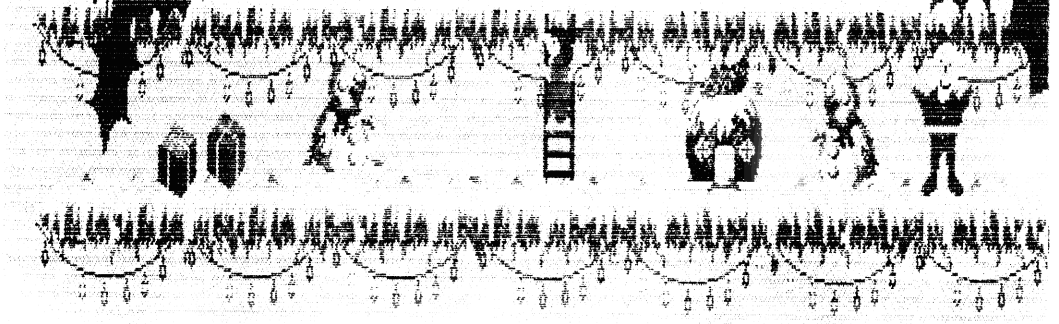
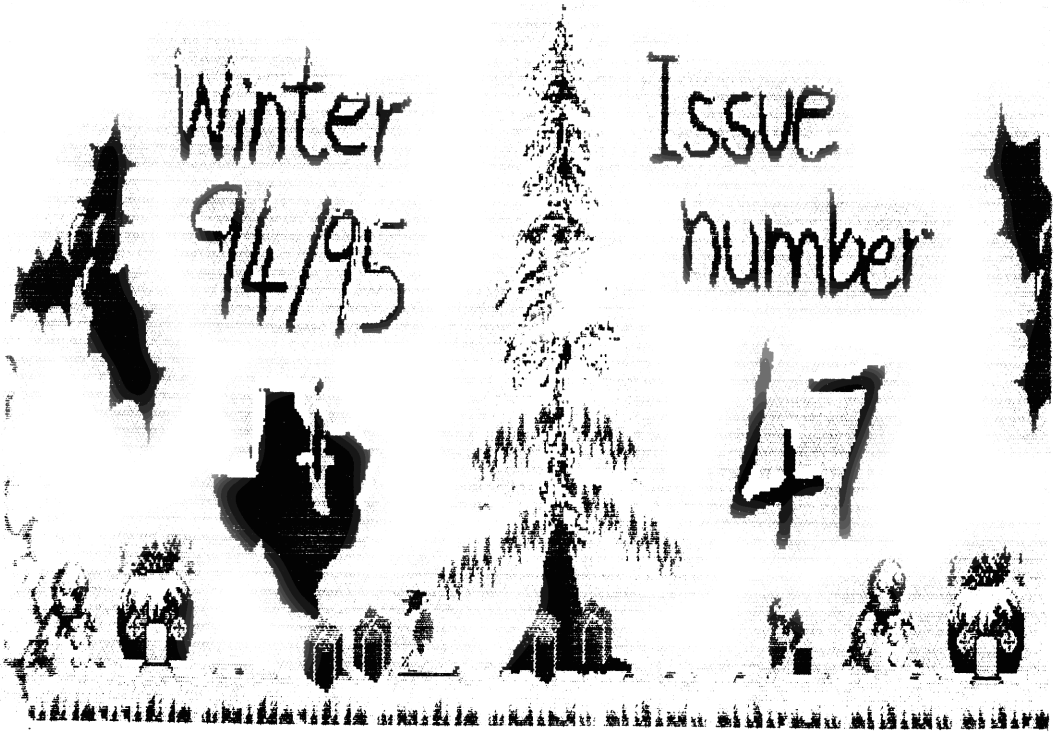
TI*MES



Winter 94/95

Issue number 47



TI-99/4A USER GROUP U.K. COMMITTEE MEMBERS

Mr. Chairman: Trevor Stevens.
249 Southwell Road East, Rainworth, Notts. NG21 0BN
Telephone: 01623 793077

Vice Chairman / Cassette Librarian: Mark Wills
20 Cocayne Green, Harlescott Grange, Shrewsbury. SY1 3QS
Telephone: 01743 350588

General Secretary: Richard Twyning
24 Peel Road, Mansfield, Notts. NG19 6HB
Telephone: 01623 27670

Membership Secretary: Alasdair Bryce
51 Dumbuie Avenue, Silverton, Dumbarton. Scotland. G82 2JH
Telephone: 01389 65903

Treasurer: Alan Rutherford
13 The Circuit, Wilmslow, Cheshire. SK9 6DA
Telephone: 01625 524642

TI*MES Editor: Gary Smith
55 Boundary Road, Newark, Notts. NG24 4AJ
Telephone: 01636 706767

Disk Librarian: Stephen Shaw
10 Alstone Road, Stockport, Cheshire. SK4 5AH

Module Librarian: Francesco L. Lama
14 Granville Court, Cheney Lane, Oxford. OX3 0HJ
Telephone: 01865 721582

NEXT COPY DATE
1st March 1995

CONTENTS

THE ULTIMATE "ACCEPT AT"
software by Bruce Harrison

(This should be very useful to XB programmers or those who like to modify existing XB programs. The disk is available from your User Group disk " to the library. The software DOES NOT work properly on a system with an AVPC card. It has not been tested on a Geneve or on a system with a TIM. Below are excerpts from Bruce's ULTIMATE ACCEPT AT documentation.)

"We've all used the Extended Basic ACCEPT AT from time to time in programs, and many have expressed the wish that it could handle strings longer than 28 characters. Some have also wished it could put the prompt on screen for inputs, instead of using a separate DISPLAY AT. Some things we liked about ACCEPT AT were, for example, the ability to specify the length of input to accept, and the ability to use a negative length so that a default "answer" could be placed on the screen before the ACCEPT.

The "ULTIMATE ACCEPT AT" is an attempt to take all the good features in the existing routine, then add features that were perhaps always needed. Making this all work in a fairly short routine (less than 2000 bytes) was helped by being able to use the TI's LINE EDITOR function through GPLLNK. Using that line editor made the routine simpler and easier to create, and left us free to include the "fancy stuff".

THE CALL LINK

This routine is exercised through the XB CALL LINK process, with either six or seven parameters to control its operation. The CALL LINK looks like this:
CALL LINK("ULTACC",R,C,CL,"PRMP",CHRS,VAR[,B])

The first two parameters, R and C stand for Row and Column, just as in a normal ACCEPT AT. The third, CL, can have three possible states. Putting 0 in for this parameter will do nothing. Putting 1 in will cause the screen to be cleared before the ACCEPT happens. Putting 2 in there will cause the computer to reset all character sets, color tables, and so on to a state much as having just started Extended Basic.

The parameter "PRMP" is the prompt for the user. This is limited to 28 or fewer characters, and may be either a direct quoted string like "INPUT A NUMBER " or a string variable that contains the desired prompt. It's important to include a space at the end of the prompt, so that the input field will not be jammed up against the prompt.

The fifth parameter, shown here as CHRS, is simply the number of characters to be allowed in the input field. For strings, this number can range from 1 through 255. For numeric inputs, it will be forced to 32 characters by the routine.

The sixth parameter is the variable into which the input is to be accepted. This may be either a string or a numeric variable. It may also be a specific member of an array variable, such as A$(I), or N(I).

THE OPTIONS

There are many ways to use this new ACCEPT routine, so let's cover a few variations. If no prompt is desired, for example, a null string (,"",) can be placed where the prompt would go, and this will cause no

prompt to appear. If the allowed length parameter is given as a negative number, then the existing value of this variable will be placed on screen in the input field as a default entry. If the length allowed is a positive number, the input field will be initially blank. A seventh parameter can be added, and this may be anything you like. The simple presence of a seventh parameter of any kind will cause the beep tone to sound. Thus the seventh parameter could be added to the above LINK as (,"BEEP"), and the routine will produce the beep when it's ready for input. Without a seventh parameter, no beep will be heard. These various options may be excercised in any combination you like.

OPERATION

Except for the screen clearing, the routine will start doing things at the location given by the Row and Column parameters. If a prompt has been included, that prompt will appear starting at R,C on the screen. The routine will clear out enough space for the prompt and the designated length for the entry field, then will either place a default there or not, depending whether the length was positive or negative.

The routine protects itself (and you) against the mistake of not having enough room for the designated input length on the screen. Suppose, for example, you started with R=23, C=1, and specified a string input with a length of 70 characters. Obviously that long a string will not fit in just two screen rows, so the routine will move your input field up by enough rows to make room for the desired input length. (As with XB's ACCEPT AT, there are only 28 columns used on each row of the screen.)

The routine also protects against the situation where the existing length of a string is greater than the allowed length given in the LINK. If that's so, the string will be truncated to the length allowed. If you want to be sure of avoiding truncation, the best way to do that is to always specify length at either one more than the expected maximum, or at 255 in all cases. Of course specifying at 255 would limit you in terms of how far down the screen you can do your ACCEPT. Normally a more sensible "safe" entry would be perhaps 81 characters. (None of this applies for numeric entries, where the allowed length is always 32 characters.)

String entries are accepted "as is", with no checking on their content. For numeric entries, there is also no checking performed on the content of the field. If it is non-numeric, it will simply cause a zero (0) value to be placed in the variable, with no error report.

FUNCTION KEY ACTIONS

As in the case of Extended Basic's normal Input and Accept At, Function-1 will delete the character at the cursor position. Function-2 will initiate Insert mode at the cursor's position. Function-3 will erase everything currently in the input field. Function-4 though Function-9 will have no effect. Function-S and Function-D will move the cursor left and right, respectively, Function-X or Function-E will have the same effect as ENTER, and Function-= will exit to the TI Title screen.

THE ERROR TRAPS

As with any Harrison product, we have tried to pre-think what kind of errors the programmer might make in trying to use this routine, and have provided on-screen reports in plain English for errors that can be anticipated.

EXTRA FEATURES

For those cases where you're using this routine and want the ability to just re-set Extended Basic to its default colors and character sets, etc., there is a second "entry point" in the ULTACC/O object file. Once the routine is loaded, the re-setting of XB (and screen clearing) can be accomplished by simply:
CALL LINK("ULTCLR")

No parameters are required. There is yet another bonus supplied on this disk, an object file called RSXB/O. This allows access to the same service as ULTCLR, by CALL LINK("RSXB"). This is provided for those situations where you don't need the Ultimate Accept At, but still want the re-setting capability available.

THE DEMOS

The disk includes demos, which have the routines embedded in the XB program with ALSAVE. The main one is called ULTDEMO, and it shows off some of the features of the routine. The second is called RSTDEMO, and it sets up a screen character font and changed colors, then uses RSXB to clear everything back to the normal XB conditions.

THE 99/4 HOME COMPUTER
description of an antique by Charles Good
Lima Ohio User Group

A SHORT HISTORY OF THE TI HOME COMPUTER

TI began shipping the 99/4 (copyright 1979 on the color bar title screen) in October 1979. It cost $1150 bundled with a 13 inch color monitor (FORTUNE, December 3, 1979, p.54). Initially you had to take the monitor and could not purchase the 99/4 separately, and most purchasers had to pay close to full price. Bundling was necessary because the 99/4 console passed but TIs TV modulator initially failed to pass FCC lab tests for noninterference with radio and TV broadcast reception.

The modulator emitted too much RF radiation (BUSINESS WEEK, March 19, 1979, p.37). However, at that time the FCC did not regulate RF radiation from computing devices not hooked directly to TVs. So TI got around the FCC regulations by offering to the public a "complete package".

It wasn't until January 1, 1981 that the FCC began testing ALL computers likely to be used in a home environment for TV/radio broadcast interference (POPULAR COMPUTING, November 1981, p.6). TI eventually came up with a TV mudulator that would pass FCC tests and on November 28, 1980 began selling the console and monitor separately. The console's list price was $650 (BUSINESS WEEK, December 8, 1980, p.28). This was in one respect was actually a price increase, because the separate prices of the console and monitor were $250 more than their previous bundled price.

TI never published any sales data for the 99/4, but an independent market research firm estimated that TI would sell 25000 between its introduction and the end of 1980 (FORTUNE, June 16, 1980, p.139). During the summer of 1981 TI quietly introduced the 99/4A with a list price of $525. By the time production of the 99/4A ceased in late 1983 or early 1984 the store price for a brand new 99/4A was $50, and over 1 million, perhaps several million 99/4As had been sold.

SUMMARY OF DIFFERENCES BETWEEN THE 99/4 AND 99/4A.

The most obvious differences are the keyboard, the lack of lower case letters on the "4", and the "4"s EQUATION CALCULATOR. Most "4"s have an earphone jack on the front for private listening, but mine doesn't. I will discuss most of these obvious differences in detail. Other differences are listed in an accompanying article by Mike Wright. The 4A gets its "A" from the fact that it has a 9918A video processor, whereas the 99/4 has a 9918 video processor. The 9918A has bit map mode, which is not found on the 9918 processor. This means that any software that uses bit map mode will not run on the 99/4. Other differences between the 99/4 and 99/4A (such as the "4"s lack of an XOP assembly directive) are referenced in the index of the Editor/Assembler manual (p.456) under the heading "Computer differences".

In general, all software written for the "4" will run on the 4A. Some complicated routines on the 4A were required to achieve this compatibility. The "4" has 256 bytes more free memory in TI BASIC than the 4A, so some BASIC software written on a "4" may not work on an unexpanded 4A.

Lots of assembly or GPL software written for the 4A will NOT work on the "4", and there is no easy way to upgrade a "4" to a 4A. The Mini Memory module and its line by line assembler, and the E/A module and its editor and assembler work OK on the "4".

A partial list of "won't work on the 99/4" software includes TI-Writer, Multiplan, Funnelweb v4.x, all the Milton Bradley game modules that were created to accompany the MBX system, Word Invasion, Parsec, Story Machine, Alpiner, Dragon Mix, and Word Radar. Most of these modules are probably incompatible because they use bit map mode. There are probably other reasons for the incompatibility of Multiplan, TI-Writer, and Funnelweb. Even the non-editor parts of Funnelweb won't work on the "4". (The internal KSCAN must be used in order to read a /4 keyboard-sjs). When you boot Funnelweb into the "4" using the extended basic module, the title screen shows blanks where there should be lower case letters. (The /4 did NOT have lower case characters!). You can then go to Funnelweb's extended basic user list, but here the "4" locks up. You can't boot any software from the XB user list.

THE KLUDGY 99/4 KEYBOARD

After playing around with my "4" for a couple of months, I am forced to agree with the statement made in an accompanying FORTUNE magazine article. The 99/4 is a real dog, mainly because of its keyboard.

