disk Drive but have the 32k in their Console and want to make use of the Extra memory.

Monthly Software.

This is not a total Review of all Disk and Cassette Software August thro' December but an overview to explain that some of the programmes on these disks are in 3 or more parts and as such have one part on each month but not necessary on the following Months.

August.

Includes 2 multi part Progs.Both are the Instructions for:-
(a)Typing Tutor 3 parts.(TI-PING/I)
(b)Out on a Limb 5 Parts.(LIMB/I)

October.

(a)TI-PING/2 The Learn Typing Tutor Practice section. (b)LIMB/2. and LIMB-D.Out on a Limb parts 2 and 3(data you can add to)
SUP-S/EDIT.Super Sprite Editor to let you design,save and use you own sprites for those Programs you are getting ready for the HV News.

SPACECOM.
STARGRULES.Rules for Game below.
STAR-GUARD.
SKYWATCH.Give visible objects,the compass bearing and altitude for a given time and dates from a master table.

TACSCOPE.Similar to Readfast,will improve your visual recognition powers.to save the Princess in the Tower of Doom.

B/B-STATS.Basketball Fans Statistic Record keeper data base.
SHRINK.Run this to remove rem's.in long programs.

November.

(a) TI-PING/3. Final part is a Quiz to find out much you have improved?

(b) D-BUILD/3 and D-TRANS/4 Out on a Limb Final parts.Build Data and transfer if from Tape to Disk or Disk to Tape if required.

WORDPOWER1.Part 1 in a 4 part Quiz Readers Digest format with multiple choice answers to Questions of Word meanings. Why have piles of RD's about when you can have the Quiz's all on disk or tape?

The full Disk will be Reviewed at a later date.

December.

WORDPOWER2 and the Best of the P.D. programs that we have access to. Maybe it will be one of yours if you get around to sending it in.

Texan 99.

One for keen Adventurer's this is a Disk based "agent Ø99" Text only Spy Adventure game in the style of the Infocom series,but nowhere near the price.This one is by a Member of the U.K.Nationwide TI User Group and is available on one Disk.

Clubline 99.
The library has 2 of the Monthly disks from this group,they include Basic,X-Basic,Assembly ,Logo,TE II, paged from an interesting LOAD Menu * Note Logo Module is required, but you can borrow the Club Module,so there is no need to miss out if you have not got one.
Their excellent Magazines are now also available to borrow and have all the Program instruction and Listings of their Monthly disks.

STOP PRESS.!!!
***** Program Image Programmes *****
As announced last Month,3 Excellent Utility Programmes from the same Farm that brought you the Programme below. are available through the Library on cassette for the Loader and Transfer Programmes .And on disk including full documentation the DISKHACK part 1. Which allows you to view hidden info between sectors .

You can if you wish contact direct Will McGovern for more information.

******* Funnel Writer V3.3 ********
Now with an optional LOADA on the DISK instead of "c 99" Files,which may not have been what you required. LOADA has Room on the OPT 3 Utility to customise your Disk with more of your Favourite Programs than was possible before.
As well there are more improvements including DM1000 V3.3. Note this is modified to run with Funelwriter and return to it when you have finished. It is now svailable through the Hunter Valley 99'ers Library for only $5.00 on a DS/SD Disk or $7.00 on 2 SS/SD inc.P and P anywhere in Australia.
Overseas extra for Surface or Airmail as requested.

We now have a full disk of separate Utilities for the above Programme . Also a full disk of User FONTS for Graphx to save you some work or give you Ideas.
??????????? WHY PAY MORE ??????????

Software Library

Any clubs or individuals interested in obtaining any PUBLIC DOMAIN software in volume disks have 2 choices
(a) Send blank initialised disks to us with return postage and $ 1.00 a disk or send us disks with programs on it and we will send at our cost an equal number of disks filled with programs requested or volume disks.
(b) We can supply programs,or volume disks on our disks for the cost of disk and P&P($4.00)

Free Software

At all Workshops we will have along some of the Club Public Domain software for you to download, So don't forget to bring along YOUR own ----!!!-CASSETTE RECORDER.-!!!----
Bring your own Disks and Tapes or buy them from the club.If you have any Programs to EXCHANGE bring them along as well.Disks or Tapes.!!!

Don't forget that the club sells CHEAP disks and cassettes also for each Dollar you spend you get a chance for the BI-MONTHLY surprize.

HELP! Anyone know of my X-BASIC Module.I left in club console(I think it had my initial stuck on) at last workshop and it may have been mistaken for the club's module. Please contact me if you know of or hear of it THANK YOU.
Fairware for now,

Happy programing,

Al Lawrence.

For all of you out there who tried to solve last month's tape 'TOWER OF DOOM' and failed miserably, here is a short description of each level from one who's gone the distance!

LEVEL 1 - Once you have passed the witch and dog you may enter the tower.
LEVEL 2 - To advance to level 3 enter letters correctly.
LEVEL 3 - The lights go out - to turn them on, enter correct letter. If you press the wrong letter your lights go out permanently!!
LEVEL 4 - Now, get passed the ghosts and out the door to continue to level 5.
LEVEL 5 - This is like pick-a-box with a 50% chance of making it to next level.
LEVEL 6 - Pick the right number between 1 & 100 before the walls play squash with your body.
LEVEL 7 - Congratulations, you have made it. You have saved your maiden and falen in love (lucky devil!).

