8412 85 ?? Dec

TI*MES

NEW RELEASES
SAMS
Entertainment Games in TI BASIC and Extended BASIC
TI-99/4A: 24 BASIC Programs
BASIC Tricks for the TI-99/4A
The Tool Kit Series TI-99/4A Edition
TI-99/4A BASIC Programs
TI-99/4A Graphics and Sounds
TI-99/4A: 51 Fun and Educational Programs

WINTER 84/85 ISSUE NUMBER 7

TI99-4a EXCHANGE presents..

in this issue of TI*MES..

Thank you for your support

2) Editorial
3) PROGRAM Hangman by Gordon Jones
5) Random Eyes - Graham Baldwin
9) PROGRAM - Lower case letters -D.Ford
10) Technical Tall. -K.Johnson
11) Peeking MINIMEM - John Roe
15) Disks to FORTH - Phillip Marsden
21) Brussels Spouts - Ian Swales
24) Scene USA
25) Howard writes - H.Greenberg
[redacted] UK ADVERTS.
[redacted]ts
36) Babble on TI-LINES - P. Brooks
41) Micro-lines - D.Brown
43) Hardware review - P.Brooks
45) TI WRITER Tips - A.D.Burt
47) Tips from Tigercub - Jim Peterson
49) PROGRAMS - Sounds and graphics
51) EXTENDED TUTOR
54) Rambles - Stephen Shaw
61) Shaping TI Sounds - Sean O'Brien

Happy New Year TI99-4 Users

```
100 REM ****************
110 REM * single pixel *
120 REM *----- drawing  *
130 REM *by p.Brooks     *
140 REM * Adapted for joy-
150 REM * stick, speeded up
        * a bit, by
          * Jim Peterson
160 REM *****************
200 CALL CLEAR
210 INPUT "SCREEN &TRACE COL
OURS ??,??
          ":SC,TC
220 INPUT "STARTING ROW & CO
LUMN ??,??
          ":R,C
230 R=R*8
240 C=C*8
250 CALL CLEAR
260 CALL SCREEN(SC)
270 FOR I=1 TO 14
280 CALL COLOR(I,TC,1)
290 NEXT I
300 CALL CLEAR
310 CALL HCHAR(1,1,31,768)
320 S=32
330 DIM C$(128)
340 B$="0000.0001.0010.0011.
0100.0101.0110.0111.1000.100
1.1010.1011.1100.1101.1110.1
111"
350 H$="0123456789ABCDEF"
360 Z$="0000000000000000"
370 GOSUB 440
380 CALL JOYST(1,DX,DY)
390 CALL KEY(3,K,ST)
400 IF ST<>0 THEN 670
410 C=C+DX/4
420 R=R-DY/4
430 GOTO 370
440 Y=INT(R/8+.875)
450 P=INT(C/8+.875)
460 CALL GCHAR(Y,P,H)
470 IF H>31 THEN 540
480 IF S=143 THEN 390
490 S=S+1
500 C$(S-31)=Z$
510 CALL CHAR(S,Z$)
520 CALL HCHAR(Y,P,S)
530 H=S
540 H=H-31
550 B=C-P*8+8
560 P=2*R-16*Y+16+(B<5)
570 IF B<5 THEN 590
580 B=B-4
590 I$=SEG$(B$,POS(H$,SEG$(C
$(H),P,1),1)*5-4,4)
600 REM
610 I$=SEG$(I$,1,B-1)&"1"&SE
G$(I$,B+1,4-B)
620 I=POS(B$,I$,1)/5+.8
630 C$(H)=SEG$(C$(H),1,P-1)&
SEG$(H$,I,1)&SEG$(C$(H),P+1,
16-P)
640 CALL CHAR(H+31,C$(H))
650 RETURN
660 REM SUB PROG HERE
670 PRINT "you can put a sub
 program":"here if you wish"
680 REM END SUB
```

*PUBLISHED TI*MES*
ISSUE NUMBER3(C)

Boundary checks have been removed for speed so don't run off the borders!

Welcome _____________ to TI99/4a EXCHANGE

TI*MES TI*MES TI*MES TI*MES TI*MESTI*MES TI*MES TI*MES TI*MES TI*MES
WINTER 84/85 NUMBER SEVEN

40,Barrhill,Patcham,BRIGHTON,East Sussex,BN18UF.Tel:0273 503968(evenings)

THE ONLY U.K. TI USERS GROUP NOTHING LESS FOR 1985...

First things first it is 1985!

a Happy New Year...

Yes more than a year has passed and we are still going strong. Many of you will not give up the TI99/4a. The renewals prove that you want us to carry on. The questionaire slips attached to the renewal returns have given us a clue as to what you would like to see in your own edition of TI*MES. In the main you all indicate a serious use of the TI99/4a home computer. This will be the theme for TI*MES in 1985. We are delighted by the huge post that keeps coming in with the kindest remarks too numerous to mention. Your letters are very welcome and thank you all for the trouble in writing to us. If you want a reply please be sure to send a STAMPED ADDRESSED ENVELOPE.

Convention 1985?

In spite of the pressure of new computers the Nationwide TI users convention held at the RITZ Manchester in November was very well supported indeed. Over a 1000 people came, it was an unprecedented response. TI Users from Scotland to Cornwall made the effort for the one day show.

Again the post bag received was full of very kind remarks about the show. We must give our thanks to all who supported the event, yes it was commercial but without the traders this show would not have happened. We have learnt much from staging the event which can be put right at the next Convention. Keep Easter free in your diary if the support is there another show could be held around that time. Watch computer press for details.

What of the future?

The future regarding software for your TI may look very bleak indeed, happy to say there is still plenty of it about to last for many more years. Books are still being produced for the TI99/4a as late as November from the USA. As regards expansion/hardware this is still available. There are however very long delays so we advise you NOT to part fully with any cash for goods ordered by mail unless the product is in stock, certainly send a deposit but get written confirmation of order. Extended basic and Mini memory should now be easy to obtain and continue to be the most essential solid state software to own. In spite of the price being high they are in fact cheaper compared to two years ago.

Group Library.

There are many new software programs to take advantage of in the TI*MES SOFTWARE LIBRARY certain to make very good use of your computer/expansion system. Cassette and disk based games and utilities from all over the world of a high standard, we will shall upgrade the library each six months. The library contains programs exclusive only to TI99/4a Exchange Britain which you as a member have access.

*** Thankyou ***

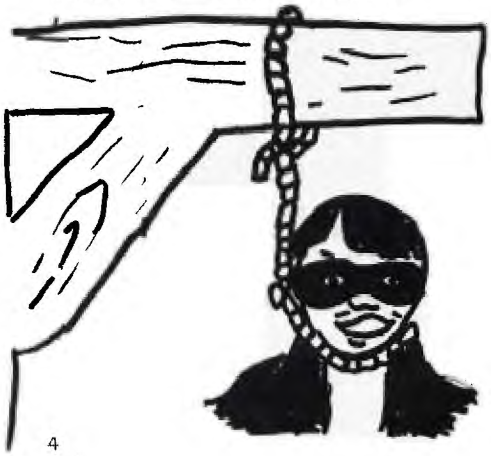
Members who continue to submit news and views to exchange with fellow TI Users. These contributions will also be found in newsletters all over the world. This group is Internationally known and recognized, so any one wanting world fame please share your knowledge, if only to make TI computing an interesting hobby. ED C&A.

```
1ØØ REM  TI*MES LIBRARY E2b
              "HANGMAN"
          BY GORDON JONES
11Ø REM TI BASIC
12Ø CALL CLEAR
13Ø CALL SCREEN(4)
14Ø CALL SOUND(2ØØØ,262,2,33
Ø,2,392,2)
15Ø GOTO 186Ø
16Ø CALL CLEAR
17Ø REM "BASE"
18Ø CALL CHAR(96,"FFFFFFFFFF
FFFFFF")
19Ø REM "POST"
2ØØ CALL CHAR(112,"3C3C3C3C3
C3C3C3C")
21Ø REM "BEAM"
22Ø CALL CHAR(113,"ØØØØØØFFF
FFFFFFF")
23Ø REM "ROPE"
24Ø CALL CHAR(128,"Ø8Ø8Ø8Ø8Ø
8Ø8Ø8Ø8")
25Ø REM "HAT"
26Ø CALL CHAR(12Ø,"3C3C3C3C3
C3CFFFF")
27Ø REM "FACE"
28Ø CALL CHAR(136,"3C6A7E243
C181818")
29Ø REM "UPPER BODY"
3ØØ CALL CHAR(144,"FFFFFF243
C3C7E7E")
31Ø REM "LOWER BODY"
32Ø CALL CHAR(145,"667EFFFF6
6666666")
33Ø REM "LEFT LEG"
34Ø CALL CHAR(152,"6Ø6Ø6ØEØE
ØØØØØØØ")
35Ø REM "BOTH LEGS"
36Ø CALL CHAR(153,"666666E7E
7ØØØØØØ")
37Ø REM "LEFT ARM"
38Ø CALL CHAR(1Ø4,"Ø7Ø7Ø7Ø4Ø
4Ø4Ø4Ø4")
39Ø REM "RIGHT ARM"
4ØØ CALL CHAR(1Ø5,"EØEØEØ2Ø2
Ø2Ø2Ø2Ø")
41Ø CALL CLEAR
42Ø CALL COLOR(9,13,1)
43Ø CALL COLOR(1Ø,9,1)
44Ø CALL COLOR(11,14,1)
45Ø CALL COLOR(13,16,1)
46Ø CALL COLOR(14,9,1)
47Ø CALL COLOR(15,5,1)
48Ø CALL COLOR(16,2,1)
49Ø LET WW=Ø
5ØØ LET YY=Ø
51Ø LET ZZ=Ø
52Ø CALL SCREEN(12)
53Ø IF AA=1 THEN 57Ø
54Ø PRINT "ENTER WORD MAX 2Ø
 LETTERS,": : :"MAX 2 WORDS
SPACES INCLUDED!": : : : :
55Ø INPUT A$
56Ø CALL CLEAR
57Ø LA=LEN(A$)
58Ø IF LA>2Ø THEN 41Ø
59Ø Y=POS(A$," ",1)
6ØØ IF Y=Ø THEN 68Ø
61Ø X$=SEG$(A$,1,Y-1)
62Ø LB=LEN(X$)
63Ø Z$=SEG$(A$,Y+1,LA-Y)
64Ø LC=LEN(Z$)
65Ø CALL HCHAR(1Ø,3,45,LB)
66Ø CALL HCHAR(1Ø,Y+3,45,LC)
67Ø GOTO 69Ø
68Ø CALL HCHAR(1Ø,3,45,LA)
69Ø T$="ENTER LETTER"
7ØØ FOR T=1 TO 12
71Ø TT=ASC(SEG$(T$,T,1))
72Ø CALL HCHAR(6,T+4,TT)
73Ø IF T=12 THEN 75Ø
74Ø NEXT T
75Ø V$="WRONG LETTER'S USED"
76Ø FOR V=1 TO 19
77Ø VV=ASC(SEG$(V$,V,1))
78Ø CALL HCHAR(15,V+2,VV)
79Ø IF V=19 THEN 81Ø
8ØØ NEXT V
81Ø CALL SOUND(2ØØ,44Ø,5)
82Ø CALL KEY(3,KEY,STATUS)
83Ø IF STATUS=Ø THEN 82Ø
84Ø LET XX=Ø
85Ø N=KEY
86Ø FOR NN=1 TO LA
87Ø IF N=ASC(SEG$(A$,NN,1))T
HEN 9ØØ
88Ø IF NN=LA THEN 138Ø
89Ø NEXT NN
9ØØ CALL HCHAR(1Ø,2+NN,N)
91Ø XX=XX+1
92Ø GOTO 98Ø
93Ø REM "WRONG LETTER COUNT"

94Ø YY=YY+1
95Ø CALL HCHAR(17,YY+4,N)
96Ø REM "HANG COUNTER"
97Ø ZZ=ZZ+1
98Ø ON ZZ GOSUB 1Ø4Ø,1Ø7Ø,11
ØØ,113Ø,116Ø,119Ø,122Ø,126Ø,
129Ø,132Ø,135Ø
99Ø IF ZZ=11 THEN 1ØØØ ELSE
82Ø
1ØØØ CALL COLOR(14,16,1)
1Ø1Ø CALL SOUND(1ØØØ,11Ø,1)
1Ø2Ø PRINT "YOU'RE HUNG!!!":
 :A$
1Ø3Ø GOTO 159Ø
1Ø4Ø CALL HCHAR(24,2Ø,96,11)
1Ø5Ø CALL SOUND(5ØØ,-1,2)
1Ø6Ø RETURN
1Ø7Ø CALL VCHAR(8,25,112,16)
1Ø8Ø CALL SOUND(5ØØ,-1,2)
1Ø9Ø RETURN
11ØØ CALL HCHAR(8,25,113,6)
111Ø CALL SOUND(5ØØ,-1,2)
112Ø RETURN
```

```
1130 CALL VCHAR(9,28,128,2)
1140 CALL SOUND(500,-1,2)
1150 RETURN
1160 CALL HCHAR(11,28,120)
1170 CALL SOUND(500,-1,2)
1180 RETURN
1190 CALL HCHAR(12,28,136)
1200 CALL SOUND(500,-1,2)
1210 RETURN
1220 CALL HCHAR(13,28,144)
1230 CALL HCHAR(14,28,145)
1240 CALL SOUND(500,-1,2)
1250 RETURN
1260 CALL HCHAR(15,28,152)
1270 CALL SOUND(500,-1,2)
1280 RETURN
1290 CALL HCHAR(15,28,153)
1300 CALL SOUND(500,-1,2)
1310 RETURN
1320 CALL HCHAR(13,27,104)
1330 CALL SOUND(500,-1,2)
1340 RETURN
1350 CALL HCHAR(13,29,105)
1360 CALL SOUND(500,-1,2)
1370 RETURN
1380 IF XX=0 THEN 930
1390 FOR WW=1 TO LA
1400 CALL GCHAR(10,WW+2,P)
1410 PP=ASC(SEG$(A$,WW,1))
1420 IF P=PP THEN 1430 ELSE
1450
1430 IF WW=LA THEN 1470
1440 NEXT WW
1450 WW=WW-1
1460 GOTO 810
1470 FOR SS=27 TO 29
1480 CALL VCHAR(11,SS,32,6)
1490 NEXT SS
1500 PRINT "**SAVED**": :A$
1510 CALL HCHAR(20,28,153)
1520 CALL HCHAR(19,28,145)
1530 CALL HCHAR(18,28,144)
1540 CALL HCHAR(18,27,104)
1550 CALL HCHAR(18,29,105)
1560 CALL HCHAR(17,28,136)
1570 CALL HCHAR(16,28,120)
1580 CALL SOUND(2000,262,2,3
30,2,392,2)
1590 CALL HCHAR(3,2,32,21)
1600 CALL HCHAR(7,2,32,21)
1610 M$="ANOTHER GO?"
1620 FOR M=1 TO 11
1630 MM=ASC(SEG$(M$,M,1))
1640 CALL HCHAR(5,M+4,MM)
1650 IF M=11 THEN 1670
1660 NEXT M
1670 CALL HCHAR(7,7,89)
1680 CALL HCHAR(7,9,79)
1690 CALL HCHAR(7,10,82)
1700 CALL HCHAR(7,12,78)
1710 CALL KEY(3,KEY,STATUS)
1720 IF STATUS=0 THEN 1710
1730 IF AA=1 THEN 1750
1740 IF KEY=89 THEN 410
1750 IF KEY=89 THEN 1950
```

```
1760 END
1770 READ X,Y,M$
1780 FOR M=1 TO LEN(M$)
1790 V=ASC(SEG$(M$,M,1))
1800 CALL HCHAR(X,Y+M,V)
1810 NEXT M
1820 RETURN
1830 DATA 5,6,WELLCOME TO HA
NGMAN,6,6,*****************
*,8,2,Do You Wish To Play Ag
ainst Me,11,2,Or A Friend?
1840 DATA 14,6,PRESS,17,6,1
For Me,20,6,2 For A Friend
1850 RETURN
1860 FOR Z=1 TO 7
1870 GOSUB 1770
1880 NEXT Z
1890 CALL SOUND(200,440,5)
1900 CALL KEY(3,K,S)
1910 IF S=0 THEN 1900
1920 IF K=49 THEN 1940
1930 IF K=50 THEN 160
1940 LET AA=1
1950 RANDOMIZE
1960 FOR RN=1 TO INT(RND*25)
+1
1970 READ A$
1980 NEXT RN
1990 RESTORE 2010
2000 GOTO 160
2010 DATA EARRINGS,PICTURE,A
USTRIA,HAMSTER,TELEVISION,IN
FORMATION,SEQUENTIAL,WARRANT
Y,ASSISTANCE
2020 DATA DIAGNOSTIC,COMPREH
ENSIVE,ATLANTIC OCEAN,RETRIE
VE,APPLICATION FORM,VEXED,ST
OMACH,ISOMETRICICAL
2030 DATA INSUBORDINATION,CH
RYSANTHEMUM,VESTIBULE,PHOTOS
YNTHESIS,WHIMSICAL,PRESUMPTU
OUS,NECESSARY,KALEIDOSCOPE
```



RANDOM EYES

Graham Baldwin.

Welcome to another Random Eyes, which has a slightly different format this issue, comprising a (fairly) serious look at programming techniques, followed by the usual rubbish that bounces round the hollow confines of my skull.

What is the most futile program it is possible to write for your own lasting amusement? The answer is surely the adventure program, which, once written, can hardly be played for enjoyment by the writer as he knows all the possible questions, answers and moves and can no lònger be amazed and delighted by his own creation. Of course, you could bore friends silly with it, or, if you get lucky, sell it for real money, but the biggest plus is the sheer enjoyment of writing a long and complicated program from your own imagination that performs exactly as planned. Apart from the exercised of a diseased (or otherwise) imagination in writing the basic plot, adventure programming gives marvellous practise in string handling, data manipulation, efficient use of memory, and not least sheer logic in making the whole thing hang together.

A 'simple' adventure program can be broken down into several distinct parts, plus one that is a little more nebulous.

1. Descriptions of locations.
2. Descriptions of objects.
3. 'Move' between locations routine.
4. Verb/noun recognition and manipulation.
5. Error checks at each stage.
6. The logic that links all the above.

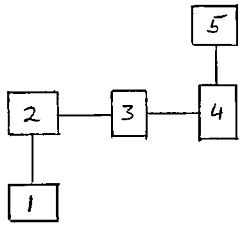
Before going any further, let me state here and now that the routines I'll be giving are pared to the bone and aren't necessarily the best, the fastest or the most memory-efficient, but they're mine, and I understand them!

The plot of an adventure is entirely up to the writer and I can offer little help here, save to say that with the amount of memory available on the console you won't get 200 locations, 150 objects and a large vocabulary squeezed in and still have room for any logic to run the thing. (In theory it IS possible, with a great deal of data compression and sophisticated retrieval techniques, but you run the risk of the player falling asleep while the computer studiously considers each move and works out a suitable response.)

To start with, aim for about 25 to 30 locations and objects and a vocabulary of about 45 to 50 words. Depending on how tightly you can program you may be able to add some more later, although the very nature of such a large program demands that it should be planned in detail before you even think of turning on the computer. Flow-chart your plot and try to weed out discrepancies and ambiguities at this stage, as trying to remove them from the finished program may make you wish you had never heard of adventures. Be as imaginative as you like when plotting but try to keep within the bounds of reasonable logic as any future player will have extreme difficulty in getting anywhere if you insist, for example, that he digs a hole with a key or opens a door with a hamster. There are plenty of legitimate dirty tricks you could use, such as leaving the player stranded on a roof because he forgot to tie the ladder securely, or enticing him into an unlit maze when he is unlikely to be carrying a lamp.

→

The location and object descriptions we'll take as read for the moment, with flashbacks as necessary, so it's on to the 'move' routine. This seems to cause problems for aspiring adventure programmers but with a little thought several workable methods can be devised. Let's look at the map of a simple five-location scenario and see how we can fit it in to a program.