There are 41 "chicklet" style keys, each slightly contoured and shaped like a narrow rectangle. The 4A keyboard has 48 keys. Although each 99/4 key depresses separately, the keys are not what experienced users would call "full travel" There is no tactile response, no click, before the keys suddenly bottom out at the end of their downward travel.

Non-alphanumeric keys include one (and only one) SHIFT, an ENTER, a SPACE bar, and a SPACE key immediately to the left of the "A" key. Alpha keys always produce upper case letters, so the SHIFT key is not used as often as it is on the 99/4A. There is are no ALPHA LOCK, FCTN, or CTRL keys on the "4". The "4"s SPACE key and bar do exactly the same thing, leave a blank space. I can see no reason at all for this space KEY, in addition to the normally positioned space bar. There are ASCII characters built into the 99/4 console that are not implemented on its limited keyboard, yet there is this stupid extra space key.

Touch typing on the 99/4 is difficult. The keys are spread apart the same distance as on the familiar 99/4A keyboard, so it is possible to get all your fingers at once onto the keys. But the small vertical size of the keys and their lack of tactile feel makes touch typing difficult. The small size and minimal contour of the "4"s keys makes it difficult for a touch typist to find by feel and seat his or her fingers in the center of the desired keys as the fingers move blindly around the keyboard. The fully contoured much larger keys of the 4A (larger because there is less space between keys) makes touch typing much easier.

A special problem to experienced touch typists is the lack of any key to the right of the "L". This means there is no "home" key for the little finger of the right hand to touch, and this will drive most touch typists crazy. Frequently, when I try to type on my "4" I end up accidently moving my fingers over one key to the left on the home key row so that all ten fingers have something to touch. My left hand pinky finger is then on the useless SPACE key instead of on the "A" where it should be. Then I type rtow fevfw. TI recognized this problem. The only application software written for the 99/A that is likely to require touch typing, the Terminal Emulator II, has a keyboard overlay with a raised area creating a fake key for the right hand's little finger.

TI provided a series of overlays specifically for use with the 99/4 and not usable with the 4A. Some overlays were packaged with the "4" and others were available with specific command modules. Because of the narrow vertical size of each key there is enough room between rows of keys on the "4" to display a text prompt immediately above ANY key, not just above the numeric keys as is the case with the 4A. The overlays have text prompts for special keypresses, and cover the entire "4" keyboard, with the keys sticking up through holes in the overlay.

Special keypress usually involve using the SHIFT key in combination with a letter key. One overlay packaged with the "4" shows the editing keys used in BASIC. SHIFT/Q=quit. SHIFT/W=begin. SHIFT/ESDX= arrows. SHIFT/R=redo. SHIFT/T=erase. SHIFT/A=aid. SHIFT/F=delete. SHIFT/G=insert. SHIFT/Z=back. SHIFT/C=clear. SHIFT/V=proceed.

There is nothing intuitive about some of these keypresses (why not SHIFT/B instead of /Z for back), so the overlay is really needed. Another overlay packaged with the "4" shows the split keyboard keys that can be used with some games to simulate the 8 positions of joysticks #1 and #2. In addition to the overlays packaged with the computer, I have seen overlays designed for use with the following command modules: Terminal emulator I, Terminal emulator II, Video graphs (PHM3005), and Video Chess. There may be other overlays I havn't seen.

ONLY UPPER CASE LETTERS

No keypress on the "4" keyboard will give ASCII codes 97-122, the lower case letters. Everything you type is in upper case, and this means you only use the SHIFT key in routine typing to shift the numeric keys and display !@@#$% &&*(). The 99/4 uses a 5x6 pixel grid to display upper case letters. The 99/4A uses a 5x7 grid to display both upper case and lower case text. If you load into the "4" BASIC software written on a 4A that includes lower case text, the program seems to work OK, but no lowercase letters are displayed on screen.

THE EQUATION CALCULATOR

When you PRESS ANY KEY TO CONTINUE from the color bar powerup screen of the "4", you get a menu with three choices. Press 1 for TI BASIC, 2 for EQUATION CALCULATOR, 3 for TITLE OF COMMAND MODULE.

The EQUATION CALCULATOR is a way of using the "4" in mathematical calculations without having to write a BASIC program to do the calculations. You can do simple arithmetic, and you can also use exponential numbers, PI, SQR, exponents, SIN, COS, TAN, and ATN in your calculations. Everything that can be done using EQUIATION CALCULATOR can also be done using a TI BASIC program, or directly from BASIC command mode.

The EQUATION CALCULATOR screen is divided into three sections. The bottom section is where you do your calculating. You can, for example, type in a simple calculation such as 1567+56.98-145+(12/98), press <enter>, and display the answer. To do the same thing in BASIC command mode, you would have to type PRINT before you typed the numbers of the calculation. A single calculation is limited to 28 characters (one line of text).

You can define variables such as LENGTH=60, press the up arrow, and have this variable stored in memory and permanently displayed in the upper third of the EQUATION CALCULATOR screen. You can display up to 6 variable names and their current values on screen in this way and not have to worry about the display scrolling off the top of the screen. You can do the same thing in BASIC command mode by pressing <enter> after typing LENGTH=60. The value of LENGTH would be stored in the computer's memory, but it would only remain on screen until it scrolled off the top due to subsequent entrys.

You can also define an equation such as PERIMETER=2*LENGTH+2*WIDTH and store this equation in the middle part of the EQUATION CALCULATOR screen. You can then define the values of the variables LENGTH and WIDTH, use the down arrow to bring the equation into the bottom work area of the EQUATION CALCULATOR screen, press <enter> and display the current value of PERIMETER. You can then redefine LENGTH and/or WIDTH, and reuse the equation to calculate the new value of PERIMETER. You can also store equations for repeated use in a BASIC program, although you cannot store such an equation in memory in BASIC command mode. In command mode you would have to retype the equation each time.

I don't think EQUATION CALCULATOR is very useful. Apparently TI idn't either, because they dropped it when the 4A was released. From BASIC (a program or from command mode) you can do all the same things, and more. The main limitation of EQUATION CALCULATOR is the 28 character size of a formula or chain calculation. The most common routine calculating I do on my 99/4A is to balance my checkbook. I enter BASIC command mode and type PRINT, followed by my initial bank balance, followed by all my subsequent withdrawals (as minus numbers) and deposits (as positive numbers).

Before I press <enter> to display my balance I can check the screen to see that all the numbers in the calculation are typed correctly and use INSERT or DELETE to correct mistakes. Such a long chain calculation requires several lines on the screen to display all the digits before pressing <enter>. TI BASIC command mode gives me 4 lines. EXTENDED BASIC command mode gives me 5 lines. EQUATION CALCULATOR allows me only one line of digits.

CONCLUDING REMARKS

When it was released in 1979 the 99/4 was the only consumer device that could really be called a "Home Computer". It was the first to utilize cartridge software. Its speech synthesis was, and still is, unequaled. It was easy to use, easy to program in BASIC, and it was powerful. Its high price was probably the major reason for its initially limited sales. Its rotten keyboard didn't help either. I'm sure glad we now have the 99/4A. The 4A is much superior to the "4".

TIPS FROM THE TIGERCUB
#35
Copyright 1986

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

The 4/86 Micropendium had a rather slow routine to count the number of words in a D/V text file. I think the following will be much faster. It ignores any lines beginning with a period (TI-Writer formatter commands), otherwise counts each cluster of characters followed by a space, plus the last cluster on the line.

```
10 !WORDCOUNT by Jim Peterso
n
100 DISPLAY AT(12,1)ERASE AL
L:"INPUT FILENAME? DSK" :: A
CCEPT AT(12,20):F$ :: OPEN #
1:"DSK"&F$,INPUT
110 A=1 :: LINPUT #1:M$ :: I
F ASC(M$)=46 THEN 130
120 X=POS(M$," ",A):: IF X=0
 THEN 130 :: IF X=A THEN A=X
+1 :: GOTO 120 ELSE F=1 :: C
=C+1 :: A=X+1 :: GOTO 120
130 C=C+F :: F=0 :: IF EOF(1
)<>1 THEN 110 :: CLOSE #1 ::
 DISPLAY AT(12,1)ERASE ALL:"
APPROXIMATELY "&STR$(C)&" WO
RDS"
```

==========

```
100 !TIGERCUB GRAPHPRINT by
Jim Peterson
110 !Will output to printer
a line graph of 31 items of
data, as for instance the
temperature for each day of
a month
120 !Values must be positive
 integers within a range of
75 from minimum to maximum
130 M$=RPT$("!_",65):: DIM T
$(31),D$(75):: MN=10000
140 DISPLAY AT(12,1)ERASE AL
L:"Input data - maximum 31":
"items. Enter to finish"
150 FOR X=1 TO 31 :: DISPLAY
 AT(14,1):X;TAB(4);CHR$(1)::
 ACCEPT AT(14,4)VALIDATE(DIG
IT)SIZE(-5)BEEP:T$(X):: IF T
$(X)=CHR$(1)THEN X=X-1 :: GO
TO 170
160 T=VAL(T$(X)):: MX=MAX(MX
,T):: MN=MIN(MN,T):: NEXT X
170 RN=MX-MN :: IF RN>75 THE
N PRINT "EXCEEDS MAXIMUM RAN
GE OF 75" :: STOP
180 IF MX>75 THEN AD=MX-75
190 OPEN #1:"PIO",VARIABLE 1
32 :: PRINT #1:CHR$(15);CHR$
(27);CHR$(51);CHR$(12):: PRI
NT #1:RPT$("_",132)
200 DISPLAY AT(12,1)ERASE AL
L:"Wait, please...": :".....
.this takes time"
210 LM=LEN(STR$(MX)):: FOR J
=1 TO 75 :: J$=STR$(76+AD-J)
220 IF J>66+AD THEN J$=J$&"
"
230 IF J/2=INT(J/2)THEN D$(J
)=RPT$(" ",LM)&SEG$(M$,1,132
-LM)ELSE D$(J)=J$&SEG$(M$,1,
132-LM)
240 NEXT J :: PRINT #1:RPT$(
" ",LM)&SEG$(M$,1,132-LM)
250 J=1 :: T=VAL(T$(J))-AD :
: T=76-T :: D$(T)=SEG$(D$(T)
,1,J*4+4)&CHR$(239)&SEG$(D$(
T),J*4+6,255):: J=J+1
260 T2=T :: T=VAL(T$(J))-AD
:: T=76-T :: FOR N=T2 TO T S
TEP (T2>T)+ABS(T>=T2):: D$(N
)=SEG$(D$(N),1,J*4+2)&CHR$(2
53+(T<T2))&SEG$(D$(N),J*4+4,
255):: NEXT N
270 J=J+1 :: D$(T)=SEG$(D$(T
),1,J*4)&CHR$(239)&SEG$(D$(T
),J*4+2,255):: IF J<=X THEN
260
280 FOR J=1 TO 75 :: PRINT #
1:D$(J):: NEXT J :: PRINT #1
290 T=8 :: FOR J=1 TO 31 ::
PRINT #1:TAB(T);STR$(J);:: T
=T+4 :: NEXT J
```

I still think of the TI as a HOME computer, and I still think that the home computer is an invaluable educational tool - but I guess not many folks agree with me. I had thought of writing full disks of a progressive series of lessons on one subject, but my present two full disks of math education have sold a combined total of 7 copies in 7 months, so that would obviously be a waste of time.

I had written this next

```
THEN RETURN
320 M$(X)=SEG$(M$(X),1,T-1)&
C$&SEG$(M$(X),T+1,255):: M$(
X)=SEG$(M$(X),1,P-1)&"c"&SEG
$(M$(X),P+1,255):: GOSUB 340
 :: F=1 :: RETURN
330 M$(1),M$(2),M$(8),M$(9)=
A$ :: M$(3)="~~123~~~~" :: M
$(4)="~~456~~~~" :: M$(5)="~
~78cAB~~" :: M$(6)="~~~~CDE~
~" :: M$(7)="~~~~FGH~~" :: R
ETURN
340 FOR J=8 TO 16 :: DISPLAY
 AT(J,10):M$(J-7):: NEXT J :
: RETURN
350 SUB CALLKEY(R,C,V$,K$)
360 CALL HCHAR(R,C+2,30):: F
OR T=1 TO 3 :: CALL KEY(0,K,
S):: IF S<>0 THEN 390
370 NEXT T :: CALL HCHAR(R,C
+2,20):: FOR T=1 TO 3 :: CAL
L KEY(0,K,S):: IF S<>0 THEN
390
380 NEXT T :: GOTO 360
390 IF POS(V$,CHR$(K),1)=0 T
HEN 360 ELSE K$=CHR$(K)
400 SUBEND
```

[^on disk from library^]

I don't think this is very useful, but somebody asked me for it - it converts decimals to fractions.

```
100 CALL CLEAR :: CALL CHAR(
95,"000000FF")
110 DISPLAY AT(12,1):"Decima
l?" :: ACCEPT AT(12,10):D ::
 T=1
120 IF INT(D)<>D THEN D=D*10
 :: T=T*10 :: DISPLAY AT(14,
1):D :: DISPLAY AT(16,1):T :
: GOTO 120
130 DISPLAY AT(14,1):D :: DI
SPLAY AT(15,2):RPT$("_",LEN(
STR$(T))):: DISPLAY AT(16,1)
:T
140 FOR J=2 TO 5 STEP 3
150 IF D/J=INT(D/J)AND T/J=I
NT(T/J)THEN D=D/J :: T=T/J :
: DISPLAY AT(14,1):D :: DISP
LAY AT(16,1):T :: GOTO 150
160 NEXT J :: GOTO 110
```

Several years ago, John Hamilton wrote a program you could use to key in a program with TI-Writer, then merge it in, delete the "!" after each line number, and run it as a program. Its only problem was with lines of over 80 characters. Since then, better programs have been written - XLATE and TEXTLOADER - which do not require deleting anything but they still have some trouble with long lines and with missing spaces. This little version overcomes those faults but you do have to delete the "!".