Well, I hope this guide assists you in your quest.

Good luck!
CHRIS BURT
(the SUCCESSOR!!!)

HUNTER VALLEY 99'ERS USERS GROUP
LIBRARY SOFTWARE GUIDE

:OLOUR VISIONS. (COLOR-DEMO) (X/B)
\ colour extravaganza with a running time of 40 minutes after which it will epeat but with different colours. To stop and hold any screen press the PACE bar, to start again press G.

ONVERT (X/B)
menu driven conversion program. The five catagories are, 1 Metric onversion, 2 Temperature Conversion, 3 Numeric Base Conversion, 4 eometric Areas, 5 Geometric Volumes.

UT ON A LIMB (OUT-ON/LIMB) (X/B)
hese instructions are the first of a four part game program and database uilder. The other parts will be released in coming months. The game is a ord guessing game with an established database of 75 words or you can reate your own database.

NCRYPT. (X/B)
his program provides for the encoding and decoding of messages with the e of a keyword that you supply. Your message is encoded and saved on ape. The keyword is then destroyed so that no one can decode your text ithout it.

NDIAN. (X/B)
his Indian does a rain dance. It is recommended that you obtain an Ark fore you run this program.

USIC TUTORIAL. (LEARN-MUS) (B or X/B)
music tutorial showing different techniques that can be used to program sic. Breakpoints are used to allow the listing of a section of the ogram which has just been demonstrated.

NE MENS MORRIS. (9MENMORRIS) (X/B)
is game is as engrossing as Chess but without the complicated moves. A o player game which starts with the board vacant, each player has nine n and move then in turn onto the board. The object of the game is to get ree men in a row on any straight line except diagonally. When either ayer gets three men in line one of the oppositions men can be removed. ter both players have all their men on the board the men are moved one ace at a time to form lines of three until one player has only two men ft, the other is the winner. Originally the men were moved by the arrow ys, E, S, D, X and placed in position with the P key, however the 99'ERS Extended Basic Group have modified the program to use the ysticks to move the men and the fire button to position them. This is a ogram that did not do much for me at first but when I actually played ainst an opponent it turned into an interesting and challenging exercise. was the game against an opponent that prompted the addition of the ysticks to the program. The program is certainly worth having a go at.

-NO. (B or X/B)
e the S and D keys to move the man to dodge the falling boxes.

OOSH BOOMER. (SHOOSH) (X/B)
e the joystick to manouver the downhill skier around the trees, the indeers and the bears. Requires good eye/hand coordination. The name of e program comes from the sound of the skier as he races along and then ts an obstacle. Comes with four levels of difficulty (the slowest is the dest).

'ING TUTOR (TI-PING) (X/B) (SPEECH)
s is the instructions for a three part typing tutor program. The rest the program will be issued on the next monthly disk or tape.

CALL LOAD(16376,67,85,82,83,79,82,48,8)

16376 = >3FF8 which is in the REF/DEF table and is the address pointed to by address >2004. Then 67="C", 85="U", 82="R", 83="S", 79="O", 82="R", the name of the programme. The other two values point to where the programme CURSOR starts. And it works like this:- 48 = >30 and 8 = >08 put together >3008, the address the programme starts at.

CALL LOAD(12288,36,60,36,36,129,66,36,24)

12288 = >3000 and the other values are the character pattern values explained earlier, and >3000 is where the storage of those values starts.

CALL LOAD(12296,2,0,3,240,2,1,48,0,2,2,0,8,4,32,32,36,4,91)

12296 = >3008 the address that is pointed to from the last two bytes in the CALL LOAD(16376,....) and is the start of the programme proper. And now for the rest of those values.

2 = >02 and 0 = >00 = >0200 which is the op-code for LI R0.

3 = >03 and 240 = >F0 = >03F0 the address of character 30 (the cursor) in the pattern descripter table.

2 = >02 and 1 = >01 = >0201 op-code for LI R1.

48 = >30 and 0 = >00 = >3000 the start address of the data for the new cursor shape.

2 = >02 and 2 = >02 = >0202 op-code for LI R2.

0 = >00 and 8 = >08 = >0008 the number of bytes to read starting at >3000.

4 = >04 and 32 = >20 = >0420 op-code for BLWP.

32 = >20 and 36 = >24 = >2024 the utility workspace pointer for VMBW.

4 = >04 and 91 = >5B = >045B op-code for RT.

Now if we put that lot together into a small assembler programme it looks like this.

```
        DEF  CURSOR   Programme name
VMBW    EQU  >2024    XB equate for VMBW
        AORG >3000    AORG the programme to address >3000
*                     the upper end of low memory expansion
        DATA >243C,>2424,>8142,>2418 data for new cursor shape
CURSOR  LI   R0,>03F0 start of programme and loads the address
*                     of character 30 (the cursor) in the
*                     pattern descripter table into R0
        LI   R1,>3000 loads the starting address of where
*                     the new character shape data is stored
        LI   R2,>0008 loads the number of bytes to read
        BLWP @VMBW    branch to VMBW routine and do it
        RT            return
        END
```

If you type that in and assemble it in the name of HVCURSOR using only the R option. Then in XB CALL INIT::CALL LOAD("DSK1.HVCURSOR")::CALL LINK("CURSOR"). Your nice newly designed cursor will appear.