(a)

LOCATIONS (LOC)

	1	2	3	4	5
W			2	3	
E		3	4		
S		1			4
N	2			5	

(b)

If we compare map (a) with table (b) we can see that each location (indicated over the top of the table) has its exits and their directions shown immediately below, thus location 3 has an exit West, leading to location 2 and another, East leading to location 4. OK so far? The first routine shows the idea in action, but beware if you bother to type it in as it is not fully error-trapped (for simplicity) and can be fooled. Incidentally, the location array L$() is, of course, a string array which can save a huge amount of memory in a full-size program, so we need an occasional VAL to take out an integer for GOSUBbing and so on, and STR$ to put one back in. As you can see, the routine could be compressed a little (you DID see, didn't you?) but I've left it 'open' for clarity

```
100 REM LISTING 1
110 REM
120 REM'MOVE' ROUTINE
130 REM
140 CALL CLEAR
150 DIM L$(4,5)
160 FOR A=1 TO 5
170 FOR B=1 TO 4
180 READ L$(B,A)
190 NEXT B
200 NEXT A
210 DATA 2,,,,,1,3,,,,4,2,5,
,,3,,4,,
220 LOC=1
230 ON LOC GOSUB 300,320,340
,360,380
240 INPUT "WHAT NOW?":D$
250 DIR=POS("NSEW",D$,1)
260 IF DIR=0 THEN 400
270 IF L$(DIR,LOC)="" THEN 4
20
280 LOC=VAL(L$(DIR,LOC))
290 GOTO 230
300 PRINT "PATH"
310 RETURN
320 PRINT "CASTLE"
330 RETURN
340 PRINT "COURTYARD"
350 RETURN
360 PRINT "DINING-ROOM"
370 RETURN
380 PRINT "BEDROOM"
390 RETURN
400 PRINT "CAN'T UNDERSTAND"
410 GOTO 240
420 PRINT "CAN'T GO THAT WAY
"
430 GOTO 240
```

→

All this routine does is extract the location number from the table, use it to display the description and change the location variable (LOC) to the new value, according to the INPUT recieved.
As things stand at the moment the player can wander unimpeded through all five locations, but we may want to introduce a locked door or similar that the player must open (traditionally with a key, but you may wish to be more imaginative and batter it down with the severed tail of the dragon you've just killed). In the above example we could bar the way between locations 3 and 4 by leaving the East exit from location 3 as a null (or "") until the player types in the correct command, whereupon we insert a "4" into the array at LOC 3, East. This needs careful mug-trapping however, or the player may be able to open the door from another location without him (or the program) understanding what is going on.

Now that our player can wander around the adventure we need a word recognition procedure to allow him to communicate with the program. Traditionally a verb/noun system is used, eg TAKE SHOVEL or EXAMINE NAVEL, often abbreviated to the first three letters of each word (although if you are clever you may be able to devise a full sentence-recognition scheme).

Put simply, every time the player inputs a command we have to compare that input with the program's vocabulary and cause the program to respond accordingly. Listing 2 demonstrates a way of extracting a verb number and a noun number from the vocabulary of five verbs and five nouns, which could then be used in a series of 'ON V GOTO...' and 'ON N GOTO...' to branch around the program. (I've been naughty in this program and jumped out of FOR-NEXT loops, as you can see. I seem to remember reading somewhere that you can do this on the 99/4A but not on the 99/4. If it causes you problems or you don't like the style then use a simple Z=Z+1 counter.)

```
100 REM   LISTING 2
110 REM
120 REM VERB/NOUN ROUTINE
130 REM
140 FOR I=1 TO 5
150 READ V$(I),N$(I)
160 NEXT I
170 DATA HEL,RIN,EXA,DOO,TAK
,STO,MOV,CHA,REA,BOO
180 INPUT "WHAT SHALL I DO?"
:Q$
190 FOR I+1 TO 5
200 IF POS(Q$,N$,(I),4)=0 TH
EN 230
210 V=I
220 GOTO 250
230 NEXT I
240 GOTO 300
250 FOR I=1 TO 5
260 IF POS(Q$,N$(I),4)=0 THE
N 290
270 N=I
280 GOTO 320
290 NEXT I
300 PRINT "I DO NOT UNDERSTA
ND"
310 GOTO 180
320 PRINT "VERB NO";V;"NOUN
NO";N
330 GOTO 180
```

The verbs in the listing are:- Help, Examine, Take, Move and Read. The nouns are:- Ring, Door, Stone, Chair and Book. You probably noticed that the words in DATA have been cut to three letters each. This allows the player to type in the shortened version and saves wear and tear on his fingers and patience.

However, you, the player and the program could confuse each other if you wanted to use different words that started with the same letters, such as BOOK, BOOKCASE and BOOKSHOP. How could the program decide which you wanted? As it stands at the moment it can't, so you'd have to re-think the vocabulary and call the bookcase 'shelves' or similar. (This goes back to the point about careful plotting and ironing out ambiguities before writing the program itself.)

As this routine compares the input with the vocabulary, starting from the beginning each time, the program will obviously run faster if we place the more commonly used words, such as TAKE or EXAMINE at the beginning of the vocabulary list and leave the lesser-used words at the end.

That's all about adventures for now but the next article will look at ways of linking the program internally, checking for INPUT errors, a TAKE/DROP routine and generally getting the program up and running.

To my regret I didn't manage to get to the convention in Manchester (the logistics of arranging time off work, travel, kids etc. defeated me) and thus missed meeting old and new friends. From all accounts it was a great success. How could it be otherwise with so many TI owners under the same roof?

Many thanks to David Vincent who wrote to me offering details about the PRK module to solve my cassette filing problems. It looks as though there is no way of using the bare console for efficient cassette filing, although I'm willing to be proved wrong! As far as the price of a disk system is concerned you could buy an awful lot of card indexes for the same price, even though you can't store programs on them...

Stephen's remark about RUN (in Ex BAS) as a program statement reminded me that RUN "CS1", entered in immediate mode, will auto-run an Ex BAS program, ie once the cassette instructions have been obeyed the program will RUN without having to type the word.

ACCEPT AT has one or two peculiarities that I've noticed. For instance, the SIZE clause, if given a negative value, will put whatever is on the screen at the given co-ordinates into the variable merely by pressing ENTER. Very neat, and useful for stepping rapidly through multiple INPUTS of common information on the screen. The clever bit comes when the bit of the screen we want to ACCEPT has a number in that position. If we specify a string variable to ACCEPT it, that number is turned into a string (ie automatic STR$) and apparently stored as such. This could be useful, but I'd hate to debug someone else's program that used the trick! By the way, the number in SIZE determines how many characters are accepted when ENTER is pressed. A snag arises when we DISPLAY a number, as the machine (mine, anyway) places a blank in front of the displayed number, effectively shifting it one place to the right, and to ACCEPT it I have to add 1 to the column value of the ACCEPT co-ordinate. There's probably a simple explaination...

Sorry about the errors in TI*MES No6 Quickies... The 'ASC value in HCHAR' does work, but of course only when we use the ASCII code... A slop of the typewriter... The bit about non-existant sprites needs amplification. Trying to use COINC or MOTION on sprites that do not yet exist may confuse the processor and give false coincidences on the sprites that DO exist.

Happy and b£g free computing.

Graham Baldwin

Graham Baldwin 32, Ellesmere Drive, SOUTH CROYDON, Surrey CR2 9EJ.

lower case

by

derek ford

This program will load 'lower case' letters (small letters NOT small capitals) into your T.I. 99.4A. By loading the program into one of three different sets of ASC11 numbers you have three choices.

1 If you use ASC11 numbers 65 to 90 inclusive your small letters will take the place of your big capitals, ie. Alpha lock down gives lower case letters and Alpha lock up gives your normal small capitals - ONLY WHILE THE PROGRAM IS RUNNING - ie. any upper case characters displayed on your screen, or any upper case input from the keyboard into your program will in fact display small letters. Your computer will respond as if you were using large capitals..

2 If you use ASC11 numbers 97 to 122 inclusive your small letters will take the place of your small capitals, ie. Alpha lock down gives normal big capitals and Alpha lock up gives small letters - AGAIN, ONLY WHILE THE PROGRAM IS RUNNING . You will have the same effect as in Choice 1, above, bearing in mind that in many instances your TI does not respond to (what in effect would be) small capitals.

In order to use the two choices above you must remember that the following listing must be part of your larger program.

3 If you use ASC11 numbers 129 to 154 inclusive then you have big capitals with Alpha lock down, you have small capitals with Alpha lock up, just as normal. Now if you press Control + A, you will have your small letter a, if you press Control + B, you will have your small letter b, and so on. (NB. Alpha lock up or down).

With Choice 3 you can 'NEW' your program without losing your small letters. But save the program on tape for next time. If you 'BYE' or 'QUIT' or switch off your computer then you lose the lower case letters. It is a simple matter then to reload the program from your tape.

To make this program work, type it in, then run it. Your screen will turn the usual green while it runs, (about 10 seconds), but nothing else happens.

But now big and small capitals remain as normal and your lower case letters are obtained by pressing Control and selected key.

NOTE: All three choices will work fine in TI. basic. Only the first two will work in extended basic because ASC11 numbers 144 to 159 are used in extended to control sprites and so are not available. So if you entered Choice 3 in extended, you would only have the first fifteen letters in lower case.

a b c d e f g h i j k l m n o p q r s t u v w x y z

```
 8 REM  LOWER CASE CHARACTERS
 9 REM  BY DEREK FORD
10 DATA 000038043C443C00
20 DATA 0020203C22223C00
30 DATA 00001C2020201C00
40 DATA 0004043C44443C00
50 DATA 0000384478403C00
60 DATA 000C101810101000
70 DATA 00003C44443C0438
80 DATA 4040407844444400
90 DATA 0010003010103800
100 DATA 0004000404042418
110 DATA 0020283030282400
120 DATA 001010101010100C00
130 DATA 00007C5454545400
140 DATA 0000784444444400
150 DATA 0000384444443800
160 DATA 0000784444784040
170 DATA 00003C44443C0406
180 DATA 00001C2020202000
190 DATA 0000384038047800
200 DATA 0010381010100C00
210 DATA 0000444444443800
220 DATA 0000444428281000
230 DATA 0000445454542800
240 DATA 0000442810284400
250 DATA 0000444444443C0438
260 DATA 00007C0810207C00
270 FOR A=1 TO 26
280 READ L$
290 CALL CHAR (128+A,L$)
300 NEXT A
310 END
```

In the listing, line 290 gives you the third choice.

For Choice 1 - line 290 should read - CALL CHAR (64+A,L$)

For Choice 2 - line 290 should read - CALL CHAR (96+A,L$)

Derek Ford

D FORD

REMS ON KEEYssss
by K. JOHNSON, SFV 99ers USA.

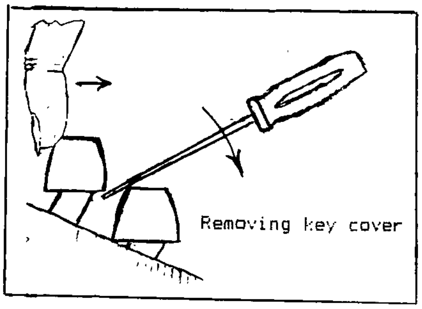
Have trouble with ssstuttering keys on your TI99/4a? It commonly happens, particularly on the most often used keys like E S D X (game arrow keys) on the older consoles. The other common problem is the failure to input when a key is pushed. Both of these problems are caused by dirty key switch contacts and the problems can be eliminated by cleaning the contacts.

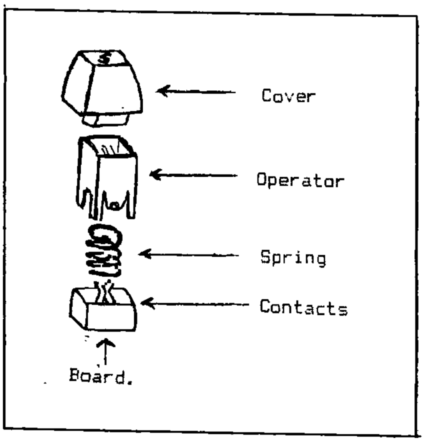
It isn't too difficult. First, make sure the console is turned off. Then, while holding down the key in front of the bad key as a pivot and a screwdriver, table knife or other flat object as a pry bar, catch the lower front edge of the bad key (see sketch below). Gently pry up on the key until the cover comes off. Pulling forward on the bad key will help.

What you should see under the cover is a little yellow or green plastic box with a small square bar running inside. Sticking above and on each side of the bar you will see two flat gold metal tabs. These are the switch contacts. These are to be sprayed with "Switch Contact cleaner", "TV Tuner Switch Cleaner" or similar greaseless spray cleaner safe for plastics. These are available at any radio, electronics or TV parts store and some hardware stores. Spray between the swtich contact faces with a short squirt of cleaner. Avoid touching the contacts with anything that might bend them out of position.

Occasionally, instead of remaining on the keyboard, the yellow(or green) plastic switch 'operator' wil come off with the key cover. This exposes the switch and return spring (see below). The yellow or green operator should be pulled out of the key cover by working it back and forth in the cover and pulling. Then carefully(!) place the operator over the spring with the bar of the operator lined up to fit between the contacts. Press the operator back into the board. It should snap back into place. Then do the cleaner trick and replace the key cover.

If the stuttering doesn't go away with just the cleaner, remove the cover and slide a narrow strip of clean bond paper between the contacts. Then slowly pull the paper out, which wipes the dirt off the contacts. Release the operator, give the contacts a squirt of the cleaner and replace the key cover.





PEEKING WITH MINIMEM

BY JOHN ROE.

Although the basic TI99/4a console has no facilities for PEEKing or POKEing, it can be done with additional hardware. Perhaps the simplist way is with just the console plus MINIMEM, ie without Disc Controller or memory expansion.

There is a distinction made between CPU memory and VDP memory. CPU (26K ROM/GROM and 8K RAM) is not directly useable by the programmer, but is reserved for the operating system. In the basic console the VDP memory is where the users program and data are held, although here again the operating system reserves some 2K of the VDP's 16K RAM for its own use.

The memory locations in the VDP RAM are numbered from Ø to 16383. The numbering system for the CPU memory also includes this block of numbers so some way had to be found of distinguishing the two blocks of numbers when PEEKing of POKing. For the CPU memory the commands CALL PEEK and CALL LOAD are used and for VDP memory the commands are CALL PEEKV and CALL POKEV.

There is some confusion of nomenclature in refering to the relative positions of different part of the VDP RAM. To avoid this I propose to refer to the low numbered memory locations as 'top' of memory and the high numbered memory locations as 'bottom' of memory, so that going from a lower numbered location to a higher numbered location is going 'down'in the memory and vica versa as going 'up' the memory. This will have the advantage that in the tables below the various items within a group are arranged in order down the page with the location numbers running likewise down the page, low numbers first.

You can either PEEK in the Command mode by CALL PEEKV(X,A,B,C...) followed by PRINT A;B;C;... or use a short program to CALL PEEKV and PRINT the values in memory one by one using the up and down arrow keys to work your way up or down the memory. Another way (not strictly PEEKing) is to enter the program lines to be examined and then SAVE MINIMEM,QUIT and select Easy bug from the Menu. When the Easy Bug Menu is displayed key in M7ØØØ, and then work your way down the Minimem RAM. The only snag with this is that both memory locations and code are in Hex, OK ifyou're proficient in mental conversions or have a Hex conversion calculator handy, but a bit wearing otherwise.

Now for some practical work after all this theory. First enter the program line 1Ø CALL CLEAR. Then, in the Command mode enter CALL PEEK(916370,A,B,C,D,E,F,G,H,I.J.K.L.M.N), then PRINT A;B;C;D;E;F;G;H;I;J;K;L;M;N You should get in your display the following series of numbers:-

```
Ø,1Ø,63,247,9,157,2ØØ,5,67,76,69,65,82,Ø
```

At first glance a rather meaningless set of numbers, but hang on a bit. Take the 5 numbers beginning with 67. Isn't there something familiar about them? If you haven't got it look at the list of ASC11 codes in your URG and all will be made CLEAR (pun intended). Yes, you're right, these numbers are the ASC11 codes for CLEAR. To cut a long story short all the other numbers can be decoded in some way. The meanings are set out in the following table.

↗

PEEKING WITH MINIMEM.

LOCATION No.	CODE	MEANING	
16370	0	}Line number	)LINE
1	10	}	)NUMBER
2	63	}63*256+247=16375	)TABLE
3	247	}Pointer to start of line 10	)
4	9	}Number of locations used to end of line	
5	157	}Token for CALL	
6	200	}Token to indicate 'unquoted string'	
7	5	}No. of characters in string	
8	67	}ASC11 code for C	
9	76	} " " L	
80	69	} " " E	
1	65	} " " A	
2	82	} " " R	
3	0	}End of line indicator.	

From this and from further extensive PEEKing over several weeks I have drawn the following conclusions:-

a) Program lines are entered from bottom of memory (ie from 16383) upwards in strict order of entry regardless of line number. For this purpose all edited lines, whether altered or not, are treated as new lines.

b) The line number entries are separated from the program lines and gathered together in a Line Number table held in memory above the program lines.

c) Line numbers and memory location numbers are held as 16 bit binary numbers(2 bytes), so occupying two memory locations. The number in the first location should be multiplied by 256 and added to the number in the second location to get the full decimal number.

d) The first two locations in a line number entry hold the line number and the next two the location number of the first & operational code item in the program line, i.e. the code item immediately following the code for the number of bytes used in the line.

e) The final item of code in a program line is always 0

f) Key Words are represented tokens, just one number. (see Peter Brooks article on Tokens in TI*MES no.5). Other Reserved Words are spelt out letter by letter in ASC11 code, and are preceded by the token 200 (for unquoted string) and the number of characters in the string.

g) Numbers appearing in program lines are also treated, in the program lines only, as unquoted strings. Thus in the line 10 A=123 the number would appear as 200,3,49,50,51.

→

PEEKING WITH MINIMEM.

Now let's take things a little further. Numeric and string variables and arrays are stored in the Symbol Table which sits somewhere above the Line Number Table. Usually immediately above the Line Number Table but this is a little unpredictable. In one or two odd examples I have had to search elsewhere for it.

The Symbol Table is not created until the program is actually RUN, so it is important that you always RUN your program lines first before PEEKing.

Enter line 1Ø X=12345678901234, and RUN. The TI holds and calculates numbers wit the maximum of 14 digits, so the above number is the longest that the program will store. In the command Mode CALL PEEKV(16364,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T). (Tedious ain't it?) The PRINT A;B;C;D;E;F;G;H;I;J;K;L;M;N;O;P;Q;R;S;T. This should produce 19,88,19Ø,2ØØ,14,49,5Ø,51,52,53,54,55,56,57,48,49,5Ø,51,51. This is the complete program line of 2Ø items but without the line number. Now try CALL PEEKV(16344,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T), and then PRINT A;B;C;D;E;F;G;H;I;J;K;L;M;N;O;P;Q;R;S;T, to get the 2Ø code items immediately above the program line. You should get the following:- Ø,65,Ø,1,Ø,Ø,63,237,7Ø,12,34,56,78,9Ø,12,34,Ø,1Ø,63,237.

The last 4 numbers are clearly the line number entry. The first 2 (Ø,65) can be disregarded. They always seem to appear in this position but have no significance that I can discover. The Symbol Table consists of the other 14 items which are tabulated below with their locations and meanings.

LOCATION NO.	CODE	MEANING
16346	Ø	}Code for numeric variable(arrays 1-7)
7	1	}Number of chars in variable name
8	Ø	}Pointer to location of next entry
9	Ø	}in the Symbol Table. No entry so Ø,Ø
5Ø	63	}Pointer to location of variable name.
1	237	}256*63+237=16365
2	7Ø	}Sign and magnitude indicator -see below
3	12	}
4	34	}2 digits of the same number stored in each
5	56	}byte, exactly the same regardless of the
6	78	}magnitude of the number
7	9Ø	}
8	12	}
9	34	}

The meaning of the 7Ø at location 16352 took some time to elucidate. It is rather like scientific notation(1E6) except that it uses 1ØØ as the multiplier (or divider) instead of 1Ø. The scale of the number indicated by the difference between the indicator and 64. If this is positive then the number expressed as 12.345678901234 is to be multiplied by 1ØØ that many times; if negative then it is divided by 1ØØ that many times. Examples:- 123456.78901234 would have an indicator of 66, and .12345678901234 an indicator of 63.

↗

PEEKING WITH MINIMEM.

How are negative numbers stored? Enter line 10 X=-12345678901234, and RUN. CALL PEEKV(16346,A,B,C,D,E,F,G,H,I,J,K,L,M,N), and PRINT A;B;C;D;E;F;G;H;I;J;K;L;M;N. The display should read:-
0,1,0,0,63,236,185,244,34,56,78,90,12,34

The two numbers 185 and 244 are a bit of a puzzle. I got a clue eventually from the Minimem handbook which explains that the first pair of digits of a negative number are held in 2's complement form but said nothing about the indicator of scale code. So the 2's complement of 12 is 244 (the easy way to get this is to subtract from 256). This is OK for the first pair of digits but doesn't quite work out for the indicator number which is always 1 too great. So in this case subtract it from 255. 255-185=70 which would be the right indicator for a positive number of that scale.

A suggested program for PEEKing is given below. It should start at 1000 or some high number to leave room for the program lines you want to PEEK. Also the entries in the Line Number Table can be distinguished by having a 3 in the most significant byte (left hand location) instead of 0.(4 when you go above 1024). Pressing E and X keys will move up and down the program from your start point,S will let you INPUT a new start point, and D will exit from the program.

```
1000 CALL CLEAR
1010 INPUT"START AT LOCATION
No? ":X
1020 CALL PEEKV(X,Y)
1030 PRINT X,Y
1040 CALL KEY(0,K,S)
1050 IF S=0 THEN 1040
1060 ON POS("EXSD",CHR$(K),1
1)+1 GOTO 1040,1070,1090,101
0,1110
1070 X=X-1
1080 GOTO 1020
1090 X=X+1
1100 GOTO 1020
1110 END
```

Other modules can be used for PEEKing and POKing but there are differences in approach. Extended Basic for instance does not provide the commands CALL PEEKV and CALL POKEV. You could use a machine code routine to provide them the commands then being CALL LINK("PEEKV") and CALL LINK ("POKEV"). You must in that case have the memory expansion plugged in, and then your programs are held in the High Memory part of the expansion memory,and as this is classed as CPU memory you can use the CALL PEEK command already provided to get the program lines and Line Number Table. The Symbol Table stays in VDP memory however so you would still need the machine code routine to PEEKV the Symbol Table. You can also ues the Editor/Assembler but not having one I don't know the details of its operation.