Try keying in a program into the Funlweb Editor, be sure to put a carriage return at the end of each program line. When finished, check each program line which has wrapped around to two lines. If the first character in that second line should be preceded by a space, insert a space as its first character. Then save the file with the PF option and run this little program. Enter NEW, merge in the output file by MERGE DSKn.filename, go through it with FCTN X and FCTN 1 deleting the "!" after each line number, and it should run as a program.

```
100 DISPLAY AT(12,1)ERASE AL
L:"Input file? DSK":"":"Outp
ut file? DSK"
110 ACCEPT AT(12,16):A$ :: A
CCEPT AT(14,17):B$
120 OPEN #1:"DSK"&A$,INPUT :
: OPEN #2:"DSK"&B$,VARIABLE
163,OUTPUT
130 LINPUT #1:M$
140 IF POS(M$,CHR$(13),1)=0
THEN LINPUT #1:M2$ :: M$=M$&
M2$ :: GOTO 140 ELSE M$=SEG$
(M$,1,LEN(M$)-1)
150 X=POS(M$," ",1):: Y=VAL(
SEG$(M$,1,X-1))
160 PRINT #2:CHR$(INT(Y/256)
)&CHR$(Y-256*INT(Y/256))&"!"
&SEG$(M$,X+1,255)&CHR$(0)
170 IF EOF(1)<>1 THEN 130 EL
SE CLOSE #1 :: PRINT #2:CHR$
(255)&CHR$(255):: CLOSE #2
```

I had a question from a friend who wanted to key in some pieces of information in Funnelweb and then sort

them. Trouble was, the data tended to be more than 80 characters long. Therefore it was saved as two or more separate records, which a sort scrambled into garbage.

So, how do you create and sort long records of varying length? The easiest way is to let the disk drive controller do it for you. Just type whatever you want, as long as you want, then save it as a separate file, using the first several letters of the text as the filename. Don't include any spaces or periods, of course. If you are using numbers as filenames, pad them with leading zeros to all the same length such as 001 to 999 or 0001 to 1000.

The drive controller will sort those files alphabetically, and this little program will print them in that sequence -

```
100 CALL CLEAR :: DIM F$(127
):: OPEN #1:"DSK1.",INPUT ,R
ELATIVE,INTERNAL :: INPUT #1
:D$,A,B,C
110 INPUT #1:M$,A,B,C :: IF
A=2 AND C=80 THEN X=X+1 :: F
$(X)=M$
120 IF LEN(M$)<>0 THEN 110 E
LSE CLOSE #1 :: OPEN #2:"PIO
"
130 FOR J=1 TO X :: OPEN #1:
"DSK1."&F$(J),INPUT
140 LINPUT #1:M$ :: IF ASC(M
$)<127 THEN PRINT #2:M$
150 IF EOF(1)<>1 THEN 140 EL
SE CLOSE #1
160 NEXT J :: STOP
```

This method is limited by the fact that you can only put 127 files on a disk, but if you have more than one drive you can have 127 on each one, and use this program -

```
100 DISPLAY AT(12,1)ERASE AL
L:"How many drives?" :: ACCE
PT AT(12,18)SIZE(1)VALIDATE(
NUMERIC):D :: DIM F$(510)
110 FOR J=1 TO D :: OPEN #1:
"DSK"&STR$(J)&".",INPUT ,REL
ATIVE,INTERNAL :: INPUT #1:D
$,A,B,C
120 INPUT #1:M$,A,B,C :: IF
A=2 AND C=80 THEN X=X+1 :: F
$(X)=M$&"*"&STR$(J)
130 IF LEN(M$)<>0 THEN 120
140 CLOSE #1 :: NEXT J :: CA
LL LONGSHELL(X,F$()):: OPEN
#2:"PIO"
150 FOR J=1 TO X :: W=POS(F$
(J),"*",1)
160 OPEN #1:"DSK"&SEG$(F$(J)
,W+1,1)&"."&SEG$(F$(J),1,W-1
)
170 LINPUT #1:M$ :: IF ASC(M
$)<127 THEN PRINT #2:M$
180 IF EOF(1)<>1 THEN 170
190 PRINT #2:"" :: CLOSE #1
:: NEXT J
200 SUB LONGSHELL(N,N$())
210 D=N
220 D=INT(D/3)+1 :: FOR I=1
TO N-D :: IF N$(I)<=N$(I+D)T
HEN 250 :: T$=N$(I+D):: J=I
230 N$(J+D)=N$(J):: J=J-D ::
 IF J<1 THEN 240 :: IF T$<N$
(J)THEN 230
240 N$(J+D)=T$
250 NEXT I
260 IF D>1 THEN 220
270 SUBEND
```

A recent article in a news letter reminded me of something I knew long ago but had forgotten. If you have been entering a lot of data into a disk file and the program crashes, all is not lost. Just enter CLOSE #1 in command mode and your data will be saved. If you get a FILE ERROR message, just try CLOSE #2 and so on until you hit the right one.

Many user group newsletter editors use a program that puts a code on the address label to indicate when membership expires. Trouble is, no one ever reads their address label!

This quick & dirty little program requires you to prepare your address file in TI-Writer or Funnelweb with name on first line, address on second, city and state on

third, the fourth line blank or you can use it for additioal address, number of expiration month on fifth line and year on sixth.

Continue with other addresses, making sure you use six lines for each. Such a file is easy to update with TI-Writer. The program will read addresses from that file and print an address label for everyone whose membership has not expired.

It will also optionally print a warning label, which you can slap conspicuously on the front page of the newsletter, if the subscription currently expires or expires next month. If you give a grace period for renewal, you can choose to print an address label and a warning label for those who are one month or two months overdue.

```
100 DISPLAY AT(1,4)ERASE ALL
:"REMINDER LABEL PRINTER"
110 DISPLAY AT(3,1):"Address
 file? DSK" :: ACCEPT AT(3,1
8):F$ :: OPEN #1:"DSK"&F$,IN
PUT
120 DISPLAY AT(5,1):"Printer
? PIO" :: ACCEPT AT(5,10)SIZ
E(-20):P$ :: OPEN #2:P$
130 DISPLAY AT(6,1):"Emphasi
zed print? (Y/N)" :: ACCEPT
AT(6,25)VALIDATE("YN")SIZE(1
):E$ :: IF E$="Y" THEN PRINT
 #2:CHR$(27)&"E";
140 DISPLAY AT(7,1):"Doubles
truck print? (Y/N)" :: ACCEP
T AT(7,27)VALIDATE("YN")SIZE
(1):D$ :: IF D$="Y" THEN PRI
NT #2:CHR$(27)&"G";
150 DISPLAY AT(9,1):"Print p
ending expiration    notice?
 (Y/N)" :: ACCEPT AT(10,15)S
IZE(1)VALIDATE("YN"):PEND$
160 DISPLAY AT(11,1):"Print
current expiration    notice
? (Y/N)" :: ACCEPT AT(12,15)
SIZE(1)VALIDATE("YN"):CUR$
170 DISPLAY AT(13,1):"Print
past    expiration    notice
? (Y/N)" :: ACCEPT AT(14,15)
SIZE(1)VALIDATE("YN"):PAST$
180 DISPLAY AT(15,1):"Print
two months past       expira
tion notice? (Y/N)" :: ACCEP
T AT(16,26)SIZE(1)VALIDATE("
YN"):PAST2$
190 DISPLAY AT(18,1):"Curren
t year?" :: ACCEPT AT(18,15)
:Y :: Y=Y+(Y>99)*1900 :: Y=Y
-92
200 DISPLAY AT(20,1):"Number
 of month?" :: ACCEPT AT(20,
18)VALIDATE(DIGIT):M :: X=M+
Y*12
210 IF EOF(1)=1 THEN 330 ::
LINPUT #1:A$ :: IF ASC(A$)=1
28 THEN 330
220 LINPUT #1:B$ :: LINPUT #
1:C$ :: LINPUT #1:D$ :: INPU
T #1:M,Y :: Y=Y+(Y>99)*1900
:: Y=Y-92 :: M=M+Y*12
230 IF M>=X THEN GOSUB 280
240 IF M=X AND CUR$="Y" THEN
 GOSUB 290 :: GOTO 210
250 IF M=X+1 AND PEND$="Y" T
HEN GOSUB 300 :: GOTO 210
260 IF M=X-1 AND PAST$="Y" T
HEN GOSUB 280 :: GOSUB 310 :
: GOTO 210
270 IF M=X-2 AND PAST2$="Y"
THEN GOSUB 280 :: GOSUB 320
:: GOTO 210 ELSE GOTO 210
280 PRINT #2:A$:B$:C$:D$:"":
"" :: RETURN
290 PRINT #2:A$:"YOUR SUBSCR
IPTION EXPIRES THIS":"MONTH.
 PLEASE RENEW NOW SO YOU":"W
ILL NOT MISS ANY ISSUES":" "
:" " :: RETURN
300 PRINT #2:A$:"YOUR SUBSCR
IPTION EXPIRES NEXT":"MONTH.
 PLEASE RENEW NOW SO YOU":"W
ILL NOT MISS ANY ISSUES":" "
:" " :: RETURN
310 PRINT #2:A$:"YOUR SUBSCR
IPTION EXPIRED LAST":"MONTH.
 PLEASE RENEW NOW SO YOU":"W
ILL NOT MISS ANY ISSUES":" "
:" " :: RETURN
320 PRINT #2:A$:"YOUR SUBSCR
IPTION EXPIRED":"TWO MONTHS
AGO":"THIS WILL BE YOUR LAST
 ISSUE":"UNLESS YOU RENEW PR
OMPTLY":" " :: RETURN
330 CLOSE #1 :: END
```

Memory just about full -

Jim Peterson

The Art of Assembly - Part 5

Useful Subroutines

By Bruce Harrison

This month's article will be relatively short, but it's accompanied by a large dose of source code (see below). The source code for today is all subroutines, one of the High level variety (a subroutine that calls other subroutines) and several smaller ones.

The major purpose in this source code is to get user input from the keyboard, display it stroke by stroke on the screen, then when the ENTER key is pressed, to report out what's on the screen into a string at one specific location in memory. In effect, this is like the Extended Basic ACCEPT AT function for a string variable. The version shown was developed for use in our Golf Score Analyzer program. In this listing, however, we've left out the lines that deal with the character offset for Extended Basic. Thus this subroutine can be easily integrated into any Option 3 E/A type program. The label names used reflect its "Golf" origins to some extent, as the name of the big subroutine CRSIN, short for Course Name Input. In that program, this was actually used for any occasion when we wanted to accept a string of characters from the keyboard.

There is an auxiliary subroutine which we call CLRFLD (clear field) also included in the sidebar. That is used before CRSIN, to clear the screen area into which we want user input. One can also use CRSIN without the CLRFLD, so that something already in that screen location can be edited or accepted as a default entry.

Let's say that we want to accept a 20 character string with a cleared field at Row 12, column 5 of the display screen. Here's what the main program would need to do to invoke the subroutines:

```
LI   R0,SCRWID*11+4  Set R0 to Row 12, col 5
LI   R4,20           Number of characters in R4
BL   @CLRFLD         Clear 20 characters at row 12 col 5
BL   @CRSIN          Accept the input string
```

Note that the subroutine CLRFLD restores the original value in R0 and retains the value in R4 upon exit, so the main program need not reload those two registers before calling CRSIN.

Also please note that this subroutine will not work if R0 is zero. If it's set to a value of 1, the accept will happen at Row 1, Column 2 of the screen. The adept student may modify it so it would work at the screen origin, but we've never found it necessary (or desirable) to accept a string at that screen position.

Before we get further into how this subroutine CRSIN works, we'd better deal again with that business of stacking the return address for this High level case. What's shown here assumes that your program contains other High level subroutines and that somewhere early in the program you'd pointed R15 at a stack location in memory. If this were the only high level subroutine in your program, you could simply stash R11 in R15 itself, so the opening line in CRSIN would read:

```
CRSIN   MOV  R11,R15
```

And the exit point would be:

```
CRIX    B    *R15      Branch to the address in R15
```

The other possible case is that you'd have CRSIN as the first High level subroutine in your program, in which case CRIX would be a label only, and would be followed by the short piece of code shown at label SUBRET.

The subroutine CRSIN uses three others to do its work. For normal keystroke inputs, it uses CURFRC to put the cursor on-screen, then uses KI2 to accept your keystroke into R8. When the input keystroke is one of the two "arrow" keys Function-S or Function-D, the special repeat-key subroutine KI2A is used. Using that subroutine allows the cursor to be moved through the input field by holding down the arrow key. There is a built-in delay in this subroutine, so the cursor will not fly to the end of the field, but move in human-speed steps. The subroutine exits immediately if you release the key. The delay imposed is modified by the subroutine, so the delay after the first cursor move is considerably less than the first move. Moving the byte at location ONE to location KI2A+2 clears the left byte of the immediate value that follows the label KI2A. When you exit by releasing the arrow key, the main subroutine re-sets the delay factor for a first arrow move.

This idea of having the code modify itself while you're using it is tricky, and many programmers shun its use. We considered it a worthwhile thing to do in this instance, to make the movement of the cursor more like what the TI user is accustomed to seeing.

Now let's start at the beginning of the subroutine. Some important things happen there. On entry, after stashing the return address, we clear our insert flag, so that we're sure insert mode won't be on when we didn't ask for it.

Next, we stash the starting value of R0, then move back one location and place an edge character on the screen. We then increment R0, add the length of the allowed string to it, and write another edge character. They are put there so our subroutine will easily be able to distinguish the two ends of the allowed input field. We also save this position of R0 (one beyond the last character to be accepted) for use later on. When operating in most modes, the edge character looks just like a space. This is not true when entering from E/A Option 3, in which case the edge character is a small square. You can redefine it to look like a space by:

```
LI    R0,32*8+>800 Point at space character
LI    R1,TEMSTR Use our temporary string buffer
LI    R2,8      Eight bytes to read
BLWP  @VMBR     Read eight bytes from space
S     R2,R0     Back up to edge character
BLWP  @VMBW     Write eight bytes
```

Finally in this opening section, we subtract R4 from R0 so we're at the first character spot in the field, then stash away the value in R4 for use later.

The section of code starting at CRSIOA is the main operating loop of this subroutine. The first order of business is to grab the character present at this spot on the screen and stash that at location ALTKEY. This will become the character that alternates with the cursor while the cursor is at this position.

The very next thing is to call the little subroutine CURFRC. CURFRC is there so that every time the cursor moves to a new input location, the cursor will appear on-screen, and start a new cycle of blinking. Were this not done, the cursor could become invisible after some of your keystrokes, and we find that disconcerting. Now we call the subroutine KI2 which simply keeps blinking the cursor, alternating with whatever character was there before, until you strike a key on the keyboard.

There are some checks now perormed on the value of the keystroke reported into R8 by KI2. The only one of these that's not immediately obvious is the check for the value 15. That's the ASCII code for Function-9, and behaves the same as if ENTER were struck. In its application within the Golf Score Analyzer, the key combination Function-9 gets you back to the part of the program which called CRSIN, which then uses the fact that you exited CRSIN by Function-9 to escape gracefully from whatever function you were into. If you don't need that feature, you can omit the two lines CI R8,15 and JEQ CRSDMY.