HAPPY PEEKING

[illegible]

JOHN ROE.

7 HARBURY CLOSE
MATCHBOROUGH WEST,
REDDITCH
WORCESTER B98 0EF.

DISK DRIVES TO FORTH

I went to the show not expecting it to be as well attended as it was, and expected a social occasion where fellow Texas owners could sit down and talk. Unfortunately from that point of view I was disappointed, as the show was so crowded, but I was happy to see that the stands were so well attended, and that the show appeared to be a commercial success. At least this should ensure continued dealer support for our machine. On the point of meeting people, would it be possible to arrange for a seperate area where people could relax away from the hurley-burley of the stands and discuss whatever matters they wished, whilst not being too far away from the central attractions. The Manchester site would have suited this purpose if there had been a public address system and a reception point where messages could be passed on to allow those people wishing to meet specific persons could announce that wish over the Tannoy.

Now for some technical points, arising from conversations that I had at the show. It is a well known fact that the hardware for the Texas is expensive, and I feel that a few words for those who are trying to expand their systems would be in order, in an attempt to save them some money. The Texas disc system is very expensive, with the drive itself being sold at one time for some £300. This at a time when I myself bought two similar drives for £250! These drives, by CDC, are fully compatible with the Texas, as are any drives of a reputable make being sold today. As I said to several people at the show, there is no reason for anyone to pay a high price for a disc drive for our machine, as it uses perfectly standard drives, although Texas would have you believe otherwise, as would many other computer manufacturers.

There is one limitation to the drive being considered, and that is that the track to track time of the drive head should be less than approximately 30mS. This is achieved by almost any drive sold today. Beware of anyone selling Teac 50A, as they are old stock, full height drives which definitely do not work with the Texas. I know, I've tried them. Any of the new slimline drives will work with perfect results. As an example, I have just finished fitting two Teac 55F drives to a friend's system, which also has the original Texas drive working in the expansion box, with no adverse results.

The Teac drives mentioned above are 40/80 track drives, switchable from one format to the other by an external switch. To cope with the narrower track width in the 80-track mode, the head must be of a narrower width, and the wider head of a 40-track drive is not bound to pick up the data; errors can thus result. The reverse is not true, the 80-track drive can read the data written on by a 40-track drive. In order to guarantee reliability, keep each type of disk to its own drive.

The above drives were sold by a firm advertising in the BBC magazines, stating compatability with the BBC computer, so those amongst us who also own a BBC can use their drives on the Texas! The slimline drives will fit straight into the Texas box, or can be attached to the outside connector on the rear of the box. In this case the ribbon cable connector will have to be of the correct type. This has to be a 34-way IDC edge connector, the ribbon cable supplied by Texas for the external drive will suffice, although the indexing moulding inside the connector will need cutting away to allow the connector to slip over the edge of the drive pcb.

For prices, one can consult the BBC magazines, but these prices are the cheapest that I have found:

40-track single side	£99
40-track double-side	£115
40/80-track single side	£130
40/80-track double-side	£159

The above drives are slimline drives which consume less power than the full height drives, so that two of these drives can be fitted to the Texas box without modification of the power supply. (They are also [illegible]plied in a metal case, with ribbon cable to suit the BBC, plus a [illegible] formatting disk. The cable is quite long, and needs the connector changing to the edge connector type to allow use with the Texas). The only extra items required would be a d.c. power connector for each drive bought, which you could buy from the drive supplier. The data connection ribbon cable can be either one of the edge connectors taken from the original external Texas cable and crimped onto the internal ribbon cable, or the external cable can be used, with a slot cut in the back of the case to allow the cable to reach the drive. Of course, if you wish to use the drives outside the box, it will be simple enough to connect up using the Texas external ribbon cable. Do not forget to take off the resistor pack on the drive nearest to the disk controller board, should you have more than one drive.

Some people may be wondering why I have mentioned drives other than 40-track single-side, and the reason is that the Texas system will cope with double-sided drives. This may not ap[illegible]ar to be the case if you have the Disk Manager 1 module, but [illegible] will cope with double-sided disks, with one vital exception, th[illegible] being the inability to FORMAT the disk on both sides. This, in my opinion, is an error in the programming of the DM1 which has been rectified in the DM2, but Texas have not seen fit to call in all the DM1 modules and replace them with the proper working version. I am informed that when the DM1 was introduced, TI did not produce a double-sided drive, and thus did not provide the facility of double-sided formatting in the DM1. This is a rather strange policy, as the Disc Controller rom routines do allow for double-sided work[illegible] and formatting, and they were produced at the same time as the [illegible]. As there are ways round the problem of formatting double-sided disks, it can be seen that there is no restriction to the type of drive that we can use on our system.

There are two methods of solving the problem of formatting a double-sided disk, for those who do not have a DM2:

1. Use another computer to do the work. This is the method I used before working out method 2.

2. Obtain a copy of TI-Forth (around £20-£30 from Pete Brooks, Stephen Shaw, or Parco), and use the format command supplied, with modifications as shown later.

In both cases, the first two sectors have to be correctly set up in order for the computer to recognise the disk. In the first method I use the Disk Fixer from Navarone Industries, and in the second method I use a modified version of the Disk-head command in Forth.

Obviously the use of my other computer to format Texas disks will be of no interest to other people, except to show that the Texas disk format is not an unusual one. The method that I now use, and which I will describe, is the use of Forth which is now cheaply available to all TI disk users.

If using only one drive with only side formatted, then to enter the following screens of Forth, one can erase some of the original Forth screens which one is unlikely to use for the time being. These screens can always be copied back from your original disk after formatting two sides of the disk, thus creating more space to work with. The screens that I suggest should be erased are the assembler screens, as these nine screens are not likely to be used by the Forth beginner. The assembler screens are numbered 74-82, so on your BACKUP COPY, type in 74 CLEAR FLUSH and so on, to screen 82. This will give you nine blank screens into which to type the following screens and any others you may find useful at the moment. There are two spare screens at the end of the disk, but this may not be enough to type in the necessary screens without cramping the typing up too much and making it difficult to read and understand the text.

*Type in the following screens of *Forth text, and then save them to disk. Now type in 74 LOAD if 74 is the first screen on which [illegible] have entered the routines. Now type in the word DISKFOR[illegible]. This will start the program (or Word as Forth buffs would say), and you only need to follow the prompts displayed. The program, as supplied, will cater for double-sided disks with any number of tracks, but there is not much error-trapping built in, so please do not enter any silly values such 80 tracks when your drive only caters for 40 tracks, as this could damage your drive mechanism. Error-trapping could be built in, but this program was knocked up to put into practice the results of quite a lot of system-probing.

The program alters the disk fences and writes the first two sectors using the information provided by you. Should you wish to format only one side of a disk, this option is allowed for. No alteration of any TI-Forth screens is necessary, and to change the disk format it is necessary only to enter different inputs to the program, rather than to change the Forth screens with the editor. This is the approach detailed in a copy of the CorComp newsletter which I have been shown, and this is somewhat inconvenient. Should anyone want an explanation of the program, please let either Clive or myself know, and I will include it in any future articles.

To those people wondering if we can use the 80-track drives mentioned above, the answer at the moment is NO! I say "at the moment" because the system-probing mentioned above has included attempts to make the 80-track drives work in the [illegible] track mode, but we have come up against an obstacle inside the [illegible] rom routines. It seems from the research being conducted by my[illegible]f and several friends, Richard Blanden and Stan Dixon, that the rom routines on the Disk Controller card will not allow the filing system to use any more than 40 tracks, even even though 80 tracks can be formatted. (Richard is approaching the problem via assembly language, and Stan is using Pascal). The Disk Fixer from Navarone Industries also seems to have this limitation built in, and so I could not check the formatting on the Texas, but was able to use my other computer and its super disk utility to verify that the Texas rom routines do in fact format 80-tracks if desired. Perhaps this is another slip-up in the programming of the rom? Either that or the policy which guided the programmers is strangely inconsistent. However the research is still going on, and we are attempting to 'patch' the rom routines in an effort to bring the benefits of greater storage capacity to TI users. New eproms will have to be supplied, and possibly a legal word in the ear of TI Inc., but it may be possible to enhance the standard Disk Controller card, and this can't be bad for TI as it could mean increased sales of the Disk Control cards left on the shelves.

A question that I have been asked both at the show and afterwards is whether the disk capacity can be increased by using double-density drives. The answer to this is categorically NO. The problem here is that the hardware will not allow this facility, for the Floppy Disc Controller chip (FDC) used on the TI card is a Western Digital 1771. This FDC is designed to work in the single density mode only. Most of the drives made today have double-density capability, but unless the FDC can operate in double-density mode to supply the drive with information, the double-density capability of the drive is wasted. Without a fairly extensive hardware rework on the Disk Controller pcb, a new FDC such as the 1791, and new software routines in the rom, double-density working is not possible on the TI. However, with two-side working on 40 tracks, we have 180K to work with, and this should be enough to those dedicated programmers amongst us who like to write large programs.

If we are to get such a double-density modification, it seems that we will have to do it ourselves, as I hear a whisper that CorComp have filed for the American equivalent of bankruptcy. It seems a shame that we will not see any of there hardware over here and that we should have to struggle to get the simple interfaces like a printer output, and the slightly more complicated disk interface. When other computers seem to have an abundance of such things on the basic keyboard box.

Moving on from disk drives, I would like to ask how many people have the expanded system, do we have an expanding group of full system members, or are we shrinking? Is even the current lower price of the expansion system enough to put off prospective buyers? It would be nice to know whether or not they intend to expand or simply to use the console alone, and buy another computer system should they desire to have disk and printer interfaces. Perhaps we could have some feedback in the pages of TI*MES?

All this is leading up to the big question: are TI going to re-enter the home computer market, and if they do, will they learn from the previous mistakes, and listen to the needs of the existing users, with their experience of TI computers? Will they do market research and supply the type of computer required, or will they once again take the sort of attitude displayed in the past and supply the type of computer that they think the market should buy, and be thankful that they were able to buy it?

I would like to offer a few suggestions which TI might like to consider, these being opinions formed by my position at the buyer's end of the market:

1. Do not try to 'lock in' the customers to their products, both hardware and software. This applies in the minicomputer market, but as shown by the success of the BBC, Commodore, and Sinclair, the greater the number of outside suppliers for hardware and software, the greater the sales of the basic machine.

2. Do not try to 'over-engineer'. This one will probably be difficult for Texas, as they obviously care about the quality of their hardware. The expansion box and its modular cards are of excellent quality, but will also be expensive to produce, and are certainly expensive to buy! I have seen many examples of computers where the same facilities are supplied in a much smaller space, and sold for much less money.

3. Do not try to be too sophisticated in the design. The use of GPL, and VDP ram as program storage is a clever idea in order to save memory at a time when memory was expensive, but at the same time as the Texas was designed, there were other computers being designed with 'straight' memory maps and they were just as cheap or cheaper than the Texas, and much quicker in operation. The memory-mapping of the Texas is atrocious in the amount of memory wasted on the memory-mapped devices: 8K is used to provide 267 bytes, and for what? To save on a few decoding chips. If there is another valid reason, perhaps Texas would like to let us know?

In spite of the way that Texas have bungled the design of the hardware and so produced a machine which is slow in operation, thus losing a lot of customers, they have produced some good software packages, especially at the 'serious' end of the market. I particularly like the Editor-assembler--the programs, not the manual, which is badly written and full of faults. I also like the Extended Basic, which, while being slow to operate, is a more powerful package than some Basics that I have seen. I echo other people in saying that it was a pity that TI saw fit to provide it as an extra rather than buildinq it into the console-this would have been a good selling point.

I like the machine better since I got the Forth package by Wycove Systems of Novia Scotia, Canada, some 2 years ago-I believe that I am no longer the only owner of this version of Forth. I also have TI-Forth, as you can see from the earlier notes, but have you seen the date on the documentation? 1982! This once again underlines the poor marketing policy of TI, not releasing this beauty of a language for almost 2 years after it was written. Perhaps they were afraid that it would allow too much access to the inner workings of the computer, too easily, to too many people? At long last we can make the machine work at a reasonable speed, without resorting to machine language. I would advise anyone who wishes to do some fast computing on the Texas but still work on a reasonably `high level' of programming to obtain a copy of Forth.

After having had a moan about the existing machine, I would like to say that I have had some very pleasant moments and learnt a lot from the use of this machine. It is by no means a bad machine and has many good points to commend it, not least the high quality of the hardware.

Now on to some thoughts on how I think the next Texas machine ought to be designed. I invite other people to write with their criticisms, in order to stimulate a public debate.

I think that Texas ought to:

No!

1. Forget any thoughts of compatibility with the 4a.

2. Make the computer a full 16 bits throughout-not a 16 bit CPU constrained to an 8 bit bus, unless the 9995 CPU is used, which although it is 8 bits outside, it is 16 bits inside and is said to be faster than the full 16 bit 9900 CPU.

3. Use their own CPU, not, as in the Professional, an Intel 8088. I like the TI 9900 CPU and its instruction set, but most people are not worried what CPU is used, as long as there is plenty of software available (at this side of the Atlantic) and plenty of memory available to run it in.

4. Put in at least 128K of ram with room for more. Memory is pretty cheap these days.

5. Use one of their own Video Processor Chips if they must, but please make it one of the later ones, with 80 column capability. The windowing that Texas incorporate into ma[illegible] of the programs is very clever, but it is not as convenient as [illegible] columns in permanent view.

6. Incorporate RS-232 or its modern equivalent, RS-422.

7. Include a parallel printer port.

8. Disk drive facilities to be included on board, with slimline drives in the case.

9. Make it a clean machine. i.e. allow a choice of disk operating systems on disk, rather than in rom. My own Genie (5 years old) has a choice of at least 6 DOS, even though it has its own Basic in rom.

10. If it is felt that a language is needed in rom, let there be a choice of several languages in rom, to fitted by the dealer at the time of sale. The choice of language would avoid a repeat of such as the Jupiter Ace, which had Forth as its language when many people were unsure of Forth's capabilities.

11. Make the memory map a `straight' affair, so as not to waste memory.

12. Incorporate a cartridge port, as this is a convenient way to load software quickly, but do not make it necessary to use the port for functions such as those performed by the Disk Manager at the moment, for they should really be on disk. DOS functions should be available from inside applications programs or languages, and this is not possible when a cartridge has to be inserted, thus resetting the machine and losing the applications program.

13. Make it easy for professional programmers to write any kind of programs without being tied up in red tape and contracts, this generates a demand for the hardware. Sinclair is a good example of this kind of approach, and look at the amount of software available for his machines. I know of a `C' compiler for the Spectrum for only £26. Sinclair's problem is that he can never deliver the hardware on time.

It may appear from the above list that I am asking for a TI Professional at Spectrum prices, but this is not the case. Machines such as the BBC incorporate many of the features outlined above and sells at an inflated (in many peoples opinion) £400. If Acorn could provide a machine like the BBC with their then limited resources, what should the huge TI Inc. be able to achieve? Machines such as the Professional are competing with the IBM PC, and provide most if not all of the above features, but it is commonly said that the IBM is overpriced, most of the price is for the badge. Somewhere in between there should be a happy medium. The BBC, when expanded, costs approximately £800. Surely Texas should be able to provide a really good, compact machine for that kind of price?

I will leave this subject for now, and would like to start winding up by saying that I am willing to provide a regular column on programming on the Texas machine, if the demand is there. Please let Clive or myself have your views and requirements. Please bear in mind that I like probing around in system routines rather than graphic displays etc, so unless anyone has any particular problems or areas of interest, my programming will be slanted towards system utilities. It would not be an absolute beginner's course, as there plenty of good books to do

that. The two that I can recommend are: Starting Forth by Leo Brodie (ignore the cartoons if they annoy you, but the text is good), and The Complete Forth by Alan Winfield. The two together should set you back about £20, but they are worth it.

Finally, would any of the other Pascal owners please let us know who they are? There are said to be about 10 of us. I know of three, where are the rest of you? What do you do with your Pascal system, have you found any sources of software, and what do are your thoughts on the system? Myself and my friend down the road who also has Pascal would be pleased to hear from you.

I'll sign off now, but look forward to hearing from fellow TI owners and Forthers.

Phillip

Phillip Marsden.

6, Kennerleigh Grove,
Leeds 15,
LS15 8NQ.

ED: A number of group members has expressed interest in Forth, so do write. Anyone wishing to comment on this article must write DIRECT to Phillip at the above address. If you require a reply then we request that you enclose a STAMPED ADDRESSED ENVELOPE.

We all look forward to further articles of special interest.

```
SCR #91

( FORMAT DISK ROUTINES PAGE 1 )
: DISKF ;
: PAGE CLS 0 0 GOTOXY ;
0 VARIABLE NAME 8 ALLOT NAME 10 32 FILL
0 VARIABLE DRIVES
0 VARIABLE TRACKS
0 VARIABLE SIDES
0 VARIABLE THIS-ONE        0 VARIABLE BASICFLAG
: INPUT QUERY 32 WORD HERE NUMBER DROP ;
: NAME? CR ." DISK NAME ? " QUERY 32 WORD HERE COUNT NAME SWAP
  CMOVE ;
: DRIVES? CR ." NUMBER OF DRIVES ?   " INPUT DRIVES ! ;
: TRACKS? CR ." NUMBER OF TRACKS ?   " INPUT TRACKS ! ;
: SIDES? CR ." NUMBER OF SIDES ?    " INPUT SIDES ! ;
: WHICH? CR ." FORMAT WHICH DRIVE ? " INPUT THIS-ONE ! ;
-->
```

```
SCR #92
 ( DISK FORMAT ROUTINES PAGE 2 )
 : READY? CR CR ." PLACE YOUR BLANK DISK IN DRIVE " THIS-ONE @ .
   CR ." AND PRESS ANY KEY " KEY DROP ;
 : MODDISK_SIZE 225 TRACKS @ * 100 / SIDES @ * DISK_SIZE ! ;
 : MODDISK_HI DISK_SIZE @ DRIVES @ * DISK_HI ! ;
 : NEWTRACKS TRACKS @ 13706 ! ;
 : NEWSIDES SIDES @ 33616 ! ;
 : FORMAT THIS-ONE @ FORMAT-DISK ;
 : TITLE ."            FORTH DISK FORMATTER" ;
 : Y/N KEY DUP 89 = IF DROP 1 ELSE 78 = IF 0 ELSE MYSELF ENDIF EN
DIF ;
 : BASIC? CR ." WILL THIS DISK BE USED WITH BASIC ? " Y/N DUP BAS
ICFLAG ! IF NAME? ENDIF ;
 : FINISH CR CR ." FORMATTING FINISHED " CR ." FORMAT ANOTHER ? "
    Y/N ; -->
SCR #93
 ( DISK FORMAT ROUTINES PAGE 3 )    HEX
 : DISK-SETUP THIS-ONE @ DISK_SIZE @ * BUFFER
   DUP NAME SWAP 0A CMOVE     DUP 0A + DISK_SIZE @ 4 * SWAP !
   DUP 0C + 0944 SWAP !       DUP 0E + 534B SWAP !
   DUP 10 + 2000 SWAP !
   DUP 12 + 26 DISK_SIZE @ 2 / + 0 FILL
   DUP 38 + 0300 SWAP !
   DUP DISK_SIZE @ 2 / 38 + DUP ROT + SWAP 100 SWAP - FF FILL
   DUP 100 + 100 0 FILL 200 + 200 E5 FILL UPDATE FLUSH ;
 : BASIC BASICFLAG @ IF DISK-SETUP ENDIF ;

 : DISKFORMAT PAGE TITLE CR CR TRACKS? SIDES? DRIVES? WHICH?
   BASIC? READY? MODDISK_SIZE MODDISK_HI NEWTRACKS NEWSIDES
   FORMAT BASIC FINISH IF NAME 0A 20 FILL ENDIF FORGET DISKF ;
 DECIMAL
 : ZX SWAP DO I BLOCK 2- 30 DUMP  PAUSE LOOP ;
```

BRUSSELS SPOUTS

Sorry to have missed last issue. (Which was excellent incidentally)
Following Peter Brooks glowing testimonial in TI*MES No 5 I must start with my Low Resolution Plotter - inspired by his plotting work. For newcomers Low Res. means that each point will be 1/4 of a screen position - like the PLOT command on a ZX81 or,I think, a TRS80 amongst others.
First of all I'll list the guts of the routine then show a few examples of use.

```
100 CALL CLEAR
110 DIM G(24,32),IN(32),D$(4), B(16,4)
120 FOR I=1 TO 8
130 B(I,1)=I+7
140 B(I*2-1,4)=I*2-1
150 NEXT I
160 DATA 00000000,060F0F06,60F0F060,
66FFFF66
170 FOR I=1 TO 4
180 B(I,2)=I+3
190 B(I+8,2)=I+11
200 B(I*4-3,3)=I*4-2
210 B(I*4-2,3)=I*4-1
220 READ D$(I)
230 NEXT I
240 FOR I=1 TO 4
250 FOR I1=1 TO 4
260 CALL CHAR(N+32,D$(I)&D$(I1))
270 N=N+1
280 IN(N)=N
290 IN(N+16)=N+16
300 NEXT I1
310 NEXT I

400 ..REM..DERIVE PLOTS........

600 REM PLOTTING SUBROUTINE
610 Y=IN(R*.5)
620 X=IN(C*.5)
630 H=B(G(Y,X)+1,C+2*R-2*X-4*Y+4)
640 IF H<1 THEN 670
650 CALL HCHAR(Y,X,H+32)
660 G(Y,X)=H
670 RETURN
```

The Plotter works quickly ,in fact as Peter said,it is as quick as some of the TI subprogs.eg SIN,SQR etc.The speed can be demonstrated in various ways.
eg.1.A Simple Loop

```
400 FOR R=1 TO 48
410 FOR C=1 TO 64
420 GOSUB 600
430 NEXT C
440 NEXT R
450 GOTO 450
```

eg.2.A Keyplot   (8 directions -Press Keys W,E,R,S,D,Z,X,C)

```
400 R=24
410 C=32
420 CALL KEY(0,K,S)
430 IF S=0 THEN 420
440 R=R-((K=69)+(K=87)+(K=82))*(R>1)+((K=90)+(K=88)+(K=67))*(R<48)
450 C=C-((K=87)+(K=83)+(K=90))*(C>1)+((K=68)+(K=67)+(K=82))*(C<64)
460 GOSUB600
470 GOTO 420
```

Press 1 of the 8 keys to begin

Final example is a circle to answer the query of A.Mohamad in TI*MES 6.
The simple mathematics gives this program to draw a circle centred on RC,CC and with radius RAD.(Note it's the same for any Plotting resolution).
eg.3a.Simple circle

```
400 RAD=10
410 RC=24
420 CC=32
430 FOR J=0 TO 8*ATN(1) STEP 1/RAD       (Note 8*ATN(1) is same as 2*PI in EB)
440 R=RC+INT(.5+RAD*SIN(J))
450 C=CC+INT(.5+RAD*COS(J))
460 GOSUB 600
470 NEXT J
480 GOTO 480
```

You will find that this produces a flat circle (more of an egg!) because the screen representation of an 8x8 pixel is actually rectangular.This can be cured by changing line 450 to C=CC+INT(.5+.75*RAD*COS(J))
The routine also runs fairly slowly mainly due to deriving all those SINES and COSINES and the following,although longer,is much quicker needing SINES and COSINES for only 45degs instead of 360.

```
400 to 420 as above
430 FOR J=0 TO ATN(1)
440 SN=SIN(J)
450 CS=COS(J)
460 R=RC+INT(.5+RAD*SN)
470 C=CC+INT(.5+.75*RAD*CS)
480 GOSUB 540
490 R=RC+INT(.5+RAD*CS)
500 C=CC+INT(.5+.75*RAD*SN)
510 GOSUB 540
520 GOTO 520
530 END
540 GOSUB 600
550 R=2*RC-R
560 GOSUB 600
570 C=2*CC-C
580 GOSUB 600
590 R=2*RC-R
595 GOSUB 600
596 RETURN
```

(Sorry about line nos.!)

Console Timings

Stephen Shaws comments in last TI*MES were interesting (as ever) and i find my console works at the same speed as his (actually 53 to 58) BUT with Mem. Exp. attached nos. become 49 to 53.This quite large difference on one console absolutely proves that you cannot rely on timing (even in your own programs!).

Quicksort

I needed a Sort routine recently and wading through piles of Mags etc. I found a bewidering asSORTment.I needed a sort SORT!
It was soon obvious that some were painfully slow and that one called (appropriately!) the Quicksort in an old 99'er was the best option.
This however was an awful example of 'Spaghetti Basic' and I managed to rewrite it to be about 1/2 as long and around 10% quicker.
The example here will sort random numbers in array A()and PRINT results.

```
40 DIM A(1000)
50 N=100
60 FOR I=1 TO N
70 A(I)=INT(RND*100)
80 PRINT A(I);
90 NEXT I
95 REM....SORT ROUTINE.....
100 P=1
110 S(P)=1
120 L(P)=N
130 R=S(P)
140 T=A(R)
150 Z=R
160 FOR I=R+1 TO L(P)
170 IF A(I)>=T THEN 210
180 A(Z)=A(I)
190 Z=Z+1
200 A(I)=A(Z)
210 NEXT I
220 A(Z)=T
230 IF L(P)-Z<2 THEN 260
240 S(P)=Z+1
250 GOTO 270
260 P=P-1
270 IF Z-R<2 THEN 310
280 P=P+1
290 L(P)=Z-1
300 GOTO 140
310 IF P THEN 130
315 REM......PRINT RESULTS......
320 FOR I=1 TO N
330 PRINT A(I);
340 NEXT I
```

If you expect almost all the items you are sorting will be different then the routine runs slightly quicker without the "=" in line 170.

A Short One

If you have E.B. and Mem Exp or Mini Memory here is a small piece of code which will allow you to test the speed of Benchmarks or parts of programs.

```
100 CALL INIT
110 CALL LOAD(-31879,0)
.....Section of program to be timed.......
200 CALL PEEK(-31879,A)
210 PRINT A/60              (A will be printed in seconds.)
```

How does it work?. -31789 is one of a number of registers in the TI that increases by 1 at each Video Interrupt (60th of a second).Loading 0 and then PEEKing to see how many 60ths have passed you therefore have an accurate timer.The only catch is that it counts up to 256 then goes back to 0 so it is only effective up to 4.2 seconds.(I think there will be a 16 bit counter some where.Does anyone know where?)
You could use this to test how long say 20 of each E.B. operation takes to learn which are fast and which slow.(Test empty loop first for differences)
You could also embed this code temporarily in a prog to see whether a small change makes that part any quicker.

Go Forth and......?

I have noticed the word Forth creeping into TI*MES e.g. S Shaw,which pleases me as earlier this year I bought the Wycove Version from Canada.I use a Cassette Version (I don't have discs) and also load it through Ext Basic (I don't have Min Mem or Ed Ass).You do however need 32k Mem Exp - and also I would guess more patience and discipline with a Cassette system.

Without overstating it the language turns the 4A into quite a fast and powerful computer with most of the routines that even I write being about 20 times as fast as the TIBasic equivalent.
I'm now convinced that I will never bother with Assembly Language as excellent programs can be developed in Forth just about as quickly as Basic.

My feeling ,however,is that Forth is not for every keyboard hacker as you need a pretty good grasp of computing to use it.This might explain why the Jupiter Ace never really caught on.Compared to Assembly Language ,though,it is a piece of cake and I'm not nearly as surprised as Stephen Shaw that he hasn't had a single UK written Assembly prog sent to him.These are jobs for full time programmers and even they often write in higher level first (eg C) and then use Compilers.Apparently many of the Atari modules are actually written in Forth.

If anyone is interested in Wycove Forth,which I think is an excellent version, write to Wycove SystemsLtd. P.O Box 499,Dartmouth,Nova Scotia,Canada B2Y 3Y8
For $50 (incl p&p) you receive both Cassette and Disc and a manual.
You might also drop me a line as I could help you in the early days and point out a couple of minor bugs in the manual.

How quick is TIFORTH?I have timed Wycove against the BenchMarks in PCW(Feb 83) and will publish next time.

Please write to Clive if you have Forth (state version!) as I might include some in my column but there's no point if there are only a tiny band of users.

Stop Press

Just got back after managing to get to Users Show in Manchester.Well done Clive and Audrey! It was great to see all those Users together.You'll have to give us more room next year.

Au revoir mes amis.
   Contact me at either:--

Ian Swales.

c/o 2,Blaydon Drive
    Marske
    Cleveland
    (England)

or. A.Van Dycklaan 43
    1980 Tervuren
    Belgium
        (Stamp is about 20p I think)

SCENE U.S.A.

TI*MES EXCLUSIVE

We bring you all the NEWS and more for your TI99/4

U.S.A. TI99/4A USER GROUPS

We have received many letters from U.S.A. TI Users groups from all over the States. We list the name and addresses. Those of you that really would like to receive special information regarding anything and everything on the TI99/4a.
The fall of the pound may mean that cost of subscriptions would be high, however if you want to get to know a group you can make the initial cost a worthy investment. There are a number of USA TI Users groups that presently give good value in a newsletter.
Please only write direct for more information:-
The LA99ers group publish a monthly newsletter which is chock-full of info with quite alot on FORTH. They link in with S.F.V. TI Users. The address of LA 99ers is P.O.Box 3547, Gardena, CA 90247-7247.
DON VEITH who was once part of LA99ers is now President of NATIONAL 99ERS. A 16 page newsletter for the more advanced TI Users. Address is 3535 SOUTH H STREET, , BAKERSFIELD, CALIF 93304.
If you are after a group newsletter that is neatly produced then MSP 99 Users Group, P.O.BOX 12351, ST PAUL MINNESOTA 55112, is the address to write to. Hi JOEL GERDEEN.
*** Again in this issue we devote a number of our pages to **Jim Peterson** of **TIGERCUB.** He is fast becomming the most popular personality of the TI Users-groups around the world. His programming skills for the unexpanded and for that matter expanded TI99/4a are published in many TI User group newsletters. You really must try some of the excellent Basic programs TIGERCUB SOFTWARE produces. You will not be unhappy. Thanks to Jim his TIPS will continue to enrich all TI Users young and old.
We have just receved the latest issue of HOME COMPUTING MAGAZINE (H.C.M.). I was delighted to find that advertisements are no longer included in the magazine. H.C.M. have been under alot of pressure from TI99/4a Users recently, because of this the content of H.C.M. has given favour to the TI99/4a. My own personal view of the new 99er is that they now set a very high standard in Computer magazines. The content is NOT 100% TI99/4a but it is interesting to see how VIC, IBM PC jr, APPLE and C64 compare in operation.
We have said not to renew your subscription because of H.C.M. not meeting present subscriptions, some GOOD NEWS they have in fact extended our subscription to cover TWELVE Issues. If you have not received your twelve issues then write DIRECT to Gary Kaplan Editor of H.C.M. and find out why. They should try and put matters right. If any member has had a problem let us know. It is possible that legal representation can be made on your behalf through an American TI Users group. You will need proof of joining which is indicated on the address label of H.C.M.
I.U.G. DOES ANYONE KNOW WHAT IS HAPPENING OUT THERE? No newsletter since JUNE!!! What is going on?
We have received a number of complaints regarding CRAIG MILLER and the non arrival of MILLERS GRAPHICS Newsletters, its no joke taking money and then not keeping up the flow of publications. Hope to publish some answers next issue.

Thank you for your support

Your Software Library has probably some of the best programs ever to be found
Thanks to ED YORK and the Cin-Day TI Users, Houston TI Users, Terrie of LA 99ers
also John Clulow New Horizons TI99 Users

Howard writes again !

Howard writes again !

FROM THE DARKEST DEPTHS OF MANCHESTER AT AN ADDRESS NOT TOO DISSIMILAR FROM OUR HEROES AT CORONATION St. HOWARD GREENBERG IS GENERATING YAWNS APLENTY AS HE PREPARES TO BORE THE TI99/4A OWNING PUBLIC BY PUBLISHING YET ANOTHER EDITION OF HIS MEMOIRS AS HE WRITHES AGAIN.

(Or was it the reader that was writhing whilst I wrote ???

To err is human and at Honda, (where they make well designed cars built to last only slightly longer than the guarantee) engineers are encouraged to make as many mistakes as they want. Only one crime is unforgivable ; making the same mistake twice. For that no longer qualifies as research, but as stupidity and rightly so.

With bated breath (and stifled yawns) you'll all be wondering what mistake I've made this time. Has the idiot blown up the 99 by trying to control his central heating by dropping it in the boiler ? Well no (but the idea did occur !), I'm simply adding my own four ha'pennyworth to the comments that have been bound to have come in from other readers who attended the first T.I. owners convention in Manchester on November 3rd.

To those of you who were making their first visit to Manchester, I'm sure you'll now appreciate why it's called the Rainy City.

So what was my mistake on the 3rd November ? The answer is that I wasn't prepared enough for the incredible number of people who turned up. Or put another way, the event was too successful. Full credit to Clive & Audrey, for their determination to organise the event. I had a very successful day, I'm a compulsive talker and too many people didn't have enough time to talk with me. Amongst those were people who'd made the trip from as far north as Aberdeen and other places in Scotland that I can't pronounce) only to have my hand shaken with "Hi, I'm so-and-so", "Great, nice to meet you at last, I'll talk in a minute". But the minute never materialised, and I think that some went home frustrated. Some went home very happy, if I ever find out who it was who helped themselves to several of my modules, I'll wring your necks and enjoy doing it. Some folks were very patient and courteous, like the gentleman from Bristol, who queued for around 20 minutes only to prove he was a gentleman when his turn came by letting a lady in ahead.

I missed out too. There were a lot of people whom I was looking forward to meeting, with whom I only had the briefest of introductions, before someone elbowed in. But, thank you all. In particular, Russell who helped me in and out with all the gear, Ivan and John who helped me home (No I wasn't drunk) and Matthew who helped on my stand. And as Hitler probably said to Himmler when they first invaded foreign soil, "we must do this again sometime."

Onto more serious subjects.

There is a growing list of items which I can no longer obtain for re-sale. Although T.I. seem to have been prolific in producing vast quantities of software prior to their final withdrawal from the consumer computer market, the pit is not bottomless. The worst affected item is Extended Basic. Many people commented on my letter published in Computer & Video games magazine, where I commented that I would sell ExBas for as long as it was available, but make no attempt to make it cheap. Now the price doesn't matter. I can no longer obtain them at any price. Mini-Memory, I won't stock since the $ did its climb. Other software which (as far I'm concerned at least) has now gone for good, includes Personal Record Keeping (a sadly missed and always underrated product), Chess, Parsec, Speech Synthesisers and RS232 cards. At least as far as hardware is

Howard writes again !

concerned, suitable substitutes are springing up, but software is more difficult. As yet, to my knowledge, no-one has procured the necessary licences to make products formerly made by Texas Instruments. Independant producers do occasionally come out with a new software item, but it's usually a game. Nothing wrong in that of course, and all credit to them for having the courage to make anything for the T.I. but there is more to life than despatching aliens to an early grave.

Navarone Industries in particular have been the most prolific producers of non-games software, but most of it is on disc, which rules out 80% of the U.K. market. Since they do make cartridges, they really ought to think again. And, on the subject of Navarone, my apologies to everyone who ordered a Widgit around November. I can't remember an item that proved so in demand that I was not only selling them faster than I was buying them, but faster than I could buy them. Which was Navarones fault, because every time I ordered them, they didn't turn up. Hopefully now, the situation has improved and I can supply off the shelf.

Once again, I'm back on the subject of Mutliplan. I've now been presented with an updated version of Microsofts answer to Visicalc, and I'm considerably more impressed with it than I was initially. Getting started with the program is still daunting. The manual seems to have been written by Manuel (Fawlty Towers) and has probably been improved by the same genius resposible for the clarity of the Editor/Assembler manual.

Having virtually forgotten everything I learned about the program, it was like learning to swim all over again. (The analogy isn't that far removed, since you can feel like you're drowning if you don't know where you're up to.) But it is now worth persevering with. What makes the product so useful is its versatility. I've now set up four sheets which I can use for placing orders and working out the final balance of that order to fit in with a given budget. Using the "what if" feature that make a spreadsheet so useful, I can now work an order up or down to a given dollar. I can even work it from dollars to sterling to fit in with the days exchange rate should I so wish. Other applications include keeping an accurate stock and stock value picture and the more boring book keeping functions that are a regular and unpleasant part of runing any business. Multiplan may not make these a pleasure, but at least I now know I get it right!

As mentioned in a previous article, Multiplan is universal. Try out a version on any machine and it will be <u>very</u> similar to the one on the T.I. I spent a very funny morning training my accountant how to use it. Some whizz-kid computer salesman had sold them an £8,000 system and they hadn't the foggiest how to use it ! There was little me, with no financial training at all, showing all these high powered types, how to use their latest toy. I didn't even get a discount on my bill from them, miserable swine.

So what's new about V2 Muliplan ? Not much, but enough to make it if not a pleasure, then at least less of a chore. It's still much too slow, particularly if you don't turn off the Re-Calc before you start work. But once it is off, you can enter data in cells and zip from cell to cell with much more speed than formerly. On V1, It was more than little tempting to turn the thing off and get out the Casio Calculator. Now, I can enter a column of data (say 70 items), give it the instruction to Calculate and then in less than thirty seconds, the answer is there. The increase in speed isn't staggering, but it is noticable. And that is enough to transform the program from something nearly unusable, to a worthwhile investment.

On the subject of Microsoft, has anyone seen their chairman ? Talk about

Howard writes again !

age being no barrier in the U.S.A. The man (Bill Gates), looks as though he isn't old enough to be allowed out without his parents permission.

The latest interest in disc drives seems to be in having two. Now three-oh's may be better than two-oh's, but two are certainly better than one. The advent of the half height drive has meant that a number of people are now having their boxes modified to take the new type of drive. I've done my own box, and a number of others, so far no problems. Whilst the box won't power two full height drives, the new type of drives use half the power, so they don't place an undue strain on the hefty current as supplied by the box.

I have several new items coming in from the U.S.A. Possibly one of the most interesting is the Basic Conversion Kit. I haven't received this yet, and if you're reading this text instead of an updated version, then I didn't have time to comment on it before I had to send my copy out. The B.C.K. is NOT a compiler. That idea has been tried out and it doesn't work. At least not well enough. The B.C.K. is designed for those who can program competently in Basic or ExBas, but has trouble in making the transition to Assembly Language. There is a fair amount of work to be done in making the conversion. The object of the kit is to make the job simple. It does mean a great deal of entering, but it is accurate. As always, I imagine the problems will be with timing. Since machine code runs several hundred times faster than Basic, it's likely that without delays built in, a program converted without delays could be over before you've assimilated the screen ! I'll be offering the Basic Conversion Kit for sale, price will be high, but it should prove a worthwhile investment to competent programmers. This could finally be the opportunity for a flood of Mini-Memory based tapes of games and other programs in machine code.

There are times when you become so used to the terminology involved in computing, that you forget that some people are still beginners. Which could be better than me, since I'm at the "A little knowledge is dangerous" stage. I've now had a couple of people asking me "What's a screen dump"? It sounds painful, with visions of T.V. sets being dropped in the council skip. (Or would they go in the bottle bank since they're glass? The object of a screen dump program is that the contents of the screen are *dumped* onto a printer, giving you a hard copy of what was on the screen at the time. This process works with varying success on the Texas, since it isn't possible to dump a module (which is what we'd like to see) onto the printer. Most of these programs use either Basic or Extended Basic to drop a Basic/ExBas program picture to an Epson compatible printer. The main difficulty is in reproducing the shades of colour. Since most of these programs change all pixels to on or off and then do the printout in that fashion. It works, but not very satisfactorily. The results are totally black and white with no shades of grey.

Enter a freind, who shall be nameless. Starting with a non-compatible printer, and no program to modify, so he was writing from scratch, he's mangaed to write the most useful screen dump I've seen yet. So far, I've been given dumps of many of the T.I. modules, plus a few other things. He's now changing the program to work with this printer, and when it's okay, I'll have it for sale. The program isn't perfect, since at present sprites can't be copied, nor can screen shots in Bit-Map mode. This rules out most of the MBX modules and a few of the Hi-res games (such as Buck Rogers), but I'm still very impressed. Incidentally, anyone who's interested will need the following : Editor/Assembler, Disc system and 32k R.A.M. RS232 and an Epson compatible printer. (Shinwa/Quendata will do, I don't know about Star.)

Because, new items are appearing in great profusion, and old ones are disappearing at the same rate, my advice is - IF YOU WANT SOMETING AND DON'T SEE IT ADVERTISED, ASK. I MAY HAVE IT, OR I MAY BE ABLE TO GET IT.

I hope you all enjoyed a happy Xmas and wish all well for 1985.

Howard Greenberg

BABBLE ON TI-LINES

The problem this time is not that I'm stuck for something to say, but that I'm stuck for what to leave out. Between each issue of TI*MES I produce three issues of TI-LINES, the monthly newsletter of OXON TI USERS, which started out as a club and is now a sort of business, owned and run by me.

Those issues cover a large amount of wordage, and in the past Uncle Clive used to stick the occasional article into TI*MES, but now the amount of information is so great that it is difficult to know what to do.

What complicates things a little is the fact that I cannot simply copy any or all of the material in TI-LINES, as from this issue TI*MES is changing format. I now have to fit whatever I write into an explicitly-defined space on special paper, so that realistically the only articles I can include are those which I can reformat using TI-Writer. As it happens, the width restriction of about 72 columns is what I use as standard in TI-LINES anyway, but the length is slightly different.

I shall manage something, anyway.

THE NEWS

Recently PAUL DICKS passed the control of the [redacted] Software Collection over to me, and I am up to my neck in work, mostly trying to finish the awesome task of recompiling the Catalogue. I know that some members have been waiting a very long time indeed (having sent in their 50p for a copy) and if any of you are reading this, my apologies for the delay, and I will be tackling it again in the New Year. I am away from Dec. 14 to Jan. 7 taking a much-needed rest, and my batteries should have been recharged by the time I return. At least I will be able to get to bed at a reasonable hour (4.30 a.m. the other day!).