We should at this point admit that this source code has not been subjected to a thorough "scrubdown" effort. The two lines following that compare to 15 and its jump instruction may be unnecessary. We're not going to stop and make that change in the program, but will leave as an excercise for the student the determination. As it is, the subroutine does work, even if it does contain a piece of sloppy coding. Your author is human, like you.

There's another piece of inelegant code in here, concerning label CRSDMY. That stands for DUMMY! During the development of this subroutine, we got into the situation where some of our jumps to label CRSIX were out of range. We could have corrected that situation by adding labels, reversing logic, and including some B @CRSIX instructions. Instead, we wedged in that phony label CRSDMY, which simply makes a second jump to CRSIX. This is really not the soundest practice, but it's a quick, cheap, and ugly way out of a problem. We're not proud of it, but it does assemble and work correctly, so we're leaving it alone. Whenever your author starts to get too elegant with his programming, he remembers a lesson taught by his first mentor in programming the TI, a man named George R. Hendershot. The lesson was "First, get it to work!" One might add a corollary to that, such as "If it ain't broke, don't fix it!"

At label CRSC4, we see whether the insert key Function-2 has been struck. If it hasn't, we move on, and if it has, we set the insert flag (INSFLG) and go back to CRSIO. Once the insert key has been struck, characters entered from the keyboard will be inserted at the current cursor position until insert is cancelled by hitting the arrow keys, Function-9, or ENTER.

The next important keystroke the program looks for is ENTER. If that's been struck, we exit the subroutine. Given it's not the ENTER key, we check for Function-1. If that's been struck, we delete the character at the current cursor position and move all the characters right of that position in the field one spot left. Next there's one final check to see if some other key with an ASCII code less than the spacebar's 32 has been struck. If so, we ignore that keystroke.

Next there's a short section that converts lower case characters to upper case. This may be omitted if you don't need it.

At label CRSI1, we check to see whether the insert flag is set by moving that word into R1 and jumping ahead if the word was zero. If insert was in effect, we perform the steps between JEQ CRSI1A and the label CRSI1A. First, we write the character that was at the cursor position to the screen, then move our variable word ENDOC into R2 and subtract R0 from it. This makes R2 equal the number of characters between the current cursor position and the edge marker at the end of the field. Now we use TEMSTR, which will be the location for the string input when we're finished, as a temporary buffer to hold all the characters from the cursor's position to the end of the field. We then DEC R2, so that the writing back of these characters will not extend to the edge character. If R2 has become zero, that means we're at the last position in the field, so we skip ahead. Now, we increment R0 so we're writing to the next screen spot, and perform a BLWP @VMBW to write the characters back to the screen one space to the right. Finally we decrement R0 so it points to where it was when we started this section of code, and then proceed at label CRSI1A to write the struck key's character to the screen.

Had we not been in insert, we would have jumped to here and put the character on the screen. After writing one character, we increment R0 so it points at the next spot, check to see if the character we've reached is an edge character, and jump back if it is, so we don't exceed the field limit.

The rest is pretty mundane stuff, simply handling the movement of the cursor in response to the arrow keys, so we'll skip ahead to CRSIX, where this string of characters gets "reported out" to the label TEMSTR.

The first order of business is to write back the ALTKEY character to the screen, then set R0 to point at the last spot in the field. Next, we get the field length from location SAV4 into R2. We now start examining the characters in the field in reverse order, looking for a non-space character, and decrementing the count in R2 each time we find a space. This eliminates trailing spaces from the length of the reported string. Once we've found a non-space, we have the length of the string in R2, so we swap the bytes in R2, place the length byte at location TEMSTR, re-swap so R2 has the length as a word value. At this point we check to see if a null string (all spaces) is in the field and get out of here if that's so. Otherwise we set R1 to point to TEMSTR+1, and read the string's content from the screen via a BLWP @VMBR.

When we finish, TEMSTR contains one byte at the beginning to indicate length of the string, plus the string's content. From here, the main program can take the string at TEMSTR and move it to the deired memory location via the small subroutine MOVSTR, which was included in Part 2 of this series.

As the saying goes, use it in good health. This subroutine can make your life a bit easier when you are writing a program. If it does that, in addition to adding to your knowledge of Assembly programming, then it's been worth the effort.

In our next article, we'll discuss, among other topics, the business of entering and returning gracefully from programs. We'll also discuss some of the ramifications of working with Assembly programs started from Extended Basic.

```
* SUBROUTINES WHICH MAY PROVE USEFUL
* DESIGNED FOR USE IN OPTION 3 E/A PROGRAMS
* CODE BY BRUCE HARRISON - PUBLIC DOMAIN
* 22 JUNE 1991
*
* REQUIRED REFERENCES
      REF  KSCAN,VMBW,VMBR,VSBW,VSBR
*
* REQUIRED EQUATES
STATUS EQU  >837C
KEYADR EQU  >8374
KEYVAL EQU  >8375
*
* THE FOLLOWING SUBROUTINE ACCEPTS A STRING OF CHARACTERS STARTING AT LOCATION
* POINTED TO BY R0, NUMBER OF CHARACTERS TO ACCEPT MUST BE IN R4
* INPUT STRING IS PLACED AT LOCATION TEMSTR
*
CRSIN
      MOV  R11,*R15+    STACK RETURN ADDRESS
      CLR  @INSFLG      CLEAR OUR INSERT FLAG
      MOV  R0,@PGNUM    STASH R0 IN MEMORY LOCATION
      DEC  R0           DECREMENT R0
      MOVB @EDGE,R1     PLACE EDGE CHARACTER IN LEFT BYTE R1
      BLWP @VSBW        WRITE EDGE CHARACTER TO SCREEN
      INC  R0           RESET R0 TO ORIGINAL VALUE
      A    R4,R0        ADD NUMBER OF CHARACTERS TO ACCEPT
      BLWP @VSBW        WRITE AN EDGE CHARACTER TO SPOT BEYOND FIELD
      MOV  R0,@ENDOC    SAVE THIS LOCATION IN MEMORY
      S    R4,R0        RESET R0 TO ORIGINAL VALUE
      MOV  R4,@SAV4     STASH R4 IN MEMORY
CRSIOA BLWP @VSBR       READ THE CHARACTER POINTED TO BY R0
      MOVB R1,@ALTKEY   STASH THAT CHARACTER AT LOCATION ALTKEY
CRSIO BL   @CURFRC      FORCE THE CURSOR ONTO THE SCREEN
      BL   @KI2         USE THE SCANNING SUBROUTINE WITH FLASHING CURSOR
      CI   R8,9         HAS RIGHT ARROW BEEN STRUCK?
      JEQ  CRSRT        IF SO, JUMP
      CI   R8,8         HAS LEFT ARROW BEEN STRUCK?
      JEQ  CRSBK        IF SO, JUMP
      CI   R8,10        DOWN ARROW?
      JLT  CRSC4        IF LESS, JUMP
      CI   R8,15        HAS FUNCTION-9 BEEN STRUCK?
      JEQ  CRSDMY       IF SO, JUMP
      CI   R8,13        HAS ENTER KEY BEEN STRUCK?
      JLT  CRSDMY       IF LESS, JUMP
CRSC4 CI   R8,4         HAS FUNCTION-2 (INSERT) BEEN STRUCK?
      JNE  CRSENT       IF NOT, JUMP
      INC  @INSFLG      ELSE SET INSERT FLAG
      JMP  CRSIO        THEN JUMP BACK
CRSENT CB  @KEYVAL,@ENTERV HAS ENTER BEEN STRUCK?
      JEQ  CRSDMY       IF SO, JUMP
      CI   R8,3         HAS FUNCTION-1 (DELETE) BEEN STRUCK?
      JEQ  CRSDEL       IF SO, JUMP
      CI   R8,32        SPACE BAR
      JLT  CRSIO        IF LESS, JUMP
* THE FOLLOWING FIVE LINES ARE NEEDED ONLY IF ONE WANTS LOWER CASE
* CHARACTERS CONVERTED TO UPPER CASE.  IF NOT, OMIT THESE FIVE LINES
      CI   R8,122       COMPARE TO LOWER CASE Z
```

```
        JGT   CRSIO          IF GREATER, JUMP
        CI    R8,97          COMPARE TO LOWER CASE A
        JLT   CRSI1          IF LOWER, JUMP
        SB    @ANYKEY,@KEYVAL ELSE SUBTRACT >20 FROM KEYSTROKE
CRSI1
        MOV   @INSFLG,R1     TEST IF INSERT FLAG ON
        JEQ   CRSI1A         IF NOT, JUMP
        MOVB  @ALTKEY,R1     ELSE WRITE CURRENT CHARACTER
        BLWP  @VSBW          TO CURRENT SCREEN POSITION
        MOV   @ENDOC,R2      MOVE LIMIT ADDRESS INTO R2
        S     R0,R2          SUBTRACT CURRENT R0 POSITION
        LI    R1,TEMSTR      POINT TO TEMSTR LOCATION
        BLWP  @VMBR          READ CHARACTERS FROM SCREEN
        DEC   R2             DECREMENT CHARACTER COUNT
        JEQ   CRSI1A         IF R2 IS ZERO, NO INSERT - WE'RE AT LAST POSITION
        INC   R0             INCREMENT SCREEN POSITION
        BLWP  @VMBW          WRITE CHARACTERS BACK
        DEC   R0             POINT BACK ONE SPOT
CRSI1A  MOVB  @KEYVAL,R1     MOVE THE KEY STRUCK INTO LEFT BYTE R1
        BLWP  @VSBW          WRITE KEY VALUE TO SCREEN
        INC   R0             POINT AT NEXT CHARACTER POSITION
        BLWP  @VSBR          READ CHARACTER THAT'S THERE
        CB    R1,@EDGE       IS THIS AN EDGE CHARACTER?
        JNE   CRSIOA         IF NOT, JUMP
        DEC   R0             ELSE BACK UP ONE CHARACTER
        JMP   CRSIOA         THEN BACK FOR ANOTHER KEY INPUT
CRSRT   MOVB  @ALTKEY,R1     TAKE CURRENT SCREEN CHARACTER INTO LEFT BYTE R1
        BLWP  @VSBW          WRITE CHARACTER TO SCREEN
        CLR   @INSFLG        CLEAR THE INSERT FLAG
        INC   R0             MOVE TO NEXT SPOT
        BLWP  @VSBR          READ THE CHARACTER THERE
        CB    R1,@EDGE       IS THAT EDGE CHARACTER?
        JEQ   CRSRT1         IF SO, JUMP
        MOVB  R1,@ALTKEY     ELSE STASH CURRENT SCREEN CHARACTER
        BL    @CURFRC        FORCE CURSOR ONTO SCREEN
        BL    @KI2A          GO SCAN KEYBOARD
        CB    @KEYVAL,@RITEV IS RIGHT ARROW STILL HELD DOWN?
        JEQ   CRSRT          IF SO, KEEP GOING RIGHT
        CB    @KEYVAL,@NOKEY HAS NO KEY BEEN STRUCK?
        JEQ   CRSRT2         IF SO, JUMP
CRSRT1  DEC   R0             BACK TO PREVIOUS SPOT
CRSRT2  MOVB  @ONOFF,@KI2A+2 RESTORE DELAY CONSTANT
        MOVB  @ALTKEY,R1     GET CHARACTER INTO LEFT BYTE R1
        BLWP  @VSBW          WRITE TO SCREEN
        JMP   CRSIO          THEN JUMP BACK FOR ANOTHER KEY
CRSBK   MOVB  @ALTKEY,R1     GET CURRENT CHARACTER IN R1
        BLWP  @VSBW          WRITE TO SCREEN
        CLR   @INSFLG        CLEAR INSERT FLAG
        DEC   R0             BACK ONE SPOT
        BLWP  @VSBR          READ CHARACTER FROM SCREEN
        CB    R1,@EDGE       IS THAT EDGE CHARACTER?
        JEQ   CRSBK1         IF SO, JUMP
        MOVB  R1,@ALTKEY     ELSE STASH CHARACTER AT ALTKEY
        BL    @CURFRC        FORCE CURSOR ONTO SCREEN
        BL    @KI2A          GO GET KEYSTROKE
        CB    @KEYVAL,@LEFTV IS LEFT ARROW STILL HELD DOWN?
        JEQ   CRSBK          IF SO, GO BACK AGAIN
        CB    @KEYVAL,@NOKEY HAS NO KEY BEEN STRUCK
        JEQ   CRSRT2         IF SO, JUMP
```

```
CRSBK1 INC  RO          MOVE TO NEXT SPOT


       JMP  CRSRT2      THEN JUMP
CRSDMY JMP  CRSIX       THIS IS A DUMMY JUMP TO KEEP JUMPS IN RANGE
CRSDEL MOV  RO,R7       STASH RO IN R7
       CLR  @INSFLG     CLEAR INSERT FLAG, SINCE WE'RE DELETING
       MOV  @ENDOC,R2   END OF FIELD ADDRESS IN R2
       S    RO,R2       SUBTRACT CURRENT CHARACTER ADDRESS
       INC  RO          POINT TO NEXT CHARACTER
       DEC  R2          DECREMENT R2 COUNT
       JEQ  CRSD1       IF R2 ZERO, PRINT SPACE - WERE AT LAST POSITION
       LI   R1,TEMSTR   POINT R1 AT TEMSTR FOR TEMPORARY STORAGE
       BLWP @VMBR       READ CHARACTERS INTO LOCATION TEMSTR
       MOV  R7,RO       PUT BACK RO
       BLWP @VMBW       WRITE CHARACTERS FROM TEMSTR TO SCREEN
CRSD1  MOVB @ANYKEY,R1  PUT A SPACE IN LEFT BYTE R1
       MOV  @ENDOC,RO   GET LIMIT SPOT INTO RO
       DEC  RO          DECREMENT BY ONE
       BLWP @VSBW       WRITE A SPACE TO SPOT JUST BEFORE LIMIT
       MOV  R7,RO       GET RO BACK AGAIN
CRSDO  B    @CRSICA     BRANCH BACK TO BEGINNING
CRSIX  MOVB @ALTKEY,R1  WRITE CURRENT CHARACTER TO SCREEN
       BLWP @VSBW
       MOV  @ENDOC,RO   SET LIMIT POSITION IN RO
       DEC  RO          DECREMENT BY ONE
       MOV  @SAV4,R2    MOVE MAX NUMBER OF CHARACTERS INTO R2
CRSIX1 BLWP @VSBR       READ THE CHARACTER AT CURRENT RO POSITION
       CB   R1,@ANYKEY  IS THAT A SPACE?
       JNE  CRSIXX      IF NOT, WE'VE REACHED CONTENT OF STRING
       DEC  RO          ELSE MOVE BACK ONE SPOT
       DEC  R2          DECREASE CHARACTER COUNT BY ONE
       JGT  CRSIX1      IF GREATER THAN ZERO, JUMP BACK
CRSIXX MOV  @PGNUM,RO   GET ORIGINAL RO POSITION BACK
       SWPB R2          PUT CHARACTER COUNT IN LEFT BYTE R2
       MOVB R2,@TEMSTR  PLACE THAT AT TEMSTR
       SWPB R2          REVERSE R2 AGAIN
       JEQ  CRIX        IF R2=0, JUMP
       LI   R1,TEMSTR+1 ELSE SET R1 TO POINT TO STRING CONTENT STORAGE
CRSIX2 BLWP @VMBR       READ THE STRING FROM THE SCREEN
CRIX   B    @SUBRET     RETURN FROM THIS SUBROUTINE
*
* SUBRET IS SHOWN HERE FOR REFERENCE.  NORMALLY IT'S MADE A PART OF THE FIRST
* HIGH-LEVEL SUBROUTINE USED IN THE PROGRAM
*
SUBRET DECT R15
       MOV  *R15,R11
       RT
*
* THE FOLLOWING SUBROUTINE GETS KEYSTROKES FROM THE KEYBOARD WHILE ALTERNATING
* THE CURSOR WITH A CHARACTER STASHED AT ALTKEY
* THE LINES LIMI 2 AND LIMI 0 ALLOW THE SENSING OF FUNCTION-QUIT AND ALSO ALLO
* A BEEP VIA GPLLNK TO OPERATE PROPERLY
*
KI2    CLR  @STATUS     KEY-IN WITH ALTERNATING
       BLWP @KSCAN      CHARACTER AND CURSOR
       LIMI 2           ACTIVATE INTERRUPTS
       LIMI 0           SHUT OFF INTERRUPTS
       DEC  R4          ENTER AFTER R4 SET TO >0200
       JEQ  CHNG        AND R1 TO >1E00 AND VSBW
       CB   @ANYKEY,@STATUS HAS A KEY BEEN STRUCK?
```

```
        JNE   KI2             IF NOT, RE-SCAN KEYBOARD
        MOV   @KEYADR,R8      ELSE PUT KEY'S VALUE IN R8
        RT                    THEN RETURN
CHNG    CI    R1,>1E00        IS R1 SET TO CURSOR CHARACTER?
        JEQ   L1              IF SO, JUMP
        LI    R1,>1E00        ELSE SET LEFT BYTE R1 TO CURSOR
        BLWP  @VSBW           WRITE CURSOR TO SCREEN
        MOVB  @ONOFF,R4       PLACE TIMING IN LEFT BYTE R4
        JMP   KI2             GO BACK TO SCANNING KEYBOARD
L1      MOVB  @ALTKEY,R1      PLACE ALTERNATING CHARACTER IN LEFT BYTE R1
        MOVB  @ONOFF+1,R4     PLACE ALTERNATE DELAY IN LEFT BYTE R4
        BLWP  @VSBW           WRITE CHARACTER TO SCREEN
        JMP   KI2             GO BACK TO SCANNING KEYBOARD
*
* THE FOLLOWING IS A SPECIAL KEY INPUT FOR REPEATING OPERATION OF
* THE RIGHT AND LEFT ARROW KEYS
* THIS SUBROUTINE INCLUDES SELF-MODIFYING CODE
*
KI2A    LI    R5,>0280        LOAD R5 WITH DELAY FACTOR
KI2B    CLR   @STATUS         CLEAR GPL STATUS
        BLWP  @KSCAN          SCAN KEYBOARD
        CB    @KEYVAL,@NOKEY  HAS NO KEY BEEN STRUCK?
        JEQ   KI2C            IF SO, JUMP
        LIMI  2               SET INTERRUPTS ON
        LIMI  0               SET INTERRUPTS OFF
        DEC   R5              DECREMENT DELAY COUNTER
        JNE   KI2B            IF NOT ZERO, SCAN AGAIN
        MOVB  @ONE,@KI2A+2    ELSE MODIFY DELAY COUNT
KI2C    RT                    THEN RETURN
*
* THE FOLLOWING SUBROUTINE FORCES THE CURSOR CHARACTER ONTO THE SCREEN
*
CURFRC  LI    R1,>1E00        PUT CURSOR CHARACTER IN LEFT BYTE R1
        LI    R4,>0100        SET DELAY FACTOR IN R4
        BLWP  @VSBW           WRITE CURSOR TO SCREEN
        RT                    RETURN
*
* FOLLOWING SUBROUTINE CLEARS AN INPUT FIELD
* BEGINNING AT R0 POSITION, EXTENDING NUMBER OF CHARACTERS IN R4
*
CLRFLD
        MOV   R4,R2           PLACE VALUE OF R4 IN R2
        MOV   R0,R3           SAVE R0
        MOVB  @ANYKEY,R1      PUT SPACE CHARACTER IN LEFT BYTE OF R1
CLRFL1  BLWP  @VSBW           WRITE ONE SPACE IN FIELD
        INC   R0              POINT TO NEXT CHARACTER SPOT
        DEC   R2              DECREMENT COUNT OF SPACES
        JNE   CLRFL1          IF NOT ZERO, REPEAT WRITING OPERATION
        MOV   R3,R0           REPLACE ORIGINAL VALUE OF R0
        RT                    RETURN
*
* REQUIRED DATA SECTION
* THE FOLLOWING DATA SOURCE LINES ARE REQUIRED BY THESE SUBROUTINES
*
ONE     DATA 1
ENDOC   DATA 0
INSFLG  DATA 0
PGNUM   DATA 0
SAV4    DATA 0
```

```
ONOFF   DATA >0201
EDGE    BYTE >1F
ANYKEY  BYTE >20
NOKEY   BYTE >FF
ALTKEY  BYTE 0
ENTERV  BYTE 13
RITEV   BYTE 9
LEFTV   BYTE 8
TEMSTR  BSS  41
* THE NUMBER IN THIS BSS MUST BE ONE MORE THAN THE LARGEST STRING LENGTH
* EXPECTED IN THE PROGRAM'S EXECUTION
```