What follows is a brief summary of some of the items which have appeared in TI-LINES since JULY 1st 1984 (up to DECEMBER 1st).

Issues 4, 5, and 6 (JULY to SEPTEMBER) were mostly taken up with a detailed analysis of the Speech Synthesizer, in particular the analysis of the make-up of the raw speech data - the stuff you can retrieve with CALL SPGET() through either the Speech Editor or Extended BASIC modules.

Since the series finished, some doubt has been thrown on its accuracy, since it was based on the TMS 5220 chip when the EDITOR/ASSEMBLER manual recommends the manual for the 5200. What causes the confusion is the fact that the Speech Synthesizer chip is the TMC 0285 - the same as in the SPEAK 'N SPELL game. No-one in Lubbock will tell me if there is a significant difference between the chips, - in fact, no-one will even tell me if I have been decoding the data correctly. All I get is PR eyewash. The degree of uncertainty notwithstanding, this is how it seems to work:

Take one valid entry in the resident vocabulary - say the letter 'A'. Use CALL SPGET() to retrieve the string of data for the pronunciation of that letter. What people usually do now is to scan through the string, printing the ASCII codes of all the characters - it always begins 96, 0, and then the number of characters following (for 'A' is 28). Other writers then successively truncate (shorten) the string from the right hand end and add the result onto the end of another similarly-cut string which is then passed to the synthesizer via CALL SAY().

However, what you can also do is to divide the string up into its true component parts: ENERGY, REPEAT, PITCH, and the REFLECTION COEFFICIENTS K1 to K10 (in some publications, K0 to K9). The VOICE SYNTHESIS PROCESSOR manual for the TMS 5220 gives the raw details. What it does NOT do is to tell you how the data is stored in the Speech Synthesizer.

I have uncovered what I think is the answer; only time (or TI) can tell. Once you have these components, you can experiment away to your heart's content, altering the pitch of existing sounds, playing with CONTOURS (e.g., the changes in pitch over a period of time), and even putting together different components to make up totally new words or sounds. The latter item is probably most important: you could, with a great deal of work, produce your own crashes, bangs, pops, whistles, etc.

Take the string of ASCII coded characters, and translate their ASCII codes into binary. 167 (first code) is 10100111 (include leading 0s to make the length up to 8 bits). Now perform a left to right 'mirror', thus reversing the sequence of bits: 11100101. Do this for all of the characters until you have a string of (in this case) 28 x 8 bits. Join 'em all together to get 1110010101010001....etc.

Now chop the string up according to a set of rules: first 4 bits is the ENERGY value. Next single bit is the REPEAT indicator. Next 6 bits is the PITCH value. The following 5, 5, 4, 4, 4, 4, 4, 3, 3, 3, bits make up the REFLECTION COEFFICIENTS. Translate the binary subgroups back into decimal and you have a set of values which are then used to look up other codes in a table. Those codes lead eventually to the actual data used to control the filter which produces the speech. (E.g., the Pitch in cycles per second or HERTZ).

There are a few special cases: if the ENERGY value is 0000, then this indicates an 'inter-word' or 'inter-syllabic' pause, and no other data is required. You would then examine the remainder of the string for another ENERGY group. If ENERGY is 1111, this is the 'stop code', which signals the end of speech. If the REPEAT bit is 1, this indicates that the REFLECTION COEFFICIENTS remain what they were last time, so all that is needed is the 6 bits for the PITCH. You then start again with ENERGY etc. If the PITCH value is 000000, this signifies an UNVOICED sound, and only the first four REFLECTION COEFFICIENTS need be specified, so you only decode the next 5, 5, 4, and 4 bits, and then start from ENERGY again. Each decoded bunch of values from ENERGY to REFLECTION COEFFs counts as one FRAME. The maximum number of bits in one frame is 50, and this is a VOICED FRAME.

The REFLECTION COEFFICIENTS are related to positions in the human vocal tract, and are very complex to study, let alone explain.

In TI-LINES, the series showed how to alter the PITCH of pronunication of the letter 'A' so that it dropped an octave or so, which involved altering all the binary values and then re-encoding back up to ASCII, and then passing the modified string to CALL SAY().

Two articles showed firstly how you might uncover any Undocumented Subprograms, and secondly that there was one in TI BASIC. It has a 'name' which is a single ASCII character - 3 - and can be obtained without any expensive additional hardware through an editing technique presented during the CONTROL & FUNCTION KEY series. Information on this subprogram was provided by RICHARD BLANDEN, who has since also provided details of a whole stack of additional subprograms resident on the DISK CONTROLLER, and some on the PRK and STATS modules - in addition to the CALLs already known (called ENHANCED BASIC - and this is being covered at the moment in TI-LINES). Richard has been a mine of information on these undocumented facilities, and I hope to get the full story from him in the New Year.

IAN SWALES provided the raw data for a table of timings of commands in both TI and EXTENDED BASIC, which was published in issue 6. I regret not having met Ian at the Manchester meeting, but if Uncle Clive and Auntie Audrey can manage it, there will be plenty of opportunities for everyone to meet everyone else in the future. I hear whispers of a possible get-together at Easter, which, come Hell, High Water, or Hemma Roids, I will attend.

Issues 7, 8, and 9 of TI-LINES carried the early part of a series on the additional PRK/Stats CALLs, known as ENHANCED BASIC to distinguish it from TI BASIC. These CALLs allow you to use ACCEPT AT and DISPLAY at equivalents, complete with INPUT VALIDATION. They go on to provide a facility whereby you can either create and manipulate a database along the lines of the PRK/Stats files, but with increased power, or modify or process existing PRK/Stats files - for example, making use of a much faster SORTING ALGORITHM to organise a large file (the built-in SORT on the module is very, very primitive).

You can both LOAD and SAVE files in PROGRAM FORMAT, which looks and sounds just like a BASIC program, and it is obviously much more powerful than the INPUT and PRINT with OPEN and CLOSE which TI BASIC and EXTENDED BASIC support.

You could also write a very, very powerful text or graphics Adventure using these commands, although a disk filing system would make it all run that much faster.

Issue 8 contained the bare bones of a FILE RECOVERY process for disks which does not require access to machine code. I finally managed to reproduce the method which I accidentally stumbled across a long time ago, and presented it for TI-LINES readers to expand upon. I have not heard anything since - I don't know if this is good or bad.

Essentially all it involves is opening a very large RELATIVE, INTERNAL, FIXED 255 file (specifying a number with RELATIVE to reserve the greater part of a disk), and then writing something small to the last record in the file. The effect of this is to 'open up' the whole of the file area, enabling you to LINPUT all the 'records' which currently contain most of the information which you inadvertently deleted or lost.

Issue 8 also carried a short piece about ECHOING, the technique whereby what you type at the keyboard is reproduced on the screen (no, not the machine code variety). This was in response to a request seen in an issue of TI*MES where a reader wanted to imitate the Cassette Operating System and have his CALL KEYed key presses placed at the right hand end of the screen, as in CHECK PROGRAM ? (Y/N)   Y

It is actually fairly simple to provide such a facility even in TI BASIC - all you need to do is to set up a PENDING PRINT CONDITION and then use CALL KEY followed by another PRINT of the character of the key pressed.

A pending PRINT condition is set up by terminating a PRINT line with either a comma or a semicolon (, or ;) so that the computer is told to wait on the current screen line (subject to certain provisos) until given another PRINT instruction. You can then use the TAB() function to print the character of the key pressed (after validation, of course) at the 28th column.

```
100  PRINT "SELECT 0 - 9";
110  CALL KEY(0,K,S)
120  IF (K<48)+(K>57) THEN 110
130  PRINT TAB(28);CHR$(K)::
140  GOTO 100
```

This is pretty pointless, but it should give someone an idea or two.

I had a request for more information on CALL JOYST(), so issue 8 also carried a short article on designing routines to use that command. The article ran to some 7 A4 sides, so maybe it wasn't that short. Either way, it is too long for me to include here, although Uncle Clive might include some of it elsewhere in TI*MES - it depends on (a) its appeal, and (b) whether he can fit it in (as the actress said to the...oh, never mind).

Both issues 8 and 9 were late in getting out, and whether they were worth waiting for, nobody has yet said. From the silence, I assume not.

Issue 9 carried an short piece in the Editorial about GPL, TI's GRAPHIC PROGRAMMING LANGUAGE, about which I now know a little more. I have scouts out looking for manuals, and sooner or later they will turn something up. I know that the manuals are around, but not why TI will not sell one or even admit to their existence. It seems odd that a firm who are heavily into Education and Information should be so reticent about an implementation of TMS 9900 code which is pretty powerful, to say the least. I hope that they change their policy quickly.

Also in issue 9 was an item on CALL KEY - did you know that setting the computer into PASCAL MODE with the dummy command CALL KEY(4,K,K) means that INPUT doesn't permit the usual editing keys to function ? You have to use other keys to BREAK, move LEFT and RIGHT, ERASE, INSERT, DELETE, etc. It might have some uses, although I can't really see anything very practical coming out of it.

As a result of talking to IVAN NIBUR in Manchester, I played around with OLDing programs and came up with some odd results, which formed the basis for an article in issue 9.

Ivan asked me how he could get the 4A to merge programs together, as he had found that he could load two programs into his machine and have them both present but only be able to access one.

I told him that it was an illusion - if he had OLDed a program, exited after an error in loading, checked the program in memory and found the original still there, then the second program had obviously not even begun to load.

However, I was able to demonstrate that what he said was essentially true: I put one program in memory, then successfully OLDed another. On receiving the DATA OK PRESS CASSETTE STOP, PRESS ENTER message, I pressed E instead, to exit. The 4A told me to CHECK PROGRAM IN MEMORY, which I duly did, and found the original program still intact. Both programs were quite small, only a few lines, so I tried again, this time with much larger listings.

Something did happen. The original program was still there, but it had been 'interfered with' (call the Pleece!) and was 'corrupted' - i.e., some vital data had been overwritten and the original would not LIST properly, let alone RUN.

TI BASIC is renowned for taking X minutes to OLD a program, and then nearly X minutes to apparently decide that everything is OK. What it may in fact be doing is to transfer the correctly-OLDed program from one area of memory to another - writing over the original program in the process. It may yet be possible, through either MiniMemory or Editor/Assembler, to 'merge' programs in TI BASIC in some way.

Future issues of TI-LINES will discuss the application of ENHANCED BASIC to a filing system by PAUL KARIS and FRANC GROOTJEN; the additional, additional (yep, twice) CALLs in PRK/Stats; the black art of DEBUGGING; ALLEN BURT's use of the TI-Writer Editor (a man after my own heart - down with the Formatter!), which also appear in this TI*MES; the beginning of a series on TMS 9900 Assembly Language; possibly something on TI Forth; a clever circuit from CENTRONICS via MALCOLM HEDLEY; a LOAD INTERRUPT SWITCH for doing interesting things in Assembly Language; and maybe even an article or two on interfacing the 4A to the outside world (if I can persuade anyone to write them!!). Whatever happens in TI-LINES will filter down in summary to TI*MES, but if anyone wants to subscribe direct the current cost is a tenner for 14 issues (this first year only). New subscribers get all the current issues to date, so that there are no staggered subscription renewal dates. However, due to some increased costs the subs may have to go up next year. I am trying my best to bring the costs down (TI-LINES has cost me over 350 quid to produce so far) and I am hoping to purchase my very own photocopier in the New Year. As I am also running TI*MES the reduction in copying costs will be very well received!

Finally, in 1985 my other venture, QUINSOFT, will be trying to publish a series of small booklets (A5 like TI*MES, about 20,000 words, for much less than a fiver) so if anyone has anything interesting to say but can not interest the major publishers, well, try QUINSOFT.

Good Programming, and Better Earnings   Peter Brooks

M I C R O - L I N E S

B y   D a v i d   B r o w n

Are you fed up with not having enough memory on your Texas ? Sick of your programs crashing with MEMORY FULL errors ? Here are some hints & tips that will save you some memory and maybe improve your programming.

When you bought your computer, you may remember seeing various adverts saying "16K RAM". This is not strictly true as the computer uses some of the memory for the screen display. In fact you have only 14847 bytes of RAM to use, and if you are using Extended BASIC (XB), you are even worse off because you have only 13928 bytes free.

Because of the small memory and the high price of expansion units, it is essential to use memory-saving techniques. Here are a few:-

Example A

```
1 CALL CHAR(32,"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFF") - uses 94 bytes of memory.
```

Example B

```
1 CALL CHAR(32,"FFFFFFFFFFFFFFFF")::CALL CHAR(33,"FFFFFFFFFFFFFFFF")::
CALL CHAR(34,"FFFFFFFFFFFFFFFF")::CALL CHAR(35,"FFFFFFFFFFFFFFFF")
 - uses 145 bytes of memory.
```

Example C

```
1 CALL CHAR(32,"FFFFFFFFFFFFFFFF")
2 CALL CHAR(33,"FFFFFFFFFFFFFFFF")
3 CALL CHAR(34,"FFFFFFFFFFFFFFFF")
4 CALL CHAR(35,"FFFFFFFFFFFFFFFF") - uses 160 bytes.
```

Example D

```
1 READ A$
2 CALL CHAR(32,A$)
3 DATA FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
 - uses 188 bytes.
```

The examples above show how you can save a lot of memory through the use of different programming techniques. Example D uses twice as much memory as Example A. Of course, the first two examples are designed to run in XB.

Try experimenting with other statements to see which uses the least amount of memory. If you have XB, type in your statements to be tested, complete with line numbers, RUN the program, and BREAK, and when the cursor appears, type SIZE. The computer will then print the number of bytes that you have free. Subtract this number from 13928 and you will be left with the number of bytes that the program uses. That is pretty straightforward, but if you aren't lucky enough to own the XB module, you will have to do a bit of programming. Type in this small program:-

```
1 A = A + 8
2 GOSUB 1
rest of your program
```

RUN, and eventually the program will *crash with a MEMORY FULL error. When the cursor appears, type in the command PRINT A. The number of bytes free will be printed. Subtract this number from 14536 and you will have the number of bytes that your program occupies.

MEMORY SAVER A

If a number is used a lot in a particular program, it may use up less memory if it assigned to a variable. For example, if the number 96 appears a lot in your program (as a character code in an HCHAR statement for instance) it might be easier to type A = 96 and then CALL HCHAR(8,9,A,6). The amount of memory saved depends on how many times the number is used in the program, but if it used more than 6 times a saving will definitely be made.

MEMORY SAVER B

Keep variable names short. A variable uses as many bytes as it is long, so the shorter it is, the less memory it will use.

MEMORY SAVER C

If your program has many lines of identical coding, you need only type them in once and use a GOSUB to get to it many times. This can save a lot of memory.

MEMORY SAVER D

If you have XB, make your lines as long as possible by using multiple-statements. Each line you use requires at least 6 bytes (that's just for a simple REM statement).

That's it for now.

*ED C&A : Your program may crash with
*NAME CONFLICT IN XXX
If this happens then replace the 'A'
= MEM + 8

David Brown

MBX REVIEW

Peter Brooks November 1984

At the Manchester Meeting on November 3rd I bought an MBX Speech Recognition Unit, together with an adaptor (from UK to US mains) and two MB cartridges: "I'm Hiding" and "American Baseball". I had intended to investigate the likelihood of being able to make use of the apparently astounding facilities of the unit as an aid for the vocally handicapped, but so far such a project seems doomed to fall at the first fence.

In the meantime I have had the unit farmed out to two families, with children ranging in ages from 4 to 13 years old. At the same time I provided a few "standard" TI modules - Video Games I, Parsec (not always with speech), and TI Invaders. Joysticks were not provided (and the MBX joystick is incompatible with the TI console).

This is a summary of the experiences of the parents of both families, and it may be of some help to those debating whether to part with the large bundle of folding stuff necessary to buy such a system.

To begin with, a short description of the unit which I obtained through HOWARD GREENBERG of ARCADE HARDWARE, 211 Horton Road, Fallowfield, MANCHESTER M14 7QE, tel: 061 225 2248.

I understand that the price I paid was a 'special' for the Show, so the £125 I paid may not be typical.

For that, I received an MBX console, a fearsome-looking joystick which would have flummoxed Luke Skywalker, a mains adaptor to alter the UK mains cycle to that of the USA, a headset consisting of a collapsible headband with microphone, and two games cartridges (detailed above).

The first disappointment is the headset: it appears to have a standard stereo jack, but closer investigation reveals that the earphones are merely pads disguised to look like earphones. The blurb says that they have been specially designed for comfort, but I'm afraid that doesn't ring true. As far as comfort goes, having two foam pads clamped unnecessarily over your ears makes them definitely uncomfortable after only a short while; it would have been better to have provided just a band which sits over the head without touching the ears. Otherwise it looks as though you have a pair of the latest mini-earphones which come with the fashionable 'personal hi-fi' units. I'm confused about the jack, but as the microphone is also covered in foam I am reluctant to rip it off and see if there are one or two mikes underneath to justify the stereo jack.

The second disappointment is the MBX console - it is not touch sensitive but 'thump' or 'finger-roll' sensitive, very much like the keyboard of the Sinclair ZX80 and '81 (which appears in the States as the TIMEX TS1000 I believe). The games supplied came with overlays to fit onto the MBX keyboard, but these were so difficult to fit without damage that I doubt if they will last for long, and if not fitted correctly will result in delays when trying to get the keyboard to respond when it is pressed.

The connection to the TI console is made through the joystick and cassette ports, and another design fault appears here. The cable for the cassette port fetches up sharply against the mains lead on the PAL 4A (I don't know whether this is the same for the NTSC 4A), and it is obvious that the MBX cable will not take much effort to break because of this position. As the jacks are 'solid state' (using the original meaning of the term) when the cable does eventually break inside, rewiring will be impossible and a new jack will have to be obtained.

I'm not impressed by the ability of some American plugs to be actually bolted into the socket, which while it might prevent the thing from being accidentally pulled out, also means that there is a hazard as far as possible electrocution is concerned. Fortunately, although it does not sit easily in the socket, the adaptor does not permit fixing permanently.

After ten days with one family, and one evening with another, the general concensus of opinion was identical: very impressive as far as speech recognition goes (although the children were not in the least impressed, and in fact became quite frustrated when the machine did not reconise their carefully-enunciated commands) but after only a short time boredom set in, and in fact the modules which stole the show were Video Games I, Parsec (even without speech) and TI Invaders. It is possible to play any of the three TI games on a monochrome TV, which one family had, while I'm Hiding demands a colour set. The American Baseball game also suffered on the monochrome set, but the longest trials were carried out on the colour TV (just in case anyone was going to comment).

After having played with the system myself for a while now, I find that I agree with the findings of the two families. In particular, the game I'm Hiding was far too slow for the younger kiddies (at whom it's aimed) and both families said that the screen generation and the I'm Hiding sequence needed to be sped up considerably.

Interestingly, everyone on whom I have tried the unit has gone through virtually the same procedure: voice-training using silly voices, odd sounds, and wrong words, followed by a spate of coughing, squawking, and playing of rubber lips during the use of the speech recognition. Eventually they settle down to using the unit 'properly', which is when the disenchantment sets in.

The voices provided during I'm Hiding also came in for severe criticism as being irritating (even the kids were taking the mickey out of them at one point) and largely unhelpful. Probably this is due mostly to language differences - anyone who has seen the way that Children's TV (i.e. produced by children for children) ridicules the Mickey Mouse Fan Club and its "'cos we luv you all" approach will understand part of the problem.

I should point out that both of the families are 'computer-naive', although one already owns a 4A (but no modules) and the other has had very limited experience on a minimal-system Commodore Vic 20.

I understand that at least one OTIU member with young children has an MBX system, and I would be interested in any feedback from members with regard to the results presented here.

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

TI WRITER TIPS

For some time I have been dissatisfied with the TEXT FORMATTER in TI-WRITER. The first problem was that it performed a Form-Feed at the start -I have since discovered that this works fine if you start part-way down a page, in which case it just rolls to the start of the next sheet. But most people start printing on a new sheet anyway.

After having spent some time disassembling the formatter program and removing the offending form-feed I obtained a copy of TI-WRITER2 in which this had been done and has, perhaps what is the best modification, -true lower case screen letters...

The next problem was having to save to disk, -load the Formatter program, -reload the script and then print. Because the program does not stay in FORMAT mode it has to be re-loaded before each printing. As a result I tended to use the EDITOR mode for most of my work.

Prior to reading an article in H.C.M magazine (Aug 1984),I used to prime the printer in IMMEDIATE mode -to set print face and emphasized printing. It was possible to do some simple formatting by utilizing the codes listed on p146 of the TI-WRITER manual this was limited to changing to wide type or compressed type but that was about all.

The HCM article showed how to use the "CONTROL 'U' " function to obtain other print instructions . This was a great improvemnt on what I could do previously, I could now set up for emphasized printing (CONTROL 'U' FUNCTION 'R' CONTROL 'U'[to cancel mode] KEY 'E') -this is equivalent to putting CHR$(27);CHR$(69) into print instructions. I could change to Italic print in mid-line -reset the printer and double strike words.

Using this method it was possible to use all the printer instructions which only required two CHR$ codes. I was not able to utilize the three-code instructions illustrated in the HCM. I tried many methods to utilize the underline codes[CHR$(27);CHR$(45);CHR$(1)], but to no avail. It appeared that CHR$(27);CHR$(45) was a default instruction which activated the underline and it ignored the other CHR$ codes used. The problem was finally solved with help from Richard Blanden and Stephen Shaw. On my printer -a STAR DP510(Gemini 10) and parallel port -it is necessary to insert the third code in the control mode.(CONTROL 'U' FUNCTION 'R' CONTROL 'U' SHIFT -[the 'minus' KEY] CONTROL 'U' SHIFT 'A') and to cancel by repeating the process but ending with SHIFT '2'[CHR$(0)].

There is very little problem identifying the codes because they appear in a special tiny form of HEX code -except for CHR$(10) which is screened as LF and CHR$(13) as CR.

It is only necessary to use the CONTROL 'U' function to access CHR$ lower than ASCII 31. All the other characters can be obtained direct from the keyboard. It is possible to obtain the CHR$ above 96 by means of the CONTROL key mode -SHIFT 1 will print a lower-case (a) -SHIFT 'full stop' prints "~"[CHR$(126)].

I have not managed to do a RIGHT justification yet but will continue to try. My next investigation will be to try to find out how the TRANSLITERATE command operates and try to enable this to be used with the EDITOR and so avoid having to bother with the FORMATTER except for special applications. I hope that this help others to enjoy using the TI-WRITER module by just using a single mode of operation.

CONTROL CODES FOR USE WITH TEXT EDITOR.

All codes are accessed by use of CONTROL key + 'U'.

CHR$	KEYS USED	CHR$	KEYS USED
0	SHIFT + 2	16	SHIFT + P
1	A	17	Q
2	B	18	R
3	C	19	S
4	D	20	T
5	E	21	U
6	F	22	V
7	G	23	W
8	H	24	X
9	I	25	Y
10	J	26	Z
11	K	27	FUNCT R
12	L	28	FUNCT Z
13	M	29	FUNCT T
14	N	30	SHIFT 6
15	O	31	FUNCT U

All other CHR$ are as ASCII codes and can be used in the usual way. The above table can be found on p146 TI-WRITER MANUAL and shows the character which is visible on the screen when using these codes.

These codes are useful when using the TEXT EDITOR mode of TI-WRITER. The text can be formatted by sending the appropriate codes for your printer.

For example if you want to overstrike a word or line of text place the codes on the preceeding line or in front of the word:

e.g TEXT EDITOR is achieved on my printer by keying "CONTROL 'U' FUNCTION 'R' CONTROL 'U' and KEY 'G' [CHR$(27);CHR$(71)] . The mode was cancelled by repeating the process but using 'H' in place of 'G'. ('bG TEXT EDITOR'bH is...[as seen on screen])

When using this method it is important to know that the codes do not take any printer space so a space must be left after 'H' otherwise the next word becomes joined to the one which is over struck.

The table above was typed on consecutive lines and the line spacing set to 1/4".(The coding on my printer is CHR$(27);CHR$(65);n -this sets the spacing to n/72").

The key sequence was CONTROL 'U' FUNCTION 'R' CONTROL 'U' KEY 'A' CONTROL 'U' SHIFT 'R' (18/72 =1/4").

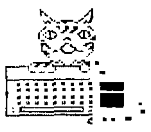
ADBurt

(A. D. Burt)

17 Wagtail Close
Twyford
Reading
RG10 9ED

TIPS FROM TIGERCUB

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus OH 43213
(614)235-3545



Tigercub Software has over 13Ø original programs for the TI-99/4A in Basic and Extended Basic, on casette or disk, at only $3.ØØ each. A descriptive catalog is available for $1.ØØ, which may be deducted from your first order.

I am presently writing Extended Basic versions of many of my programs, enhancing some programs and adding some new ones, and will soon be publishing a new catalog.

No, the Tips are not available by subscription, and I do not have back issues available. However, the entire contents of Tips #1 through Tips #14 are now available on disk, with more added - a full disk of 5Ø programs, routines and files for only $15.ØØ postpaid.

If your local school has purchased TI-99/4A computers for their classrooms, why not let them know that Tigercub has educational programs at a price they can afford?

Disk Menu Loader.

Several people have sent me enhancements to my Menu Loader, and I greatly appreciate them. The trouble is, if I incorporated them all the program would take up about 25 disk sectors' So, I have borrowed some ideas, added a few of my own, and here is the result. It will list and load up to 99 programs, stopping at the end of every screenfull or stopping whenever any key is pressed and then offering you the choice of loading, deleting or quitting. It will ask you to verify a deletion by name before deleting it, and will display the name of the program it is loading. It also contains a feature to warn you if you are getting a bad count of disk sectors used - which I find happening more often than you might realize.

```
1ØØ !by A. Kludge/M. Gordon/
T. Boisseau/J. Peterson/etc.
11Ø CALL CLEAR :: CALL INIT
:: CALL LOAD(8196,63,248)::
CALL LOAD(16376,67,85,82,83,
79,82,48,8)
12Ø CALL LOAD(12288,129,195,
126,165,129,153,1Ø2,6Ø)
13Ø CALL LOAD(12296,2,Ø,3,24
Ø,2,1,48,Ø,2,2,Ø,8,4,32,32,3
6,4,91):: CALL LINK("CURSOR"
)
14Ø CALL CLEAR :: CALL SCREE
N(5):: FOR S=1 TO 14 :: CALL
 COLOR(S,7,16):: NEXT S :: C
ALL VCHAR(1,31,1,96):: CALL
COLOR(Ø,2,16)
15Ø OPTION BASE 1 :: DIM PG$
(99),T$(5)
16Ø T$(1)="dis/fix" :: T$(2)
="dis/var" :: T$(3)="int/fix
" :: T$(4)="int/var" :: T$(5
)="program"
17Ø IMAGE ##
18Ø DISPLAY AT(1,9):"DISKETT
E MENU"
19Ø ! IF YOU HAVE MORE THAN
ONE DISK DRIVE, DELETE THE !
 IN LINE 2ØØ AND THE FIRST S
TATEMENT IN 21Ø
2ØØ ! DISPLAY AT(12,6):"DISK
? (1-3):" :: ACCEPT AT(12,19
)SIZE(-1)VALIDATE("123"):D$
:: D$="DSK"&D$&"."
21Ø D$="DSK1." :: OPEN #1:D$
,INPUT ,RELATIVE,INTERNAL ::
 INPUT #1:N$,A,J,K :: DISPLA
Y AT(1,2):SEG$(D$,1,4)&" - D
iskname= "&N$;
22Ø DISPLAY AT(2,2):"Availab
le=";K;"Used=";J-K:" Prog Fi
lename  Size    Type":"----
---------- ---- -------" ::
I,VT=Ø :: TT=J-K
23Ø FOR X=1 TO 99 :: IF X/2Ø
<>INT(X/2Ø)THEN 26Ø
24Ø DISPLAY AT(24,1):"Type c
hoice or 99 for more" :: ACC
EPT AT(24,27)VALIDATE(DIGIT)
:K :: IF K=99 THEN 25Ø :: IF
 K>Ø AND K<NN+1 THEN 42Ø ELS
E 24Ø
25Ø X=1
26Ø I=I+1 :: IF I>127 THEN K
=X :: GOTO 36Ø
27Ø INPUT #1:P$,A,J,B :: NN=
NN+1
28Ø IF LEN(P$)=Ø THEN 32Ø
29Ø DISPLAY AT(X+4,2):USING
17Ø:NN :: DISPLAY AT(X+4,6):
P$ :: PG$(NN)=P$ :: DISPLAY
AT(X+4,18):USING 17Ø:J :: DI
```

```
SPLAY AT(X+4,22):T$(ABS(A)):
: VT=VT+J
3ØØ CALL KEY(Ø,KK,ST):: IF S
T=Ø THEN 31Ø :: FLAG=1 :: GO
TO 32Ø
31Ø NEXT X
32Ø DISPLAY AT(X+4,1):" " ::
 DISPLAY AT(X+4,2):USING 17Ø
:NN :: DISPLAY AT(X+4,6):"Te
rminate" :: DISPLAY AT(X+5,2
):STR$(NN+1)&" Delete"
33Ø IF VT=TT OR FLAG=1 THEN
35Ø :: DISPLAY AT(2,25)SIZE(
4):VT
34Ø FOR @=1 TO 1Ø :: DISPLAY
 AT(2,25)SIZE(1):CHR$(3Ø)::
DISPLAY AT(2,25)SIZE(1):" "
:: CALL SOUND(-99,11Ø,Ø,-4,Ø
):: NEXT @
35Ø DISPLAY AT(X+6,1):"    C
hoice?" :: ACCEPT AT(X+6,16)
SIZE(2)VALIDATE(DIGIT):K ::
IF K<>NN AND K<>NN+1 THEN 41
Ø
36Ø IF K=NN THEN CALL CLEAR
:: CLOSE #1 :: END
37Ø DISPLAY AT(X+5,11)SIZE(1
8):" #?" :: ACCEPT AT(X+5,15
)SIZE(2)VALIDATE(DIGIT):KD :
: IF KD<1 OR KD>NN THEN 37Ø
38Ø DISPLAY AT(X+6,1)SIZE(27
)BEEP:" Verify - Delete ";PG
$(KD);"?" :: DISPLAY AT(X+6,
28)SIZE(1):"Y" :: ACCEPT AT(
X+6,28)SIZE(-1)VALIDATE("YN"
):Q$ :: IF Q$<>"Y" THEN 4ØØ
39Ø DELETE D$&PG$(KD)
4ØØ CLOSE #1 :: CALL VCHAR(1
,3,32,672):: NN=Ø :: X=Ø ::
GOTO 18Ø
41Ø IF K<1 OR K>99 OR LEN(PG
$(K))=Ø THEN 32Ø
42Ø CLOSE #1
43Ø CALL INIT :: CALL PEEK(-
31952,A,B):: CALL PEEK(A*256
+B-65534,A,B):: C=A*256+B-65
534 :: A$=D$&PG$(K):: CALL L
OAD(C,LEN(A$))
44Ø FOR I=1 TO LEN(A$):: CAL
L LOAD(C+I,ASC(SEG$(A$,I,1))
):: NEXT I :: CALL LOAD(C+I,
Ø)
45Ø CALL VCHAR(1,3,32,672)::
 CALL SCREEN(8):: FOR S=Ø TO
 14 :: CALL COLOR(S,2,1):: N
EXT S :: DISPLAY AT(12,2):"L
OADING ";A$
46Ø RUN "DSKX.123456789Ø"
```

If you don't like my Tigercub cursor, just delete lines 11Ø (after the CALL CLEAR), 12Ø and 13Ø. That routine for redefining the cursor has appeared recently in various newsletters without attribution, and I'd like to know who to credit for it. The secret of it is in line 12Ø, where the numbers after 12288 are the decimal equivelants of the hexadecimal numbers (which are the hex equivelant of the binary numbers represented by the off/on pixels) used to redefine a character.

You may have noticed that all programs published in the Tigercub's Tips are in 28-column format, just the way they will appear on the screen. And they are printed directly from LISTed actual programs, so that they cannot contain typographical errors - don't you wish the computer magazines did that!? The problem is that when a program listing is merged into the TI-Writer buffer and printed in the formatter mode, the @, &, * and the exponent sign are treated as control characters, and strange things happen!

The following program will convert a program, which has been listed to disk with LIST "DSK1.FILENAME", into a file in 28-column format which can be loaded into TI-Writer, and will optionally substitute the left and right braces, ASCII 124 and the tilde for the @, &, * and the exponent sign, and transliterate them so that they will print correctly in the formatter mode. However, for that very reason this program will not print correctly! When you come to line 28Ø, type DATA shift 2, fctn F, shift 7, fctn G, shift 6, fctn W, shift 8, fctn A.

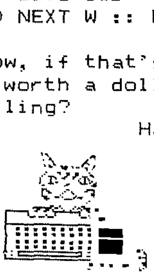
```
1ØØ DISPLAY AT(2,4)ERASE ALL
:"28-COLUMN CONVERTER" :: DI
SPLAY AT(5,12):"by Jim Peter
son"
11Ø DISPLAY AT(7,1):" To con
vert a program, saved":"with
 LIST ""DSK1.FILENAME"",":"i
nto 28-column format which":
"can be merged into the text
"
12Ø DISPLAY AT(11,1):"buffer
 of TI-Writer."
13Ø DISPLAY AT(13,1):" Optio
nally with transliter-":"ate
```

```
d @, &, * and ^ for cor-":"r
ect printing from formatter"
:"mode."
140 DISPLAY AT(18,1):" Do yo
u want to print the":"file f
rom the":" (E)ditor?":" (F)o
rmatter?"
150 ACCEPT AT(23,1)VALIDATE(
"EF")BEEP:Q$
160 DIM A$(1000):: CALL CLEA
R :: INPUT "What is the FILE
NAME?          DSK1.":FN$ :: FN
$="DSK1."&FN$ :: PRINT : :
170 INPUT "what is the new F
ILENAME?   DSK1.":PN$ :: PN$
="DSK1."&PN$ :: OPEN #1:FN$,
DISPLAY ,VARIABLE 80,INPUT :
: OPEN #2:PN$,DISPLAY ,VARIA
BLE 80,OUTPUT
180 IF Q$="E" THEN 190 :: PR
INT #2:".TL 126:94;" :: PRIN
T #2:".TL 123:64;" :: PRINT
#2:".TL 125:38;" :: PRINT #2
:".TL 124:42;"
190 FOR L=1 TO 1000 :: LINPU
T #1:A$(L):: IF LEN(A$(L-1))
=80 OR LEN(A$(L-1))=160 THEN
 A$(L-1)=A$(L-1)&A$(L):: L=L
-1
200 IF EOF(1)THEN L=L+1 :: G
OTO 220
210 NEXT L
220 FOR J=1 TO L-1 :: S=1
230 FOR T=1 TO 10 :: B$(T)=S
EG$(A$(J),S,28):: IF Q$="E"
THEN 240 :: GOSUB 280
240 S=S+28 :: NEXT T
250 FOR N=1 TO 10 :: IF B$(N
)<>"" THEN PRINT #2:B$(N)
260 NEXT N
270 NEXT J :: CLOSE #2 :: CL
OSE #1 :: END
280 DATA @,@,&,&,^,^,*,*
290 RESTORE 280
300 FOR W=1 TO 4 :: READ CH$
,R$
310 X=POS(B$(T),CH$,1):: IF
X=0 THEN 330
320 B$(T)=SEG$(B$(T),1,X-1)&
R$&SEG$(B$(T),X+1,LEN(B$(T))
):: GOTO 310
330 NEXT W :: RETURN
```

Now, if that's what I give away, isn't it worth a dollar to find out what I'm selling?

Happy hackin'

Jim P.

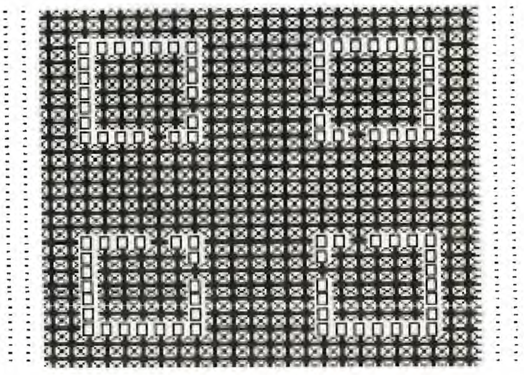


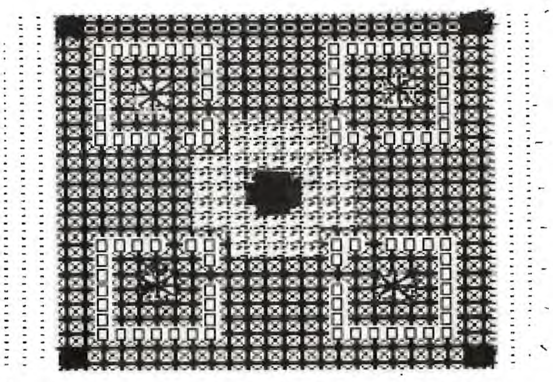
TIGERCUB

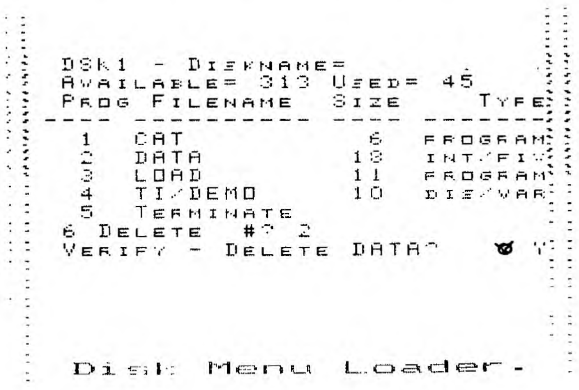
SOFTWARE

```
100 REM  CONTINUOUS MUSIC
     AND FULL COLOUR
     PATTERNS RUNS IN
        BASIC OR EXT BASIC
110 REM ***************
120 CALL CLEAR
130 CALL SCREEN(16)
140 PRINT "  COLUMBIA, THE G
EM OF THE"::TAB(12):"OCEAN":
:::"  programmed by Jim
Peterson"::::::"runs better
 in Ext. Basic"
150 FOR D=1 TO 500
160 NEXT D
170 DIM L$(12),S(12)
180 CH$="995A3CFFFF3C5A99"
190 FOR CH=40 TO 136 STEP 8
200 CALL CHAR(CH,CH$)
210 NEXT CH
220 DATA XXXXX(h`xHPPHx`h(XX
XXX,XPPPP(hh`xHHx`hh(PPPPX,X
P0000000`xx`000000
0PX,XP0@@(HH0p`xx`p0HH(@@0PX
230 DATA XP0@@(HH0pX``Xp0HH(
@@0PX,((0(((HH0pp``pp0HH(((0
((,hh0HHH8888888888888HHH
0hh,`h0HHH8@0XH((HX0@8HHH0h`
240 DATA x`0000800H((H008000
0`x,x`ppp8XH(PP(HX8ppp`x,Hx`
Xp8H(PHHP(H8pX`
xH,PHx``8(PHXXHP(8``xHP
250 FOR SET=2 TO 13
260 CALL COLOR(SET,1,1)
270 NEXT SET
280 FOR J=1 TO 12
290 READ L$(J)
300 NEXT J
310 FOR J=1 TO 12
320 PRINT TAB(3);L$(J)
330 NEXT J
340 FOR J=12 TO 2 STEP -1
350 PRINT TAB(3);L$(J)
360 NEXT J
370 PRINT TAB(3);L$(1);
380 CALL VCHAR(1,29,1,192)
390 DATA 277,294,330,370,392
,440,494,523,554,587,659,400
00
400 DATA 3,2,0,1,2,7,4,5,0,3
,5,7,1,5,0,4,6,0,3,10,0,1,8,
0,2,7,0,10,5,0
410 DATA 2,2,0,4,3,0,2,11,0,
2,10,0,2,8,0,2,7,0,2,6,0,2,5
,7,8,5,0,6,4,0
420 DATA 2,6,7,4,6,0,2,6,7,2
,6,0,4,6,7,2,7,0,2,9,0,2,10,
0,10,6,0
430 DATA 4,10,0,2,9,0,2,7,0,
2,6,0,2,5,0,2,4,0,2,3,0,2,6,
0,2,1,0,14,2,0
440 DATA 2,2,0,4,6,0,3,6,7,1
,6,0,2,5,0,2,4,0,2,3,0,2,2,0
,2,2,7,10,5,0
```

```
450 DATA 2,5,0,2,6,0,4,7,0,2
,7,7,2,8,0,2,7,0,2,6,0,2,5,0
,12,6,0
460 DATA 4,6,0,2,10,7,4,10,0
,2,10,7,2,8,0,2,7,0,2,6,0,2,
5,0,2,4,0,10,3,0
470 DATA 2,11,0,2,10,0,2,8,0
,2,7,0,2,6,0,2,5,0,4,4,0,2,3
,0,2,4,0,12,5,0
480 DATA 2,4,0,2,5,0,4,6,0,3
,6,7,1,6,0,4,6,7,3,10,0,1,8,
0,12,7,0
490 DATA 2,4,0,2,5,0,4,6,0,3
,6,7,1,6,0,4,6,7,3,10,0,1,8,
0,12,7,0
500 DATA 2,5,0,2,7,0,2,10,0,
4,10,0,2,8,0,2,7,0,2,6,0,2,5
,0,2,4,0,2,4,7,10,3,0
510 DATA 2,11,0,2,10,0,2,8,0
,2,7,0,2,6,0,2,5,0,4,4,0,2,3
,0,2,4,0,8,5,0
520 FOR N=1 TO 12
530 READ S(N)
540 NEXT N
550 CX=1
560 NX=1
570 FOR J=1 TO 117
580 READ T,N,V
590 FOR M=1 TO T
600 CALL SOUND(-999,S(N),V,S
(N)+5,V)
610 CALL COLOR(N+NX,N+2,CX)
620 CALL COLOR(N,CX,CX)
630 NEXT M
640 NEXT J
650 RESTORE 400
660 XX=XX+1
670 ON XX GOTO 680,700,720,5
70,750,770,790,860
680 CX=2
690 GOTO 570
700 CALL SCREEN(2)
710 GOTO 570
720 CX=16
730 CALL SCREEN(14)
740 GOTO 570
750 CALL SCREEN(5)
760 GOTO 570
770 CALL SCREEN(7)
780 GOTO 570
790 FOR SET=1 TO 13
800 CALL COLOR(SET,2,2)
810 NEXT SET
820 CALL SCREEN(16)
830 CX=2
840 NX=0
850 GOTO 570
860 CALL SCREEN(2)
870 XX=XY-2
880 GOTO 570
```







```
DSK1 - DISKNAME=
AVAILABLE= 313 USED= 45
PROG FILENAME   SIZE      TYPE
---- ---------- ---- -------
  1  CAT           6  PROGRAM
  2  DATA         18  INT/FIX
  3  LOAD         11  PROGRAM
  4  TI/DEMO      10  DIS/VAR
  5  TERMINATE
6 DELETE  #? 2
VERIFY - DELETE DATA?    Y

 Disk Menu Loader-
```

We welcome our new columnist to the Sydney NEWSDIGEST. This will be a continueing feature set of articles, to ensure that you get the best out of your Extended Basic. Tony is a member of the Newcastle group, and we understand he has been a great source of inspiration to our friends in that group. So, here he is, from the FUNNELWEB FARM (049) 52 3162

In this series of notes on TI Extended Basic for the TI-99/4a we will concentrate on those features which have not received due attention in User-group newsletters or commercial magazines. In fact most of the programs published in these sources make little use of that most powerful feature of XB, the user defined sub-program, or of some other features of XB. Worse still is to find commercially available game programs which are object lessons in how to write tangled and obscure code. The trigger for this set of tutorial notes was a totally erroneous comment in the TI.S.H.U.G Newsdigest in Jun 1983. Some of the books I have seen on TI Basic don't even treat that simpler language correctly, and I don't know of any systematic attempts to treat XB. The best helper is TI's Extended Basic Tutorial tape or disk. The programs in this collection are unprotected and so open for inspection and it's worth looking at their listings to see an example of how sub-programs can give an easily understood overall structure to a program.