Y.U.V TO R.G.B CONVERSION.

I suppose the first question has to be why do people use Y,U,V colour components rather than the easier R,G,B components and how did this standard come about?

In the beginning there was black and white. The B.B.C. looked at what was created and said it was good! The decision to send the brightness or luminance of the scene was rushed into without any thought of future compatibility. This is the stem from where Y,U,V came from.

A black and white system sends the luminance of the scene along with some pulses called synchronisation. These pulses tell the T.V. screen or monitor when to start a new line and when to start at the top of the screen again.

Now. when the thought of colour came,the people in power decided against sending the brightness of the three primary colours. This was due to the fact that the amount of radio frequency space is limited and would take up three times as much room in the radio spectrum. There was another reason to not use this method and this was compatibility. Sending three pictures, one in blue. one in red and one in green, was incompatible with all black and white sets and sending the old black and white image would be incompatible with the new colour sets.

Problem, what do you do?

It was by luck that the old black and white cameras had no blocking effects on any colour. In fact they read the scene better than the human eye could. They summed up the total brightness of each colour from each position in the picture and by trial and error and one or two laboratory tests the equation for the luminance of any point in the scene was found to be:

$$Y = 0.3\,R + 0.59\,G + 0.11\,B$$

This is how well our human eye responds to the different colour frequencies. Red and blue are at each end of our vision so they are less powerful and green is sat in the centre so giving a stronger response.

We can now have three cameras reading the primary colours and we can reconstruct what the luminance signal should be. Therefore by using the primary colours and a bit of circuitry we have a signal which will be compatible with the old black and white system. Also on this Y signal will be the synchronisation pulses as before.

Y,U,V TO R,G,B CONVERSION.

One problem with the above is that it still does not give us colour if we require it. Thus two more signals were carefully created from the three primary colours so that they could be manipulated with the Y luminance signal to retrieve the three primary colours on a colour set or could be ignored on the black and white set. These other two signals are referred to as the U and V or I (in phase) and Q (quanrature phase). The equations are as follows:

$$I = U = 0.6\,R - 0.28\,G - 0.32\,B$$

$$Q = V = 0.21\,R - 0.52\,G + 0.31\,B$$

Now what happens is that the Y signal is sent as normal and the U and V signals are cleverly combined and sent at a different sub carrier and finally the sound is added. The whole lot of this is then modulated up to some daft frequency to allow the use of small aerials on roofs.

All T.V. sets contain circuitry to demodulate the signal then strip off the sound, U, V, and then Y components. The Y,U,V signals are mixed to produce the R,G,B signals. These are then amplified and sent to three electron guns at the back of the cathode ray tube. These guns are all the same and just emit a stream of electrons which is proportional to the current supplied to them. The colour is produced by phosphorous elements on the screen. The three guns are all slightly out of alignment which means when the magnetic coils try to focus on a point on the screen they will be fractionally out. This allows red, green, and blue phosphor to be in a mosaic on the back on the screen. One gun will hit all of the blue elements, one will hit all the red and so on.

This is why Texas Instruments used Y,U,V. The 4A computer was to be used on T.V. sets and Y,U,V needed to be sent just like with the three colour T.V. cameras. R,G,B would have been of no use and was very little thought about until computer monitors were specifically being made. Todays computer systems use R,G,B which goes directly into the monitor, gets amplified, and sent to the guns without extra Y,U,V to R,G,B conversion. The interesting thing is that you could take your 4A to any television transmitting station, plug in the Y,U,V signal into the modulator and hey presto the colour bar screen would be transmitted to millions of viewers around the U.K.

Down to business. To make the Y,U,V to R,G,B converter a few things need to be done. First of all the synchronisation signal needs to be taken out of the Y channel. This signal is then fed into the monitor being used. Secondly the Y,U,V components need to be mixed to make R,G,B. To find out how they are mixed, rearrange the equations above and solve in terms of R,G, and B.

Y.U.V TO R.G.B CONVERSION.

The new equations are:

$$B = 1.73\,Q - 1.10\,I + Y$$

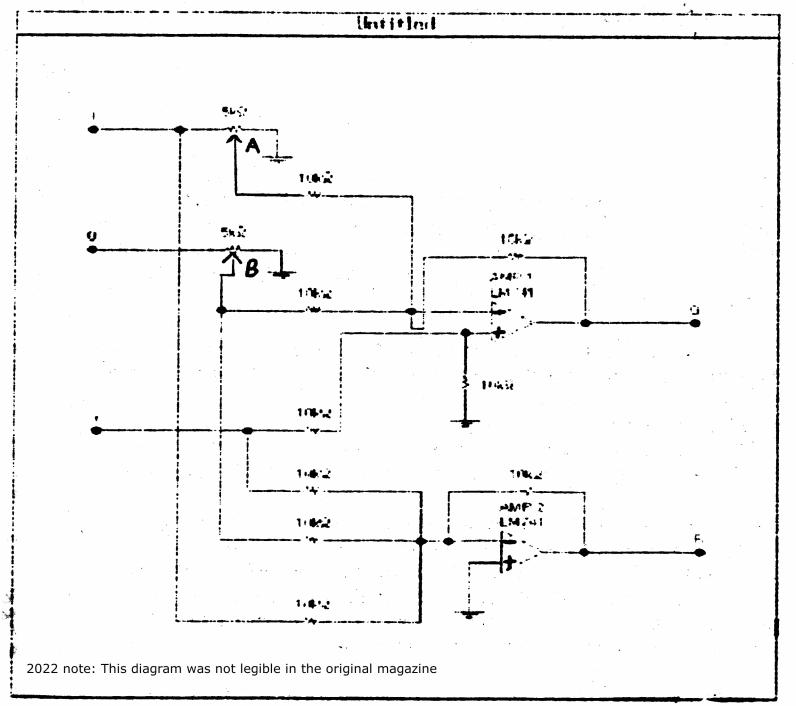
$$R = Y + 0.95\,I + 0.62\,Q$$

$$G = Y - 0.64\,Q - 0.28\,I$$

These equations can be tested by giving R,G, and B any value the first three equations and finding the corresponding values for Y,U, and V. Then place the Y,U, and V values into the three equations on this page and R,G, and B should appear as the values that you first placed into the equations on the other page.

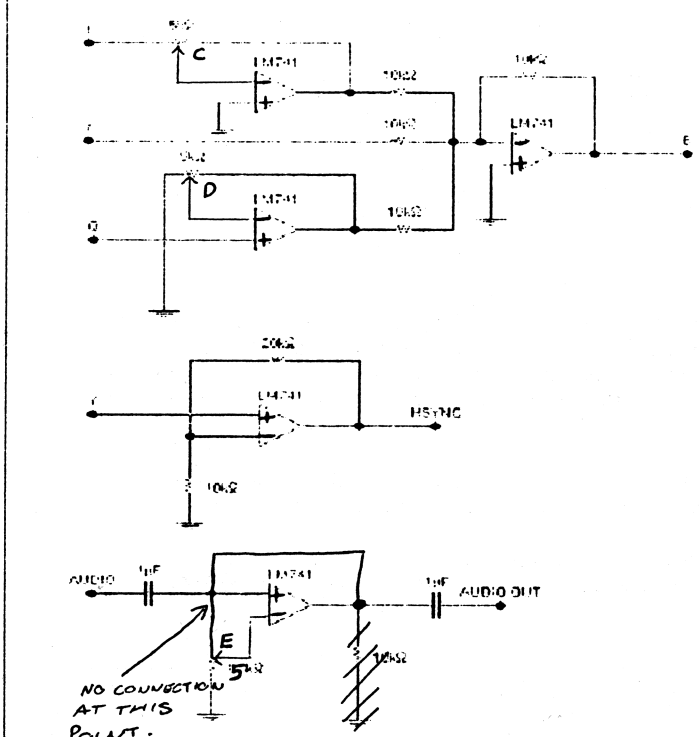
All that is needed now is some circuitry to whip of the synchronisation signal and some electronics to provide the formulae manipulation.

The following circuit is untested so I can not claim it works perfectly but it should cause no damage to the 4A or the monitor as the voltages are all very low.



2022 note: This diagram was not legible in the original magazine

2022 note: This diagram was illegible in the original magazine



With this circuit diagram I am taking the assumptions made by M.Shibuya in Mike Goddards book of hardware projects to be true. I don't have much time at the moment to test the circuit but the only problem I can foresee is that the synchronisation pulses are not fully stripped off the Y signal and therefore this may cause some colour jitter. However, this should be in a blanking period on the screen and did not seem to bother Mr. Shibuya.

Calibration can be done by eye or with a voltmeter. The by eye method will take into account any colour attenuation in the specific monitor so may look better than the exact calibration method.

To do a full calibration :

Place 1volt on I and adjust variable resistor A until 0.28 volts appear. (MEASURE VOLTAGE AT - INPUT OF OP AMP)

Place 1volt on Q and adjust variable resistor B until 0.64 volts appear.

This should have created the correct green and red signals. There are no variable resistors for the red signal. This was done to simplify the design and calibration. The I signal is 0.95 so we can call this 1 and the Q signal for the red is almost the same as for the green so we might aswell tap the voltage from the green setting. Using the same procedure, let C have 1.10 volts and let D have 1.73 volts. The negative signs from the equations are catered for by the opamps. (MEASURE C OR D VOLTAGE AT OUTPUT OF OP AMPS C OR D)

Finally note, the 12 volt power for the opamps is taken from the Texas video output socket pin 1. Ground is pin 6. Y is pin 2. I think the U or I is pin 3 and so Q or V is on pin 4. Also this circuit contains an amplifier to allow the use of a small speaker. The volume level is altered by E.

A Blast from the Past

I'll begin this by saying that there's no point in we members complaining about the lack of diverse articles in TI*MES unless we all start contributing and give the editor more of a choice. So here is my contribution. I'm not especially gifted in the area of writing, but I figure if I can do it, then anybody can.

I'll start by giving a brief potted history of my initially stormy relationship with the TI and why I abandoned it to my attic about 7 years ago...

I'm not sure what started it. It might have been a conversation on CIX. It might have been a sense of dissatisfaction with my PC. More likely it was Roy Robinson, but after six or seven long years in the wilderness I removed my mothballed TI from the attic, dug out all the bits and pieces I still had for it, checked the power supply and plugged it in for a serious dose of nostalgia for the time when there were more hobbyists than businessmen in the home computer market and we were all pioneers...

My TI99 was a Christmas present bought at just about the time when TI decided to drop the range of products. My friends had Spectri or Commodores and I had something that looked decidedly 70s with a really pathetic range of cassette games (and an incredibly expensive range of cartridge games). It was not a good start - my father had asked the IT manager at his work to recommend a home computer, and the TI99 had been the suggestion. For about a year I detested that man.

So, all my friends were enjoying the delights of Manic Miner (£5.95) while I struggled along with Adventure Mania and Hunt the Wumpus. I consoled myself with TI BASIC and set about writing my own games and utilities. It wasn't until the following Christmas that my affair with the TI began in earnest. I was given Extended BASIC. Suddenly, the balances tipped - here was I with 28 hardware sprites (with no attribute clash), three sound channels, one noise channel and a decent structured language. I was soon churning out software far superior to any that my friends could manage. Commodore's have never really had any kind of programming language and Sinclair BASIC was at best limited.

The big break came when an expansion box the size of a coffee table arrived complete with 32 massive kilobytes of RAM, a 90 kilobyte floppy disk, serial and parallel ports and a 9 pin printer. Now I was word processing, running all the Infocom games that non disk based systems could only dream of, using Pascal and so on. Great!

As time went by I succumbed to the charms of an Atari ST, but returned to the TI swiftly after. The Amiga was the final blow... do I spend £399 on an Amiga or on upgrading the TI? No competition really, especially since the Amiga had far better graphics and a much easier way of using them - Deluxe Paint II. The TI was shelved and gradually sold off piece by piece to fund university (I had to sell my Mini too :-( ).

The Amiga was later sold to make way for a mighty 486DX-33 (well, it was mighty when I bought it), and I began work as a Visual BASIC programmer. Funny how these things work out - started with BASIC, finished with BASIC. Two years down the line you find me here, still in Visual BASIC running on a 90 MHz Pentium with a lovely 17 inch screen displaying at 1152x852 in 24 bit colour. How fickle is the heart of a computer programmer.

But the TI was still with me, like the moth-eaten teddy bear you can't bear to throw out - it had survived three upgrades and was still with me. I hadn't lost it during four house moves, and all the parts were intact (minus the expansion box).

So one stormy night (when the wife was out), I unpacked it, assembled the components, and switched it on. Nothing. I hopefully unplugged Extended BASIC and the Speech Synthesiser. Not a sausage. I've since learned from such luminaries as Stephen Shaw that the years or disuse may have damaged the board, and Richard Twyning who was correct in saying that the heat sink on the vdp chip had melted but was sadly mistaken in asserting that a clean of the pins would cure the problem. TI was dead. By this point I was well and truly hooked again. I had to get Extended BASIC going and start typing in programs. In desperation I turned to my PC and loaded up the TI Emulator... bliss, a fully working TI99 (running considerably faster than normal) and my humble PC. With Windows I found I could have four or five TI99s all running at the same time! Spoiled for choice or what?

And that's where its at now. I'm using the emulator which works superbly (recommended to anyone with a 386 or higher) while I hunt about for a new system (console + peb with 32k, disk drives and pio card). Even the speech synth works!

So, having been out of the TI world for a good seven years or so, what does it all look like now to an outsider? Well, (and apologies to Richard Tywning in advance), there seems to be a hell of a lot of vapourware floating about... hell, that's no bad thing (unless its costing people anything) - living in the PC world, I'm well used to vapourware; take Windows 95 (aka Chicago, aka Windows 4) for example... or IBM's PowerPC. The list is endless. So that's ok - I don't mind the vapourware, its interesting to know what's being developed and what the TI can be coaxed into doing in its advanced state of years. A SCSI card sounds very interesting; it would be great if a SCSI hard disk could somehow be attached to it... hard disk prices are dropping all the time and imagine the BBS with 500mb of on-line storage accessing at hard disk speeds.

Yep, the TI needs a hard disk interface; that's for sure. IDE is probably out of the question, but SCSI... then you'd be talking.

The BBS also sounds like a great idea - with modem prices currently dropping through the floor (lets face it, TI programs are nowhere near as bloated as it modern contemporaries so modems capable of in excess of 9600baud are not a lot of use to somebody logging onto to the BBS) you can pick up 2400baud (240bps) modems for under £40 if you hunt around. I supposed you could also use your modem to connect to some of the on-line services like CIX. I wonder if there's any BBS software available that would support conferencing.... now there's a thought.

Yes, a BBS is an excellent idea. 99er's could download software to cassette or floppy (maybe even documents as well). One thing I'm not sure about is how its going to work. The server would need massive (in terms of 90k disks) on line storage, and I don't know how that could be achieved without fitting a hard disk to a TI. Or using a different computer.

That reminds me of something that does annoy me in TI*MES (doubtless along with many other members) - why is there so much berating of other computer systems? Are we so insecure that we feel the need to criticise and

slam other hardware designs? True, the ISA bus (and to a certain extent the EISA and VL bus) that forms the backbone of the IBM Pcs design (I'll not mention MCA) is not as sophisticated as the TI system but it has endured for 14 years and it still the most supported bus in the world. Suggest people are stupid for buying such machinery and you're alienating people. I know full well that the sub £2,000 computer I'm typing this on is infinitely better in every way to my TI99 (display, speed, software etc. etc.), but I'm not afraid to admit this. The TI is now, to me, more like as classic motor car. I drive a fast, quiet, comfortable, boring car all week to work, but at weekends I drive my Mini Cooper and enjoy myself. I hope that makes some sense...

I couldn't make the workshop (unfortunately) and so missed out on the demo. I hope all went well and Richard got the CD-ROM working. Maybe he'd consider donating it for use on the BBS? I'm kidding, honest!

I'll sign off now and carry on trying to make the emulator fall down... maybe some of those tricky CALL LOADs will do it.

If anyone wants to write to me, I'm at
8 Corfe Close
Southwater
Horsham
West Sussex
RH13 7XL

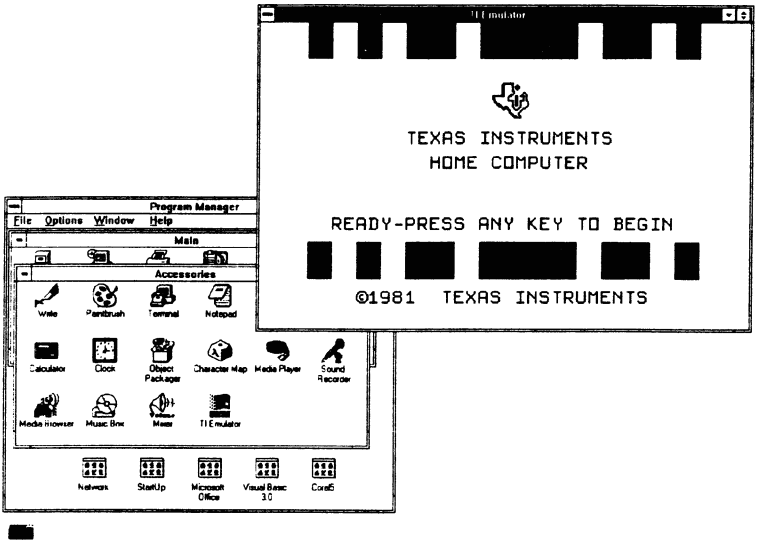
or (preferably) use one of the on-line services, my address at these are

CIX	:	SEG
Compuserve	:	100023,74
Internet Mail	:	SEG@CIX.COMPULINK.CO.UK

and I'll reply as soon as I can.

If anyone violently disagrees with something I've said then write to TI*MES and lets get some real discussion going!

And finally, something to cheer those of you up who are forced to use Windows at work...



Now that's my kind of desk top!

Richard Speed

2022 note: If the name Richard Speed rings a bell- for many years Richard regularly contributed articles to the tech news website theregister.com, often working in a reference to the TI99/4a. Richard retired around 2020, and as he was using the emulators of the TI on his PC, he donated his TI equipment to the UK User Group to sell for group funds.

28/10/94

K.F.HUGHES
220 BROADLAND DRIVE,
LAWRENCE WESTON
BRISTOL.AVON
BS11 OPN

Dear Gary,

Here is an article for the magazine on E-MAIL, I don't know if it is any interest to any of the members.But on past experience I have found that most members have more than one computer at hand,also they deal with COMMS like I do.So this might be of some interest to a few members.

TRADEMARK + COPYRIGHT
ON THE USE OF E-MAIL ADDRESSES

E-MAIL addresses everyone wants one nowadays but how to find a name or handle no-one else has got or who are using.To find out on the UNIX INTERNET it is easy to do,by typing, " whois< domain name>" = $whois nbc.com and within a few seconds you get your answer.

NATIONAL BROADCASTING COMPANY INC.
(address etc.)

When you have found out that the name you want to use,is not being used by anyone or is not registered.You inform the - INTERNET HEADQUARTERS that you want to register your name - handle and they inform you it is O.K. or not.
The people involved in registering are THE INTERNET NETWORK INFORMATION CENTRE,they assign domain names and rules of use. The agency receive over ONE THOUSAND requests for domain names a month and rising.The problem with names are that are TRADE-MARKS etc. are the problem of the requester.

Which as quoted in WIRED (OCT 94) A reporter requested to use the name MCDONALDS.COM .He was told that it was o.k. as the company of McDonalds hadn't requested that name to use on the network themselves.But he was told it is up to him if he wanted to use it,as there is no policy on the Net for domain names and their use.So if they are not registered they are fair game to all,so if you think of an usual name not used by anyone, it's up to you.How about TI*MES.COM - - - - ha,ha.

The problem with registering domain names that are not in use,like MCDONALDS etc. are breach of trademark etc. Though there are no written Law or Policy on domain E-MAIL names,one will have to be written soon.

Here is a sample from WIRED (oct 94) as an idea of the problem -: Adam Curry in June 93 was an MTV video jockey who registered his domain name as MTV.COM. with the Internet. Curry was an computer disc jockey using his mtv.com on the net to take messages on all subjects musical (Rock + Pop). MTV the company had no objections,also he paid for the site himself and all costs involved.Then in April 94 he resigned from MTV,he was promptly sued for copyright for using mtv.com.The lawyers of MTV fought on the TRADEMARK grounds.At the moment he is not using mtv.com,as he is fighting in the courts for

the Rights of the name.Millions of net users associate mtv.com with Curry not MTV the company. (big boy bullying tatics) will not succede I hope.Most people who apply for domain names are usually scared off ,from doing so with frets of court action, from the companies concerned,So do they have the right?
Any opinions will be appreciated on the question of E-MAIL addresses,write to me ,as above ,or E-MAIL ....
ken.hughes@walusoft.centron.com

Or ask the Editor to write a Question + Response Section ...

Another idea on the infringement on Trademarks + Copyright .. are CB handles etc. Who owns them ,for an example... rubber duck ..etc ,You know what I mean,they will get writs from Walt Disney next.Also the name TIMES ,WATCH OUT the news paper giants will be after you,knock-knock..

Here's a silly scenario who owns the rights to the alphabet or the English language,inventions of your mind,makes you think
Oh well I could go on - and -on but that's enough from me for now. Hi to all TI-er's and a MERRY XMAS and a HAPPY NEW YEAR to all...Idea's and suggetsions on the subject would make a good column in the magazine (hey ED) WHAT DO YOU THINK?

Thanx to all in the past and present
for help!

Ken Hughes

P.S, anymore Email users ,committee or you Ed.
or an Email list.What's the score on the BBS....

NOTE FROM THE ED.....

First of all I would like to thank everyone who has contributed to this issue of TI*MES. I will take this opportunity to remind you that the positions of Editor and Membership Secretary are open. If you are interested than please contact the Chairman.

Secondly, the "S&T" bulletin board has finally arrived and is in the hands of the Chairman. If you wish to know more about this facility or maybe give it a go then please contact the Chairman again.