Well, what are we going to talk about then ? Intentions at the moment are to look at

(1) User-defined sub-programs
(2) Prescan switch commands
(3) Coding for faster running
(4) Bugs in Extended Basic
(5) Crunching program length
(6) XB and the Peripheral Box
(7) Linking in Assembler routines

Initially the discussion will be restricted to things which can be done with the console and XB only. Actually, for most game programming the presence of the memory expansion doesn't speed up XB all that much as speed still seems to be limited by the built-in sub-programs (CALL COINC, etc) which are executed from GROM through the GPL interpreter. The real virtue of the expansion system for game programming, apart from allowing longer programs, is that GPL can be shoved aside for machine code routines in the speed critical parts of the game, which are usually only a very small part of the code for a game. Even so careful attention to XB programming can often provide the necessary speed. As an example, the speed of the puck in TEX-BOUNCE is a factor of 10 faster in the finally released version than it was in the first pass at coding the game.

Other topics will depend mainly on suggestions from the people following this tutorial series. Otherwise it will be whatever catches our fancy here at Funnelweb Farm.

(1)

II. SUB-PROGRAMS in OVERVIEW

Every dialect of Basic, TI Extended Basic being no exception, allows the use of subroutines. Each of these is a section of code with the end marked by a RETURN statement, which is entered at some earlier point by a GOSUB statement somewhere else in the program. When RETURN is reached control passes back to the statement following the GOSUB. As an example

```
290 ....
300 GOSUB 2000
310 .....
.
.
2000 CALL KEY(Q,X,Y):: IF Y=1
THEN RETURN ELSE 2000
```

This simple example waits for and returns the ASCII code for a fresh keystroke, and might be called from a number places in the program. Very useful, but there are problems. If the line number of the subroutine is changed, other than by RESequencing of the whole program (and many dialects of Basic for microcomputers aren't even that helpful) then the GOSUBs will go astray. Another trouble, which you usually find when you resume work on a program after a lapse of time, is that the statement GOSUB 2000 doesn't carry the slightest clue as to what is at 2000 unless you go and look there or use REM statements. Even more confusingly the 2000 will usually change on RESequencing, hiding even that aid to memory. There is an even more subtle problem -- you don't really care what the variable "Y" in the subroutine was called as it was only a passing detail in the subroutine. However, if "Y" is used as a variable anywhere else in the program its value will be affected. The internal workings of the subroutine are not separated from the rest of the program, but XB does provide four ways of isolating parts of a program.

(1) Built-in sub-programs
(2) DEF of functions
(3) CALL LINK to machine code routines
(4) User defined BASIC sub-programs

The first of these, built-in sub-programs, are already well known from console Basic. The important thing is that they have recognizable names in CALL statements, and that information passes to and from the sub-programs through a well defined list of parameters and return variables. No obscure Peeks and Pokes are needed. The price paid for the power and expressiveness of TI Basic and XB is the slowness of the GROM/GPL implementation.

DEF function is a primitive form of user defined sub-program found in almost all BASICs. Often its use is restricted to a special set of variable names, FNA,FNB,... but TI Basic allows complete freedom in naming DEFed functions (as long as they don't clash with variable names). The "dummy" variable "X" is used as in a mathematical function, not as an array index

```
100 DEF CUBE(X)=X*X*X
```

(2)

doesn't clash with or affect a variable of the same name "X" elsewhere in the program. "CUBE" can't then be a variable whose value is assigned by a LET (or =), but "X" may be. Though DEF does help program clarity it executes very slowly in TI Basic, and more slowly than user defined sub-program CALLs in XB.

CALL LINK to machine code routines goes under various names in other dialects of Basic if it is provided (eg USR( ) in some). It is only available in XB when the memory expansion is attached, as the TI-99/4a console has only [illegible] bytes of CPU RAM for the [illegible]00 lurking in there. We will take up this topic later.

You should have your TI Extended Basic Manual handy and look through the section on SUB-programs. The discussion given is essentially correct but far too brief, and leaves too many things unsaid. From experiment and experience I have found that things work just the way one would reasonably expect them to do (this is not always so in other parts of XB). The main thing is to get into the right frame of mind for your expectations. This process is helped by figuring out, in general terms at least, just how the computer does what it does. Unfortunately most TI-99/4a manuals avoid explanations in depth presumably in the spirit of "Home Computing". TI's approach can fall short of the mark, so we are now going to try to do what TI chickened out of.

The user defined sub-program feature of XB allows you to write your own sub-programs in Basic which may be CALLed up from the main program by name in the same way that the built-in ones are. Unlike the routines accessed by GOSUBs the internal workings of a sub-program do not affect the main program except as allowed by the parameter list attached to the sub-program CALL. Unlike the built-in sub-programs which pass information in only one direction, either in or out for each parameter in the list, a user sub-program may use any one variable in the list to pass information in either direction. These sub-programs provide the programming concept known as "procedures" in other computer languages, for instance Pascal, Logo, Fortran. The lack of proper "procedures" has always been one of the major limitations of BASIC as a computer language. TI XB is one of the BASICs that does provide this facility. Not all BASICs, even those of very recent vintage are so civilised. For example the magazine Australian Personal Computer in its last issue (Mar 84) carried a review of the IBM PCjr computer just released in the US of A. The Cartridge Basic for this machine apparently does not support procedures. Perhaps IBM don't really want or expect anyone to program their own machine seriously in Basic. You will find that with true sub-programs available, that you can't even conceive any more of how one could bear writing substantial programs without them (even within the 14 Kbyte limit of the unexpanded TI-99/4a let alone on a machine with more memory).

The details of how procedures or sub-programs work vary from one language to another. The common feature is that the variables within a procedure are localised within that procedure. How they communicate with the rest of the program, and what happens to them when the sub-program has run its course varies from language to language. XB goes its own well defined way, but is not at all flexible in how it does it.

(3)

Now let's look at how Extended Basic handles sub-programs. The RUNning of any XB program goes in two steps. The first is the prescan, that interval of time after you type RUN and press ENTER, and before anything happens. During this time the XB interpreter scans through the program, checking a few things for correctness that it couldn't possibly check as the lines were entered one by such as there being a NEXT for each FOR. The TI BASICs do only the most rudimentary syntax checking as each line is entered, and leave detailed checking until each line is executed. This is not the best way to do things but we are stuck with it and it does have one use. At the same time XB extracts the names of all variables, sets aside space for them, and sets up the procedure by which it associates variable names with storage locations during the running of a program. Just how XB does this is not immediately clear, but it must involve a search through the variable names every time one is encountered, and appears to trade off speed for economy of storage.

XB also builds a table of the built-in sub-programs that are called. How can it tell the difference between a sub-program name and a variable name? That's easy since sub-program names are always preceded by CALL. This is why sub-program names are not reserved words and can also be used as variable names. This process means that the very slow search through the GROM name tables is only done at prescan, and Basic then has its own list for each program of where to go in GROM for the GPL routine without having to conduct the GROM search every time it encounters a sub-program name while executing a program. In Command Mode the computer has no way provided to find user defined sub- program names in an XB program in memory even in BREAK status. XB also establishes the process for looking up the DATA and IMAGE statements in the program.

Well then, what does XB do with user sub-programs? First of all XB locates the sub-program names that aren't built into the language. It can do this by finding each the name after a CALL or SUB statement, and then looking it up in the internal GROM files of built-in sub-program names. You can run a quick check on this process by entering the one line program

```
100 CALL NOTHING
```

TI Basic will go out of its tiny 26K brain and halt the prescan with a BAD NAME IN 100 error message, while XB, being somewhat smarter, will complete the prescan but halt execution with a SUBPROGRAM NOT FOUND IN 100 message.

The next thing that XB has to do in its prescan is to locate the sub-programs and take care of them. The XB manual insists that the sub-programs come at the end, with nothing but sub-programs after the first SUB statement (apart from REMarks which are ignored anyway). XB then scans and establishes new variable storage areas, starting with the variable names in the SUB xxx(parameter list), for each sub-program from SUB to SUBEND, as if it were a separate program. It seems that XB keeps only a single master list for built-in and user sub-programs. Data statements are also thrown into the common data pool. Try the following little program to convince yourself.

```
100 DATA 1
110 READ X :: PRINT X :: READ
X :: PRINT X
120 SUB NOTHING
130 DATA 2
140 SUBEND
```

When you RUN this program it makes no difference that the second data item is apparently located in a sub-program. IMAGEs behave likewise. On the other hand DEFed functions, if you care to use them, are strictly confined to the particular part of the program in which they are defined, be it main or sub. During the pre-scan DEFed names are kept within the allocation process separately for each subprogram or the main program. Once again try a little programming experiment to illustrate the point.

```
100 DEF X=1 :: PRINT X;Y ::
CALL SP(Y) :: PRINT X;Y
110 SUB SP(Z) :: DEF X=2 ::
Z=X :: DEF Y=3
120 SUBEND
```

This point is not explicitly made in the XB manual and has been the subject of misleading or incorrect comment in magazines and newsletters. A little reflection on how XB handles the details will usually clear up difficulties.

TI BASICs assign nominal values to all variables mentioned in the program as part of the prescan, zero for numeric and null for strings, unlike some languages (some Basics even) which will issue an error message if an unassigned variable is presumed upon. This means that XB can't work like TI LOGO which has a rule that if it finds an undefined variable within a procedure it checks the chain of CALLing procedures until it finds a value. However, unlike Pascal which erases all the information left within a procedure when it is finished with it, XB retains from CALL to CALL the values of variables entirely contained in the sub-program. The values of variables transferred into the sub-program through the SUB parameter list will of course take on their newly passed values each time the sub-program is CALLed. A little program will show the difference.

```
100 FOR I=1 TO 9 :: CALL
SBPR(0):: NEXT I
110 SUB [illegible]):: A=A+1 ::
B=B+1 :: [illegible] A;B
120 SUBEND
```

The first variable printed is reset to 0 each time SBPR is called, while the second, B, is incremented from its previous value each time. Array variables are stored as a whole in one place in a program, within the main program or sub-program in which the DIMension statement for the array occurs. XB doesn't tolerate attempts to re-dimension arrays, so information on arrays can only be passed down the chain of sub-programs in one direction. Any attempt by a XB sub-program to CALL itself, either directly or indirectly from any sub-program CALLed from the first, no matter how many times removed, will result in an error. Recursive procedures, an essential part of TI LOGO, are NOT possible with XB sub-programs , since CALLing a sub-program does not set up a new private library of values.

All of this discussion of the behaviour of TI Extended Basic comes from programming experience with Version 110 of XB on a TI-99/4a with 1981 title screen. Earlier Versions and consoles are not common in Australia, but TI generally seems to take a lot of trouble to keep new versions of programs compatible with the old. On the other hand TI has also been very reticent about the details of how XB works. The Editor/Assembler manual has very little to say about it, less by far even than it tells about console Basic. I am not presently aware of any discussion of the syntax of the Graphics Programming Language (GPL), let alone of the source code for the GPL interpreter which resides in the console ROM of every 99/4a.

Another simple programming experiment will demonstrate what we mean by saying that XB sets up a separate Basic program for each sub-program. RUN the following

```
100 X=1 :: CALL SBPR :: [illegible]
110 SUB SBPR :: X=2 :: [illegible]
:: SUBEND
```

When the program BREAKs examine the value of variable X by entering the command PRINT X, and then CONtinue to the next program BREAK, which this time will be in the main program, where you can once again examine variable values.

We will now summarize the properties of XB sub-programs as procedures in complete XB programs, leaving the details of joining up the various procedures to the next section.

(a) XB treats each sub-program as a separate program, building a distinct table of named (REFed) and DEFed variables for each.

(b) All DATA statements are treated as being in a common pool equally accessible from all sub-programs or the main program as are also IMAGE s[illegible]nts, CHARacters, S[illegible], COLORs, and File spe[illegible]cations.

(c) All other information is passed from the CALLing main or sub- program by the parameter lists in CALL and SUB statements. XB does not provide for declaration of common variables available on a global basis to all sub-programs as can be done in some languages.

(d) Variable values confined within a sub-program are static, and preserved for the next time the sub-program is CALLed. Some languages such as Pascal delete all traces of a procedure after it has been used.

(e) XB sub-programs may not CALL themselves directly or indirectly in a closed chain. Subject to this restriction a sub-program may be CALLed from any other sub-program.

(f) The MERGE command available in XB with a disk system (32K memory expansion optional) allows a library of XB sub-programs to be stored on disk and incorporated as needed in other programs.

Next issue of TI*MES

TONY CONTINUES HIS TUTORIAL ON
EXTENDED BASIC [illegible]
WITH SUBPROGRAM
PARAMETER L[illegible] etc.
BE SURE NOT TO MISS EACH AND EVERY ONE OF THESE LESSONS. Ed

LETTERS

Dear Editor,
Here are details of how to connect a cheap printer to the TI 99/4A RS232 Interface. The printer is the Tandy GCP115.

The two plugs can be easily obtained from Tandy Stores. They are a 25 pin D Plug and a 4 Pin Din Plug. The Pin numbers are as follows:-

Pin 1 on 25 Pin to Pin 4 on Din
Pin 2 on 25 Pin to Pin 1 on Din
Pin 7 on 25 Pin to Pin 3 on Din
Pin 20 on 25 Pin to Pin 2 on Din

[illegible] entry code for files R[illegible]..BA=600.DA=7.PA=N IN QUOTES.

This may be a simple way for T.I. users to have print outs and simple forms of word processors.

Your Faithfully,
D. Atkinson,
Burley In Wharfedale.

Thanks to TISHUG,
PO Box 149,
Pennant Hills,
N.S.W.Australia
2120.

RAMBLES

by Stephen Shaw

Greetings and Happy New Year !

As forewarned in the last RAMBLES, this edition will be a trifle shorter and less technical, due to shortage of time. (On the evening of Dec 19th I had a letter from Clive asking for copy for the 20th... I'm glad to say the copy date has been 'slipped' for me! Sorry for the delay to your TI*MES gentle reader!).

After making a loss in the quarter equivalent to turnover, and a profit in November of two pounds(!), December has become a little busier for Stainless Software, and added to the usual business of work and family, time is in short supply! Stainless Software has seen quite a large drain on resources in the last year, and will not survive in its present form past this Summer. I am looking for ways to cut costs, inevitably this will mean dropping the magazine advertising: leaving it to personal recommendation to spread the word!

Now... I suspect everyone is mentioning THE SHOW in this issue....
On a Saturday in November, some 88% of members attended, with a total attendance of some 1,500. Mr Brooks was taking new memberships. People continued to come in all day, some travelling long distances. Star attraction was Don Bynum (of Parsec fame) who brought along his TI99/8..... confused?
This was November 10th, and the place was CHICAGO!

The UK SHOW I suspect attracted 1000 or so folk, many of them not members of TI*MES. I was surprised at the rate of membership drop out Clive has indicated : do you know any TI owners who are not members? Have you spoken to them?
Of course a number of owners are either upgrading or losing interest in computing : I was certainly not expecting to receive so many letters this year along the lines of 'I am getting a TI for Christmas....'.

Many thanks to Howard Greenburg who made it possible for Stainless Software to attend by sharing his stand (sorry for any confusion folks!) and thereby gained an extra pair of hands to sell his modules! Many thanks to to Edwin P Lees, a Nothern chain of electrical shops, who made possible the video demo of my programs, with service above and beyond the usual! Unfortunately we did not have time to produce a colour picture from the TI, which interfaces very badly to video recorders...!

I certainly look forward to further shows, but in a private capacity only... despite the crush, Stainless made a loss on the day.

I now have a huge mound of paper (notes, magazines, print outs, letters....) to work my way through. Sorry, it will be a bit scrappier than usual...

COMING SOON: in the next issue, after I've had time to tidy up my programs a bit more: SECTOR DISK ACCESS IN BASIC (with Minimem)
USING VDP REGISTERS IN BASIC with minimem

WELCOME to the second / third / first (third in planned delivery, second in actual delivery, first edited in the UK) issue of the PARCO magazine. A very nice presentation. Note that TI charge from £16 to £32 for a repair.

If you thought the article on Sound was similar to the article on Sound in the last TI*MES, so it was! Both articles appeared with the knowledge and consent of the originators... and there were differences of presentation!

Some questions appeared in the Parco magazine awaiting answers... well, as the questions involved ARE common, here follow some of my answers...

MODULE PROBLEMS (Especially Extended Basic):
Every product has its weak point, the TI's is the module port, and the modules.
EXTENDED BASIC is the LARGEST module made, and seems to be the first one you will have trouble with (leading to Parco getting a lot of perfectly good modules back as "faulty").
The problem will show itself in various forms:
Perhaps a st range error message or hang-up in a running program,
perhaps the module name will not appear on the menu...
perhaps you are treated to a pyrotechnic display on the screen!

The mechanism: Inside the flap is a fairly standard socket, mounted on a PCB which in turn in mounted in a socket on the main PCB.
The socket the module slots into has silver plated contacts. Before the module hits these contacts, its board passes through a foam strip soaked in oil of some sort.
This soggy foam is intended to clean the module board and by placing a film over the silver contacts, reduce corrosion.
Inevitably, the oil will dry out, and oily PCBs will attract dirt, and the foam will become very dirty and dirty the board instead of cleaning it.... and we have quite a problem! In extreme use the actual socket contacts can be subject to wear too.

Solutions:
a. Replace the silver plated socket with a phosphor bronzed job. The socket is quite standard, and you are not soldering to the main PCB. Still, perhaps best left for the more nimble solderers!
b. REMOVE THE FOAM STRIP! It just clips on (& off) the socket.
c. Buy a Navarone cartridge expander (Widget)... not cheap, but it brings out to the front the weak area and prevents the console socket becoming worn. The Widget uses phosphor bronzed contacts and shouls last a very long time.
(Contact Howard for details!)
d. Regularly remove contamination and tarnish from your module contacts by gently wiping the boards with a cotton bud. Do not touch the board with your oily fingers! If badly contaminated, you may use pure alcohol on the bud (or a tape head cleaning fluid). On my ExBas, I find a weekly wipe to be needed!

Also, console lock ups can be caused by STATIC. I have problems with this, causing not only my computer, but also my radio and scales to go wild... my solution may be extreme but if you have problems, try it!
i. Remove all clothing made from artificial fibres, especially acrylics.
(It is fair to warn your girlfriend BEFORE she comes to see your computer..)
ii. Install a humidifier, especially if you have central heating.
Dry air can carry quite a charge.
iii. Try an earthed strip near the front of the keyboard and earth yourself before use!
iv. An ion generator MAY help.

It is not necessary to even touch a computer. Someone with a good charge on them can cause your console to lock up by walking a couple of feet away!

HOW CAN I KNOW WHICH MOODULES WILL RUN ON MY CONSOLE BEFORE I BUY THEM?
Most modules will run on every console.
However, if your console title screen says Vn2.2, you will ONLY be able to run module produced by T.I.... the Atarisoft modules will not be recognised by the console.
Some of the older 99/4As will not properly use some of the newer games modules... the last TI modules and the first Atari modules. This is due to the module programmers taking short cuts, and not allowing for constant rewrites of the console operating system. The most severely affected module is PICNIC PARANOIA. The problem is in the use of the Large Character Set: on old consoles, these modules produce what looks like Japanese writing! The only way to tell is to try it: the consoles cannot be previously identified. Apart from Picnic Paranoia you should still be able to play the games.