TIPS FROM THE TIGERCUB
#35
Copyright 1986

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

The 4/86 Micropendium had a rather slow routine to count the number of words in a D/V text file. I think the following will be much faster. It ignores any lines beginning with a period (TI-Writer formatter commands), otherwise counts each cluster of characters followed by a space, plus the last cluster on the line.

```
10 !WORDCOUNT by Jim Peterso
n
100 DISPLAY AT(12,1)ERASE AL
L:"INPUT FILENAME? DSK" :: A
CCEPT AT(12,20):F$ :: OPEN #
1:"DSK"&F$,INPUT
110 A=1 :: LINPUT #1:M$ :: I
F ASC(M$)=46 THEN 130
120 X=POS(M$," ",A):: IF X=0
 THEN 130 :: IF X=A THEN A=X
+1 :: GOTO 120 ELSE F=1 :: C
=C+1 :: A=X+1 :: GOTO 120
130 C=C+F :: F=0 :: IF EOF(1
)<>1 THEN 110 :: CLOSE #1 ::
 DISPLAY AT(12,1)ERASE ALL:"
APPROXIMATELY "&STR$(C)&" WO
RDS"
```

==========

```
100 !TIGERCUB GRAPHPRINT by
Jim Peterson
110 !Will output to printer
a line graph of 31 items of
data, as for instance the
temperature for each day of
a month
120 !Values must be positive
 integers within a range of
75 from minimum to maximum
130 M$=RPT$("!_",65):: DIM T
$(31),D$(75):: MN=10000
140 DISPLAY AT(12,1)ERASE AL
L:"Input data - maximum 31":
"items. Enter to finish"
150 FOR X=1 TO 31 :: DISPLAY
 AT(14,1):X;TAB(4);CHR$(1)::
 ACCEPT AT(14,4)VALIDATE(DIG
IT)SIZE(-5)BEEP:T$(X):: IF T
$(X)=CHR$(1)THEN X=X-1 :: GO
TO 170
160 T=VAL(T$(X)):: MX=MAX(MX
,T):: MN=MIN(MN,T):: NEXT X
170 RN=MX-MN :: IF RN>75 THE
N PRINT "EXCEEDS MAXIMUM RAN
GE OF 75" :: STOP
180 IF MX>75 THEN AD=MX-75
190 OPEN #1:"PIO",VARIABLE 1
32 :: PRINT #1:CHR$(15);CHR$
(27);CHR$(51);CHR$(12):: PRI
NT #1:RPT$("_",132)
200 DISPLAY AT(12,1)ERASE AL
L:"Wait, please...": :".....
.this takes time"
210 LM=LEN(STR$(MX)):: FOR J
=1 TO 75 :: J$=STR$(76+AD-J)
220 IF J>66+AD THEN J$=J$&"
"
230 IF J/2=INT(J/2)THEN D$(J
)=RPT$(" ",LM)&SEG$(M$,1,132
-LM)ELSE D$(J)=J$&SEG$(M$,1,
132-LM)
240 NEXT J :: PRINT #1:RPT$(
" ",LM)&SEG$(M$,1,132-LM)
250 J=1 :: T=VAL(T$(J))-AD :
: T=76-T :: D$(T)=SEG$(D$(T)
,1,J*4+4)&CHR$(239)&SEG$(D$(
T),J*4+6,255):: J=J+1
260 T2=T :: T=VAL(T$(J))-AD
:: T=76-T :: FOR N=T2 TO T S
TEP (T2>T)+ABS(T>=T2):: D$(N
)=SEG$(D$(N),1,J*4+2)&CHR$(2
53+(T<T2))&SEG$(D$(N),J*4+4,
255):: NEXT N
270 J=J+1 :: D$(T)=SEG$(D$(T
),1,J*4)&CHR$(239)&SEG$(D$(T
),J*4+2,255):: IF J<=X THEN
260
280 FOR J=1 TO 75 :: PRINT #
1:D$(J):: NEXT J :: PRINT #1
290 T=8 :: FOR J=1 TO 31 ::
PRINT #1:TAB(T);STR$(J);:: T
=T+4 :: NEXT J
```

I still think of the TI as a HOME computer, and I still think that the home computer is an invaluable educational tool - but I guess not many folks agree with me. I had thought of writing full disks of a progressive series of lessons on one subject, but my present two full disks of math education have sold a combined total of 7 copies in 7 months, so that would obviously be a waste of time.

I had written this next

```
program for that purpose and
I guess it's no use wasting
it, so -
100 CALL CLEAR :: CALL TITLE
(5,"TAKE AWAY")!by Jim Peter
son
110 DISPLAY AT(3,10):"COPYRI
GHT":TAB(10);"TIGERCUB SOFTW
ARE":TAB(10);"FOR FREE":TAB(
12);" DISTRIBUTION":TAB(11);
"SALE PROHIBITED"
120 CALL PEEK(-28672,A@):: I
F A@=0 THEN 150
130 DATA FINE,NO,GOOD,UHOH,R
IGHT,TRY AGAIN,YES,THAT IS N
OT RIGHT
140 FOR J=1 TO 4 :: READ RIG
HT$(J),WRONG$(J):: NEXT J
150 FOR D=1 TO 1000 :: NEXT
D :: CALL DELSPRITE(ALL)
160 CALL CLEAR :: CALL CHAR(
95,"FFFF"):: CALL MAGNIFY(2)
:: RANDOMIZE :: CALL SCREEN(
14):: FOR SET=5 TO 8 :: CALL
 COLOR(SET,16,1):: NEXT SET
170 CALL CHAR(120,"E70042001
8007E0000E700420099423CE7004
20099423C00E7004218003C4200"
)
180 CALL CHAR(124,"0E0004010
0070800700020800OE01000")
190 DISPLAY AT(3,10):"TAKE A
WAY" :: CALL CHAMELEON
200 CALL COLOR(14,2,2):: CAL
L HCHAR(4,4,143,2):: CALL HC
HAR(5,4,143,2):: CALL SPRITE
(#25,120,11,25,25)
210 T=T+1 :: N=1-(T>5)-(T>15
):: G=10-(T>5)*80-(T>15)*810
 :: H=0-(T>5)*10-(T>15)*90
220 X=INT(G*RND+H):: Y=INT(G
*RND+H):: IF Y>X THEN TT=X :
: X=Y :: Y=TT
230 IF X=X2 OR Y=Y2 THEN 220
 :: X2=X :: Y2=Y :: Z=X-Y
240 GOSUB 250 :: GOTO 210
250 GOSUB 260 :: GOSUB 280 :
: GOSUB 310 :: FOR D=1 TO 20
0 :: NEXT D :: CALL DELSPRIT
E(ALL):: DISPLAY AT(18,1)::
CALL CHAMELEON :: CALL SPRIT
E(#25,120,11,25,25):: RETURN
260 FOR J=1 TO LEN(STR$(X)):
: :: A(J)=VAL(SEG$(STR$(X),J
,1)):: NEXT J :: FOR J=1 TO
LEN(STR$(Y)):: B(J)=VAL(SEG$
(STR$(Y),J,1)):: NEXT J
270 FOR J=1 TO LEN(STR$(Z)):
: C(J)=VAL(SEG$(STR$(Z),J,1)
):: NEXT J :: W=LEN(STR$(Z))
-LEN(STR$(X)):: RETURN
280 R=96 :: CC=96 :: FOR J=1
 TO N :: CALL SPRITE(#J,4@+A
(J),11,R,CC):: CC=CC+16 :: N
EXT J
290 R=116 :: CC=96 :: FOR J=
1 TO N :: CALL SPRITE(#4+J,4
8+B(J),11,R,CC):: CC=CC+16 :
: NEXT J
300 CALL HCHAR(18,12,95,N*3)
:: CC=CC-16 :: RETURN
310 R=140 :: FOR J=LEN(STR$(
Z))TO 1 STEP -1 :: IF LEN(ST
R$(X))=1 THEN M=CC :: GOTO 3
30
320 FOR M=CC TO CC+8 :: CALL
 LOCATE(#J-W,96,M,#J+4-W,116
,M):: NEXT M
330 IF A(J-W)>=B(J-W)THEN 36
0 :: CALL SPRITE(#28,49,16,9
6,M-9)
340 IF F3=1 THEN 360 :: F1=1
 :: A(J-W-1)=A(J-W-1)-1 :: I
F A(J-W-1)<0 THEN A(J-W-1)=9
 :: F2=1 :: A(J-W-2)=A(J-W-2
)-1
350 CALL SPRITE(#22,48+A(J-W
-1),16,80,M-24):: IF F2=1 TH
EN CALL SPRITE(#21,48+A(J-W-
2),16,80,M-40)
360 CALL SPRITE(#27,45,16,11
6,M-12)
370 CALL SPRITE(#20,63,11,R,
M)
380 CALL KEY(3,K,ST):: IF ST
<1 OR K<48 OR K>57 THEN CALL
 PATTERN(#20,32):: CALL PATT
ERN(#20,63):: GOTO 380
390 CALL DELSPRITE(#20,#28):
: CALL SPRITE(#12+J,K,11,R,M
)
400 IF K-48<>C(J)THEN GOSUB
450 :: CALL DELSPRITE(#12+J)
:: F3=1 :: GOTO 330
410 CALL DELSPRITE(#27):: IF
 F1=1 THEN 420 ELSE IF F2=1
THEN 430 ELSE 440
420 F1=0 :: CALL DELSPRITE(#
J-W-1):: FOR P=80 TO 96 :: C
ALL LOCATE(#22,P,M-24):: NEX
T P :: CALL SPRITE(#J-W-1,48
+A(J-W-1),16,96,M-24):: CALL
 DELSPRITE(#22):: GOTO 440
430 F2=0 :: CALL DELSPRITE(#
J-1-W):: FOR P=80 TO 96 :: C
ALL LOCATE(#21,P,M-24):: NEX
T P :: CALL SPRITE(#J-1-W,48
+A(J-1-W),16,96,M-24):: CALL
 DELSPRITE(#21)
440 CC=CC-16 :: NEXT J :: GO
```

```
SUB 480 :: F3=0 :: RETURN
450 DATA 123,124,125,123,124
,125,123,120
460 IF A@=0 THEN 470 :: CALL
 SAY(WRONG$(INT(RND*4+1)))
470 RESTORE 450 :: FOR JJ=1
TO 8 :: READ P :: CALL PATTE
RN(#25,P):: XX=2^250 :: NEXT
 JJ :: RETURN
480 DATA 121,122,121,122,121
,122
490 IF A@=0 THEN 500 :: CALL
 SAY(RIGHT$(INT(4*RND+1)))
500 RESTORE 480 :: FOR JJ=1
TO 6 :: READ P :: CALL PATTE
RN(#25,P):: XX=2^250 :: NEXT
 JJ :: RETURN
510 SUB CHAMELEON
520 M$="1800665AC342DB667E18
8100995AC3A5E78142BD24DB6600
81429924007E5AC3A53C241800FF
DB5AFF7EFF0099188100660018"
530 RANDOMIZE :: CALL CHAR(1
28,SEG$(M$,INT(43*RND+1)*2-1
,16)):: X=INT(14*RND+3)
540 Y=INT(14*RND+3):: IF Y=X
 THEN 540 :: CALL COLOR(13,X
,Y)
550 CALL HCHAR(1,2,128,30)::
 CALL HCHAR(24,2,128,30):: C
ALL VCHAR(1,31,128,96):: SUB
END
560 SUB TITLE(S,T$)
570 CALL SCREEN(S):: L=LEN(T
$):: CALL MAGNIFY(2)
580 FOR J=1 TO L :: CALL SPR
ITE(#J,ASC(SEG$(T$,J,1)),J+1
-(J+1=S)+(J+1=S+13)+(J>14)*1
3,J*(170/L),10+J*(200/L))::
NEXT J
590 SUBEND
```

==========

When you give your printer instructions, it remembers them until you turn it off. That is why you may find that your letter to Aunt Sally is being printed in double width underlined italics. The solution is found in another gobbledegook paragraph in the Gemini manual - "when (ESC "@") is sent to the printer, the conditions of the printer are initialized."

In plain English, `OPEN #1:"PIO" :: PRINT #1:CHR$(27);"@"` in your program or CTRL U, FCTN R, CTRL U, SHIFT 2 at the beginning of your TI-Writer text will cancel out any special orders the printer is still remembering and return it to its default conditions.

==========

Here's a bright idea by Scott King in the AVTI UG newsletter. When you load a program in order to modify it, put a reminder of its filename in the first line, such as `1 ! SAVE DSK1.NAME`. Then, when you are ready to save it, just list line 1, FCTN 8, use the space bar to erase the 1 !, and Enter.

==========

TIPS FROM THE TIGERCUB
#36

Some old business to take care of - Tom Wible (? - handwritten signature), in the MANNERS NEWSLETTER for April, points out that I am all wrong in my comments about updating a FIXED SEQUENTIAL file. There is no such thing as a fixed sequential or fixed relative file, only fixed files accessed sequentially or randomly (relative). Sequential and relative are access modes, not file attributes.

There is no reason to opena fixed file in anything other than RELATIVE mode, because if you do not specify the REC clause in your INPUT or PRINT, the computer defaults to sequential processing.

In one paragraph, that gentleman told me something about files I had'nt learned from the TI manuals and from the 2000+ newsletters on my shelf. File handling is apparently easy to understand for those who have had formal computer training, but it is a frustrating mystery to those of us who try to learn by hacking it. Won't somebody

please write a series of articles, somewhere, in plain, non-computerese English?

=======

And here is the last word on printing lines of more than 80 characters out of the TI-Writer Formatter, by W. Stewart Ash in a MANNERS newsletter of May-June 1986. It is only necessary to use the .FI command, and to set the right margin to the length you want, for example .FI;RM 120 for lines of 120 characters; and then use .TL or CTRL U commands to select a type font which will fit that many characters on a line (136 or 132 in condensed, depending on your printer; 96 in elite).

==========

Here's a new way to make music, for you music programmers and country music fans.

```
100 CALL CLEAR
110 PRINT "       WILDWOOD FL
OWER": : :"  on the hammered
 dulcimer": : : : : : : :"
          by Jim Peterson"
120 DIM S(26)
130 F=262
140 FOR N=1 TO 25
150 S(N)=INT(F*1.059463094^(
N-1))
160 NEXT N
170 READ N
180 C=S(N)
190 D=S(N)
200 CALL SOUND(-350,S(N),0)
210 RESTORE 350
220 FOR J=1 TO 63
230 GOSUB 260
240 NEXT J
250 GOTO 200
260 READ N
270 CALL SOUND(-350,S(N),0)
280 X=1^100
290 CALL SOUND(-350,S(N),0,C
,9)
300 X=1^100
310 CALL SOUND(-350,S(N),0,C
,9,D,19)
320 D=C
330 C=S(N)
340 RETURN
350 DATA 5,6,8,8,10,13,5,5,6
,5,3,3,5,3,1,1
360 DATA 5,6,8,8,10,13,5,5,6
,5,3,3,5,3,1,1
370 DATA 8,13,17,17,17,15,13
,13,8,8,10,10,13,10,8,8
380 DATA 1,1,1,3,5,5,8,5,3,3
,5,3,1,1,1
```

Lines 120-160 set up a scale of two octaves, beginning with the frequency in line 130 - to change the key, just change that frequency. Lines 170-190 set up the initial values, line 200 prevents a pause while data is being restored. Then the routine reads the data and plays the music.

Note the dummy calculation in lines 280 and 300, which does nothing but create a brief pause while the value of X is computed. This is a good method for a delay because it can be adjusted so exactly by changing the exponent, but use a value of 1 to avoid a numeric overflow.

Leave out lines 280 and 300 if you run it in Basic, but it is better in XBasic.

To write your own music by this method, just list the notes of a 2-octave scale from your starting frequency C C# D Ef E F F# G - etc. and number them 1 to 25.

Then, list the notes of your song by their number in the DATA statements. For a longer note, list it twice or more. Change the TO value in line 220 to your total number of notes, and RUN!