MUSICAL COPYRIGHT:
Music is copyright in the same manner as books and computer programs. The copyright extends to the mechanical storage of a tune, for instance on a record, a tape, or a computer program.
Musical copyright is good for fifty years from the death of the composer: most music you know is probably copyright, including even many classical works, such as Peer Gynt.
There is a Society in the UK which 'polices' abuses of musical copyright, the MCPS, and they are now taking action in respect of music used in computer programs. BE WARNED and do not use copyright music in your programs without consent.
To find out if your music IS copyright, or to apply for a licence to use it, write to:
Video/Computer Licencing, MCPS Ltd., Elgar House,
41, Streatham High Road, LONDON SW16 1ER
telephone number is 01 769 4400.
Incidentally..... a computer program containing original music may have stronger legal protection than otherwise, as action can be taken in respect of the unauthorised copying of the music as well....

TERMINAL EMULATOR 2...
and SPEECH...
Two little programs for your amusement:

```
100 OPEN #1:"ALPHON",INTERNAL
110 FOR I=250 TO 254
120 PRINT #1:CHR$(I)
130 NEXT I
140 GOTO 110
150 END
```

Before you run this little program, have the TE2 list the program by using the command: LIST "SPEECH".
Now RUN the program....

```
100 OPEN #1:"ALPHON",INTERNAL
110 FOR I=73 TO 84
120 PRINT #1:CHR$(I)
130 NEXT I
140 GOTO 110
150 END
```

Keep this one running for an hour or so......

Thanks to Pete Brooks for pointing me in these nutty directions...

TO BUY: EX BAS OR MINI MEM:
Mini Memory opens up the whole system to you, but using it to produce Machine Code programs is NOT easy. The articles in TI*MES on what you can do with mini mem in TI Basic may be of help! A point to consider: as far as we know, nobody is picking up production of MiniMem, although we hear that two people are manufacturing ExBas. Mini Mem may not be available much longer!
EX BAS is essential to the Basic programmer. It is a VERY powerful Basic, and some very nice programs are available. EX BAS can take advantage of the 32k ram for a BASIC program (disk drive also required for best use of 32k ram).
Where is your interest? In digging and hacking? Buy minimem. In powerful programs, good games? Buy ExBas.
Note that there are few commercial programs available for mini mem, but many programs are available for ExBas.

SAVING DATA:
A big subject and will be held over to next issue. I will try to cover all aspects of data saving. If things are urgent, take a look at Chapter 5 of my book (Chapter=File Processing, Book=GETTING STARTED WITH THE TI99/4A, cost=5.95) for details, and also take a look at the SAMS package TRIVIA DATA BASE (more on that later!). The section in my book occupies seven pages...

LOGO...
Am I the only one with Logo? Noone has written to me about Logo after my request in the last issue.
SCIENTIFIC AMERICAN recently had a computer issue, and LOGO was discussed. It is a pity their sample program was bugged...
The illustration is of a very sharp nine pointed star.
The procedure is:

```
 TO STAR
 PENDOWN
 REPEAT  9[FORWARD  80 RIGHT 720/9]
 PENUP
```

What sort of 9 pointed star will this produce? Will it be very spiky? How can we draw a very spiky 9 pointed star? (Hint: the angle is 160!)
What conceptual error did the programmer make?

A tip from Ray Elliot:
If you use PRINT lines to set up a screen display,it can be handy to type it onto the screen EXACTLY as it will appear on screen:
TYPE IN:

```
  >100 PRINT                    "
   THIS IS THE FIRST LINE
   THIS IS THE SECOND LINE"
```

...spaces after PRINT, the " at the very end of the first screen line.
The console will close up the spaces when you press enter, but in the meantiee you have been able to see EXACTLY how your print line will appear on the screen!

Two FORTH programs to hand, for next issue, as I need first to ensure we are all setting it up the same way!

JOHN SEAGER wrote to tell me of a problem he had with printing out a list of characters:

```
  FOR T=33 TO 40
  PRINT USING "## "&CHR$(T):T
  NEXT T
```

Try it. Notice what happens with Character 35?

Now try:

```
  FOR T=33 to 40
  PRINT USING "## "&CHR$(T):T;T
  NEXT T
```

Explanation:
(Print Using is available only in Extended Basic by the way!)
The HASH MARK (#) is a control character in the USING statement.
In the second line, a third hash is added when the loop value is 35, resulting in a line looking like:

```
  PRINT USING "## #":35   and
  PRINT USING "## #":35;35
```

In the first case, the 35 is printed, but not chr$(35), the hash, as this has been taken over by the USING command.
In the second case, the first 35 is printed, bit the second number 35 tries to fit into a single digit (the 3rd hash), fails, and thus the 'error' asterisk is printed.

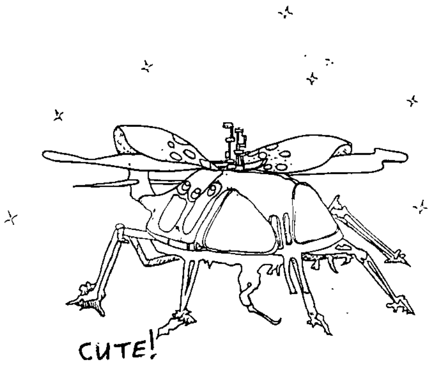
Ian Kilgour reminds me that SOME TV sets do not allow you to properly tune the sound AND picture with the 99/4A.
 This seems to be especially bad with Japanese sets (eg Hitachi), although I do have a black and white Phillips with the same problem.
 When Channel 4 were setting up, the same problem as found on their broadcasts one day, on the Phillips!
 The problem seems to be that some TV's require a signal to be closer to the UK Standard than the Standard actually requires! The problem is one of bandwidth between the sound and picture channels. If you have a multi standard set, switching it to GERMAN PAL will cure the problem. Otherwise you will need a new, less fussy television.

SINGING COMPUTER:
 As seen in Manchester and Wembley (making Fullers talking Spectrum hang its head in shame...)... the program has been submitted to the Club library! and is a TE2 demo requiring Speech Synth plus of course the TE2 module. In theory it should be possible to make the computer sing over a background three part harmony. Who is going to write that one?

INTERNATIONAL USER GROUP, BETHANY
  Clearly things are not going too well over there, and at least one issue of ENHUSIAST 99 <u>HAS</u> been cancelled. I understand an issue is on the way, and that subs may have been reduced. No hard news though yet.

SPRITE ANIMATION:
 Firstly, for all you fans out there with MINI MEMORY, some TI BASIC sprites!

```
100 A$="1030327FF82"
110 B$="000032FF7070301"
120 FOR T=96 TO 120
130 CALL CHAR(T,A$)
140 NEXT T
150 CALL CLEAR
160 CALL SCREEN(5)
170 CALL POKEV(768,38,154,192,15,91,122,193,15,140,84,194,15,208)
180 CALL POKEV(1920,0,20,0,20,0,20,0,20,0,20,0,20)
190 CALL LOAD(-31878,3)
200 FOR T=96 TO 98
210 CALL CHAR(T,B$)
220 NEXT T
230 FOR T=96 TO 98
240 CALL CHAR(T,A$)
250 NEXT T
260 GOTO 200
270 END
```



 Hows that!

 Now in boring old Extended Basic:

```
100 A$="1030327FF82" :: B$="000032FF7070301"
110 CALL CLEAR :: CALL SCREEN(5)
120 CALL CHAR(96,A$,97,B$)
130 CALL SPRITE(#1,96,1,38,154,0,5)
140 CALL SPRITE(#2,96,1,91,122,0,5)
150 CALL SPRITE(#3,96,1,140,84,0,5)
160 CALL COLOUR(#1,16,#2,16,#3,16)
170 CALL PATTERN(#1,97,#2,97,#3,97)
180 CALL PATERRN(#1,96,#2,96,#3,96)
190 GOTO 170
```

 If you find them flapping too quickly' add in:
```
   172 FOR T=1 TO 50 :: NEXT T
   182 FOR T=1 TO 40 :: NEXT T
```

BASICODE:
A few TI owners have expressed an interest in a BASICODE for the TI, for use with BBC's Chip Shop transmissions: first the bad news: the BASICODE translator HAS to be in machine code. If you only have a console you cannot run machine code! I doubt if the coding required would fit into the MiniMem, but as I understand ut, the BASICODE is read into the computer as an ASCII stream... that is to say, you would also need the BOX and the RS232.... and the programs transmitted do not justify that sort of expense, let alone the cost of developing the program! We have many more high quality programs available for the TI already.
BASICODE is a means of transmitting bog standard programs to a lot of people: only the economics of VERY LARBE SCALE production make the cost reasonable.

In PAUL DICKS column in TIHCUC Autumn, he mentions (column 2) CALL FILES(1)... CALL FILES is only available when a disk controller is connected, and is used to reserve VDP RAM for the disk system to use. In doing so, it removes some of the VDP ram available for your program!

Please keep the LETTERS flowing in... but if a reply is needed by post, enclose an S.A.E.... overseas readers, send an International Reply Coupon (seamail) or 4x IRC for airmail outside Europe.
STEPHEN SHAW
10 Alstone Road   STOCKPORT   Cheshire   ENGLAND   SK4 5AH
(The post code is ESSENTIAL)

Gossip, discoveries, queries, ANYTHING, all most welcome!

SAMS PUBLICATIONS:
Available from TI*MES. UK Agents are PITMANS, which speaks well for the quality. I have had time for only a cursory look at two packages so far, but am most impressed. I hope to review more in detail in next issue (provided they haven't sold out!):
TI99/4A BASIC PROGRAMS. Knight & LaBatt. Cassette plus 120pp book.
30 TI Basic programs plus explanations plus suggestions for amendments.
aim: to permit users to see how the programs work so that they can achieve greater knowledge about the TI99/4A and programming
aim: to suggest certain modifications to the programs that users can try on their own. These modifications provide opportunities for further learning since modifying an existing program is excellent experience in program development.
TARGET ACHIEVED! There is also a small section on program development and debugging (the most important part of computing!).
The programs include graphics and music demos, some simple games, and even two "computer tutor" programs! With 30 programs in there, the analysis of each is quite short, and the total novice will need to have his wits about him to sort out what is going on. The programs are (within the limitations of size) well written and I suspect many owners would benefit by studying the listings. The suggested modifications will lead you into quite advanced areas of programming. This package provides raw material and basic guidance: the work is for you to do! Nice package.

TRIVIA DATA BASE. Book & tape. NEEDS EXTENDED BASIC.
Don't let the name put you off. It refers to a tv quiz game. The package deals with the development of a data base: one program creates the data base, the other program is the game which uses the database.
The programs are very advanced indeed, and some knowledge is required to obtain maximum benefit from the package. The intention is to show how a fairly advanced program is developed. By following this through you can not only learn the programming 'tricks' used, but more importantly, you can discover HOW to start tackling difficult programs.
...../ continued..........>

Trivia Data Base, continued...

The book discusses the need for user friendliness, error checking, gives an introduction to the use of databases. Sub programs are used, and there are lots of flowcharts. This is not a book for beginners: conversely, most owners will benefit by looking through the listings. The program itself is not useless, and you will probably be able to use quite large chunks of the code in your own programs. If you would like to develop a database, read this first.

THEFT:

Sorry to say some visitors left the Manchester Convention with goods they didn't pay for. I doubt if any were members of TI*MES, but in case they read this: Why not give a proper reward to the few who strive to support the TI? Send the cost of your freebie to the relevant stand holders today. Anonymously is OK. Even one theft can have a big effect when you trade on low margins.

PIRACY:

Really another form of theft, and also a criminal offense. Take a look at your software collection... is every program paid for? Every program that is NOT paid for brings the termination of software support for the TI that much sooner. Unless you are scrapping your console next month, fight piracy strongly. Don't give your friends copies of programs you have bought... unless you fully understand that next time you wish to buy a program there may be none to buy, as the programmer stopped writing due to insufficient royalties, and the publisher went bankrupt due to insufficient sales.

BUG ! ! !

Is anyone out there reading RAMBLES? Nobody tells me when I put my big (well, wide!) foot in it....TI*MES issue No.6, page 55, 6 lines from bottom:

```
  650 CALL PEEK(-31747,A)
```

RUBBISH... that really should have been:

```
  650 CALL PEEK(-31794,A)
   to test for end of tune.
```

A fully revised program in the next issue, pulling in all the amendments!

NAVARONE SUPER BUGGER:

SEE LATER FOR HOW TO GET IT IF YOU HAVENT GOT IT ALREADY....

A magazine called MICROpendium, November 1984, suggests that there is a bug in superbugger, inasfar as you must print to a printer: it does not like to printing to a disk drive, for later use with TI Writer.

The following amendments are suggested, and can be made by loading SBC with MiniMemory and using EASY BUG to make the changes:

It is assumed that you have initialised and then loaded SBC, and no other machine code program is in there!

ADDRESS:	A15A	B2DE	B2F2	B32A	B342	B356	B366	B37A	B382
PRESENTLY:	3F20	7F00	3F09	7F20	7F05	7F00	3F09	7F00	3F09
CHANGE TO:	101F	0FFF	1009	101F	1005	0FFF	1009	0FFF	1009

These locations are all references to either the PAB or data buffer which is used by DSRLNK.

This information is passed on through TI*MES without liability!! Try it and see!

A small picture produced from a program sent in by Sean O'Brien

SHAPING TI SOUNDS by Sean O'Brien:

Wondering how to get your TI to sound better? Well, here is the answer:

When specifying a CALL SOUND on the TI, you must enter:
  DURATION
  FREQUENCY
  VOLUME
    e.g. 10 CALL SOUND(1000,239,0)

The duration being a number between 1 and 4250.

The frequency is a number from 110 and 44733, and the volume is a number between 0 and 30: 30 being the quietest.

The TI also offers you the capability of using one of eight noises. These go in place of the frequency:
  e.g. 10 CALL SOUND(1000,-3,0)

The noises are very good for explosions and go from -1 to -8.

Single note sounds do not go very well in some programs that play music, TI lets you define a three channel sound (or chord). You can also include a noise:

```
  10 CALL SOUND(-1000,I10,0,111,0,112,0)
  20 GOTO 10
```

or

```
  10 CALL SOUND(-1000,110,0,111,0,112,0,-3,0)
  20 GOTO 10
```

The minus value in the duration cuts out the problem of a delay between two sounds, very useful in some short music programs.

You want a short routine that can give interesting results, so try this:

```
   10 FOR V=1 TO 30 STEP 2
   20 FOR T=110 TO 230 STEP V*2
   30 CALL SOUND(-100,T,V)
   40 NEXT T
   50 NEXT V
```

You can replace line 30 with: CALL SOUND(-100,T,V,T+1,V,T+2,V)
                         or with: CALL SOUND(-100,T,V,T+1,V,T+2,V,-8,0)

This short routine will change the volume from 1 (loud) to 30 (hsssh) and at the same time jump through the frequencies from 110 to 230 stepping through the volume, doubled. As the volume reduces, the stepping is larger and the sound 'gets faster'.

Small routines are great for many things in a program, losing a life for instance:

```
 10 for l=230 to 112 step -10
 20 CALL SOUND(100,L,4,L-1,4,L-2,4,-3,6)
 30 NEXT L
```

Notice that to generate interesting sounds it is usually necessary to include loops like this.

The next program will read two numbers from a DATA statement, and will perform the sound loop using them. It will check the final figures to see if they are the same as the final two data values, and if so, it will RESTORE the data statements ready to be re-read. Otherwise it will take the next two data values and use them.

continued.......>

cont....

If you change the numbers in the last DATA statement, change also the CHECK value (line 260) or the program will crash! Just add 100 onto the last value used.

This program runs slightly better in Extended Basic due to the faster processing speed:

```
100 RESTORE 290
110 CALL CLEAR
120 CALL SCREEN(5)
130 READ A,B
140 FOR LOOP=1 TO 10
150 V=30
160 FOR L=0 TO 30 STEP 10
170 FOR K=A TO B STEP 5
180 CALL SOUND(-1000,K,L,B,V,A,V)
190 V=V-1
200 IF V=3 THEN 210 ELSE 220
210 V=30
220 NEXT K
230 NEXT L
240 A=A+10
250 B=B+10
260 IF B=999 THEN 100
270 NEXT LOOP
280 GOTO 130
290 DATA 600,650
300 DATA 110,120
310 DATA 893,899
320 END
```

I have received from AUSTRALIA two programs for review and possible UK sale: they are of an extremely HIGH quality and I am negotiating UK sales rights, hopefully more details in next issue. We have one game (with a sprite moving so fast you will wonder how collisions are detected) and one extremely powerful listing utility (23k of program!). Thanks to FUNNEL WEB for sending it. It is pleasant to receive such professional software from an area of the world not only geographically remote, but one we hear little of as regards the TI.

SPECIAL MEMBERS OFFERS:

NAVARONE SUPER BUGGER, for Ed/As, MiniMem or ExBas +32k ram.
On disk only. Price £4.00 inc p&p

TI WRITER MODIFICATION. True lower case with EDITOR. Initial form feed on FORMATTER removed, and the printer name of your choice as default for FORMATTER. On disk only. Price £4.00 inc p&p

MULTIPLAN REWRITE. Extensive rewrite resulting in much faster set up and data entry. Slightly faster recalc. Auto key repeat. On disk only. £4.00

NB: TI WRITER & MULTIPLAN: You will require the relative TI module and documentaion.

TI FORTH: Program disk only, £4.00
Manual, ring bound, laminated cover, inc p&p, £34.00

PRICES FOR UK ONLY: OVERSEAS PLEASE ASK BEFORE SENDING FUNDS

If ordering more than one disk: One disk, £4, Two disks £7, Three disks £10
OR Send your own disk plus adequate packaging, and cost is:
One disk £2, two disks £3, three disks £4

A copy of the TI Forth manual can also be lent for you to copy, deposit £34, of which £32 is refunded on safe return of the loan copy.

PRK CALLS: Xeroxed booklet detailing extra calls (CALL A, CALL D, CALL G etc) available in TI BASIC when you have the PRK or STATISTICS modules inserted. £1.50

FROM: STEPHEN SHAW
10 Alstone Road STOCKPORT Cheshire ENGLAND SK4 5AH

COMPUTER HOME SERVICE

SAMS TOOL KITS HELP MICRO PROGRAMMERS EVERYWHERE

The *Tool Kit* series teaches you how to program with modular subroutines—the faster programming method used by professionals who don't have a lot of time to spend cranking out BASIC code, line by line. Features some 70 different color, sound, music, graphics, animation, and computational BASIC subroutines as "tools" you can combine, modify, and recombine to form 4 traditional games, 4 arcade games, and 5 educational programs. All program lines are fully explained.
Buchholz and Dusthimer

TI-99/4A EDITION
216 pages, 8 × 9¼, Soft
ISBN 0-672-22310-4, © 1984
No. 22310

TI-99/4A: BASIC LANGUAGE REFERENCE MANUAL

Forms a complete reference manual to TI BASIC and Extended BASIC for users of the TI-99/4A, including the rules for formulation of expressions, information on file processing, and detailed descriptions of all commands and statements, with examples and error analysis. Contains more than 130 sample programs to illustrate usage, plus color and sound tables.
Casciato and Horsfall
312 pages, 5½ × 8½, Comb
ISBN 0-672-22246-9, © 1984
No. 22246

TI-99/4A: 51 FUN AND EDUCATIONAL PROGRAMS

A series of 51 programs in BASIC th you can run as is on the TI-99/4A, adapt to almost any other brand computer. Begins with easy, short programs and progresses to those which are longer and more comple in order to ensure thorough coverag Ideal for first time computer users any age.
Gil M. Schechter
BOOK: 80 pages, 5½ × 8½, Soft
BOOK/TAPE: ISBN 0-672-26168-5

ENTERTAINMENT GAMES IN TI BASIC AND EXTENDED BASIC

A highly organized, fully listed collec tion of 20 original game programs fo the TI-99/4A computer, 9 of which are in standard TI BASIC with the re mainder in Extended BASIC. Each line of code is clearly explained in term of its relationship to the whole program, and a unique how to play it section precedes each game. About half are arcade type games, with the remainder assorted. Provides you with hours of inexpensive entertainment.
Ton and Ton
BOOK: 176 pages, 5½ × 8½, Soft
BOOK/TAPE: ISBN 0-672-26169-3

TI-99/4A: 24 BASIC PROGRAMS

Provides you with an inexpensive source of fun and useful, completely tested BASIC programs that take ful advantage of your TI-99/4A's sound and graphics capabilities. All programs can be run as is, or easily customized as your understanding of BASIC increases. Also covers fundamental programming commands, debugging, utilities, and more.
Casciato and Horsfall
BOOK: 160 pages, 5½ × 8½, Soft
BOOK/TAPE: ISBN 0-672-26172-3

SAMS

Entertainment Games in TI BASIC & Exten BASIC	£12.50
TI99/4A 51 Fun & Ed Programs	£9.50
TI99/4A BASIC Language Ref Manual	£14.50
Took Kit Series: TI-99/4A Edition	£6.95
TI-99/4A BASIC Programs	£13.50
TI-99/4A 24 BASIC Programs	£15.95

PLEASE SEE

CLASSIFIED

ADVERTS