==========

```
100 CALL CLEAR :: ON WARNING
 NEXT :: CALL CHAR(128,"FF00
0000000000FF8181818181818181
8180808080808080FFFF8080808080
8081")
110 CALL CHAR(132,"810101010
10101FFFF0101010101018181000
00000000081"):: T=1 :: DIM K
$(15)
120 DISPLAY AT(3,7):"GORDIAN
 KNOT": : :TAB(12);"by Jim P
eterson"
130 DISPLAY AT(8,1):" Use ar
```

```
row keys to create a":"3-dim
ensional maze."
140 DISPLAY AT(11,1):" When
you cross your track,":"pres
s O to go over, U to go":"un
der, C to go across."
150 DISPLAY AT(15,1):" You m
ay at any time press":"Q to
clear the screen, or P":"to
save a manually created":"sc
reen."
160 T=1 :: DISPLAY AT(20,1):
"Choose - ":" (1) Manual":"
(2) Automatic":" (3) Retrace
":" (4) Load"
170 ACCEPT AT(20,11)VALIDATE
("1234")SIZE(1)BEEP:Q :: ON
Q GOTO 180,230,290,400
180 GOSUB 430
190 CALL KEY(3,K,ST):: IF ST
=0 THEN 190 ELSE D=POS("EDXS
QP",CHR$(K),1)+1 :: ON D GOT
O 190,210,210,210,210,200,36
0
200 CALL CLEAR :: GOTO 160
210 D=D-1 :: IF ABS(D-D2)=2
OR R+(D=1)=0 OR R-(D=3)=25 O
R C+(D=4)=2 OR C-(D=2)=31 TH
EN 190 :: GOSUB 490 :: IF D<
>D2 THEN GOSUB 440
220 GOSUB 480 :: GOSUB 500 :
: GOTO 190
230 GOSUB 430 :: RANDOMIZE
240 D=D+1+(D=4)*4 :: CALL KE
Y(0,K,ST):: IF ST=0 THEN 260
250 IF K=80 THEN 360 ELSE IF
 K=81 THEN CALL CLEAR :: GOT
O 160
260 T=INT(4*RND+2)*2-INT(2*R
ND):: FOR J=1 TO T :: IF D<>
D2 THEN GOSUB 440
270 GOSUB 480 :: CH=128-(D=1
)-(D=3):: CALL GCHAR(R,C,G):
: IF G<>32 THEN IF INT(2*RND
+1)<>1 THEN CH=G
280 GOSUB 510 :: NEXT J :: G
OTO 240
290 IF LEN(K$(1))=0 THEN DIS
PLAY AT(24,1):"CAN'T DO THAT
" :: GOTO 170
300 CALL CLEAR :: GOSUB 430
:: FOR J=1 TO T :: FOR JJ=1
TO LEN(K$(T)):: D=POS("EDXS"
,SEG$(K$(T),JJ,1),1)
310 IF D=0 THEN 350 :: IF D<
>D2 THEN GOSUB 440
320 GOSUB 480 :: CH=128-(D=1
)-(D=3):: CALL GCHAR(R,C,G):
: IF G=32 THEN GOSUB 510 ::
GOTO 350
330 K=ASC(SEG$(K$(T),JJ+1,1)
):: IF K<>67 AND K<>79 AND K
<>85 THEN JJ=JJ+1 :: GOTO 33
0
340 GOSUB 470 :: GOSUB 510
350 NEXT JJ :: NEXT J :: GOT
O 170
360 IF LEN(K$(1))>0 THEN 370
 :: DISPLAY AT(12,1)ERASE AL
L:"CAN'T DO THAT!" :: GOTO 1
60
370 DISPLAY AT(12,1)ERASE AL
L:"Save to - ":" (1)Cassette
":" (2)Disk" :: ACCEPT AT(12
,11)VALIDATE("12")SIZE(1):S
:: IF S=1 THEN OPEN #1:"CS1"
,INTERNAL,OUTPUT,FIXED 192 :
: GOTO 390
380 DISPLAY AT(16,1):"Filena
me DSK" :: ACCEPT AT(16,13):
F$ :: OPEN #1:"DSK"&F$,INTER
NAL,FIXED 192,OUTPUT
390 PRINT #1:T :: FOR J=1 TO
 T :: PRINT #1:K$(J):: K$(J)
="" :: NEXT J :: CLOSE #1 ::
 GOTO 160
400 DISPLAY AT(12,1)ERASE AL
L:"Load from -":" (1)Cassett
e":" (2)Disk" :: ACCEPT AT(1
2,13)VALIDATE("12")SIZE(1)BE
EP:L :: IF L=1 THEN OPEN #1:
"CS1",INTERNAL,FIXED 192,INP
UT :: GOTO 420
410 DISPLAY AT(16,1):"Filena
me? DSK" :: ACCEPT AT(16,14)
BEEP:F$ :: OPEN #1:"DSK"&F$,
INTERNAL,FIXED 192,INPUT
420 INPUT #1:T :: FOR J=1 TO
 T :: INPUT #1:K$(J):: NEXT
J :: CLOSE #1 :: GOTO 300
430 CALL CLEAR :: CALL COLOR
(13,5,11):: R,R2=12 :: C,C2=
14 :: D2=3 :: CH=129 :: CALL
 HCHAR(R2,C2,CH):: RETURN
440 CH2=128+((D2=1)*(D=2)*3)
+((D2=1)*(D=4)*5)+((D2=3)*(D
=2)*2)+((D2=3)*(D=4)*4)+((D2
=2)*(D=1)*4)+((D2=2)*(D=3)*5
)
450 CH2=CH2+((D2=4)*(D=1)*2)
+((D2=4)*(D=3)*3):: CALL HCH
AR(R2,C2,CH2):: RETURN
460 CALL KEY(3,K,ST):: IF ST
=0 THEN 460 ELSE IF POS("COU
",CHR$(K),1)=0 THEN 460
470 GOSUB 490 :: IF K=67 THE
N CH=134 :: RETURN ELSE IF K
=85 THEN CH=G :: RETURN ELSE
 RETURN
480 R=R+(D=1)-(D=3):: C=C+(D
```

```
=4)-(D=2):: RETURN
490 IF Q<>1 THEN RETURN ELSE
 K$(T)=K$(T)&CHR$(K):: IF LE
N(K$(T))<193 THEN RETURN ELS
E T=T+1 :: RETURN
500 CH=128-(D=1)-(D=3):: CAL
L GCHAR(R,C,G):: IF G>32 THE
N GOSUB 460
510 CALL HCHAR(R,C,CH):: R2=
R :: C2=C :: D2=D :: RETURN
```

==========

I think that educational programs should teach, not just test. This one makes up the kind of problems we all hated in school, but if you get the answer wrong it will show you how to work it. (see different version in Tips #45. - Ed.)

```
100 CALL CLEAR :: RANDOMIZE
110 DATA LUMBERJACK,CUT,CORD
S OF WOOD,BOY,PICK,QUARTS OF
 BERRIES,ELEPHANT,EAT,BALES
OF HAY,COW,GIVE,GALLONS OF M
ILK
120 FOR J=1 TO 4 :: FOR L=1
TO 3 :: READ M$(J,L):: NEXT
L :: NEXT J
130 A=INT(5*RND+2):: IF A=A2
 THEN 130 ELSE A2=A
140 B=INT(9*RND+2):: IF B=B2
 THEN 140 ELSE B2=B
150 C=INT(9*RND+2):: IF C=C2
 THEN 150 ELSE C2=C
155 X=B/C/A :: IF LEN(STR$(X
))>4 THEN 130
160 D=INT(4*RND+1):: IF D=D2
 THEN 160 ELSE D2=D
170 DISPLAY AT(3,1)ERASE ALL
:"IF";A;M$(D,1);"S CAN ";M$(
D,2):B;M$(D,3);" IN":C;"DAYS
,"
180 DISPLAY AT(6,1):"HOW MAN
Y ";M$(D,3);" CAN 1 ";M$(D,1
);" ";M$(D,2);" IN 1 DAY?"
190 ACCEPT AT(9,1)VALIDATE(N
UMERIC)BEEP:Q
200 IF Q<>X THEN 300 :: DISP
LAY AT(11,1):"CORRECT!"
210 DISPLAY AT(23,1):"PRESS
ANY KEY" :: CALL KEY(0,K,ST)
:: IF ST=0 THEN 210 ELSE 130
300 DISPLAY AT(11,1):"NO -":
"IF";A;M$(D,1);"S CAN ";M$(D
,2):B;M$(D,3);" IN":C;"DAYS,
"
310 DISPLAY AT(15,1):"THEN";
A;M$(D,1);"S CAN ";M$(D,2):B
;"/";C;M$(D,3);" IN 1 DAY":B
;"/";C;"=";B/C
320 DISPLAY AT(19,1):"SO 1 "
;M$(D,1);" CAN ";M$(D,2);B/C
;"/";A;M$(D,3);" IN 1 DAY":B
/C;"/";A;"=";X :: GOTO 210
```

==========

Here's a new way to put a title on the screen -

```
100 !SCATTERPRINT by Jim Pet
erson
110 CALL CLEAR :: M$="TIGERC
UB SOFTWARE" :: L=LEN(M$)::
IF L>28 THEN 110 :: C$=SEG$(
"ABCDEFGHIJKLMNOPQRSTUVWXYZ[
\",1,L)
120 FOR J=1 TO L :: RANDOMIZ
E :: X=INT(LEN(C$)*RND+1)::
Y=ASC(SEG$(C$,X,1))-64
130 DISPLAY AT(2,13-L/2+Y):S
EG$(M$,Y,1);:: C$=SEG$(C$,1,
X-1)&SEG$(C$,X+1,255):: NEXT
 J
140 GOTO 140
```

==========

This one is very basic, but if you have Terminal Emulator II, Speech Synthesizer, and a preschool child, it's a fine way to learn the alphabet, the keyboard, to spell his name, or just to have fun with - try a string of KK's for a train chugging uphill.

```
100 OPEN #1:"SPEECH",OUTPUT
110 CALL KEY(3,K,S)
120 INPUT M$
130 PRINT #1:M$
140 GOTO 120
```

Memory full - Jim P.

Dear TI'ers,

I haven't had time to produce an article, and I haven't been bothered after some comments have been made. Don't complain to me if you think TI*MES is getting thinner!

Some members seem to think that other articles get left out because I usually write alot, but we include what we've got. Stephen's articles disappeared from TI*MES because he sent them on disk, and Gary didn't receive all of the disks, and those he did receive he couldn't afford the time to sort them out, and not because of any "muting !" Stephen's input to the magazine is greatly valued, and we don't want it to disappear. Let's bring back **RAMBLES**. *

I couldn't afford any time to really do anything this quarter. I've been depressed over a couple of things, including my course.

My heart isn't in it anymore. All I want to do is win the lottery so I'm free to devote my life to TI programming! Very few people at The Nottingham Trent University care about true computing. We're forced into doing "business studies" and international studies! I'm the laughing stock of the rest of the students for owning an obsolete home computer, but they're all parasites, and I know that they're missing something that all of us TI'ers enjoy. To them, computing is just something they do during the day at University. It isn't an adventure, with new discoveries to be made. When the next assignment is out of the way, they can concentrate on their next party!!!

I sit in the computer room, eavesdropping on conversations about intel SX this, and DX that. Why don't TI have a TMS9900SX and TMS9900DX? That's because TI got it right first time, and why a GENEVE with a 1980 TMS9995 can give a 486 a run for its money.

Before I get carried away, I'll update you on what's happening. The Sandbach workshop on November 12th was very poorly supported, but great thanks to those who did attend. No one seemed interested in buying any T-Shirts, but they're still £10 each. I didn't take much to the show, just my video camera and box of electronic gear, which came in handy for Trevor to sort out a glitch with his 80-column card card, and for his monitor to be expertly set up correctly by Ross Bennett. It's all recorded for posterity on around 3 hours of video tape, along with Gary and myself meeting with Richard J. Simpson from Canada, during his visit to England in October, when he kindly smuggled three UCSD P-Code cards through customs for us!!!

There is no sign of the SCSI cards yet, but if anyone is in any doubts, I have recordings of my telephone conversations with Bud Mills, and I think he'll be getting another call from me very soon!

Today is the 8th of December, and I have so little time that I'm typing this into my organizer whilst sitting in the coffee bar in the university basement.

For reasons beyond my control, my CAD program wasn't approved for my project, so you won't see this until next Christmas. I still can't explain what they thought was wrong

* There were a number of editor changes, never advised, resulting in my contributions disappearing as wherever I sent them they never arrived at the editor of the day. No editor ever contacted me for copy or to ask for a specific format. As Gary is quoted above: "He couldn't spare the time to sort out Rambles". sjs (Rambles) 2022.

with the idea, but their answer was that CAD programs had been done before. Well, I haven't done one before!!!

CAD and Graphical User Interfaces are my main areas of interest at the moment, so imagine how it will look if I go for an interview at a CAD company and they ask what experience I have had with CAD software:

"Well, I was going to write a CAD program for my final year project, but my university wouldn't approve it!!"

I want to send special Christmas greetings to John Murphy, who recently broke his hip. He and other Dorset TI users made an excellent job of modifying my ScreenWriter program.

I promise that I won't neglect my duties as General Secretary too much, but my programming projects are on hold until my course has finished. University tends to get in the way of computing!

That's all for now.

Merry Christmas to all TI'ers, and also my apologies to Jim Cavanaugh for not sending him any disks. I'm too disorganized until I've got my SCSI card, and my Optical drive is on-line.

That's all from me until the spring issue.

Bluesman over and out.

CONSOLETATION ZONE

The following item was submitted by George Michel. As I do not know all the answers, and I though it would be of interest to most users, I decided to enter it into the Consoletation Zone for this quarter. If anyone can help then please drop me a line and I shall compile the answers for the next issue.....

The 640K of the "messy" DOS ! I am not clear how it does not affect the TI/99 architecture; Is it not a far cry from the 16K in our console? A lot of adverts for PCs talk about expandability to 4,8,16, 32 Megabytes. Software advertisers often call for some minimum megabyte requirement. Regarding the need for drivers for PC software, once they are installed, does it matter if they are there or not?

Copying of cartridges: There is nothing I would like better than to have TIXB on disk. The trouble I have with the games port has to be seen to be believed. Specially, if it will settle in my hard drive forever. Is CS a file in FW? My version 4.21 which was sent to me many moons ago by Jan Alexandersson of the Swedish Programbiten group does not have the CS file.How about a dream like this:

OLD WDS1.XB.LASVEGAS

from TI BASIC? with a call for XB IN LASVEGAS.

Music maker cartridge: TI said you need a TI thermal printer (which understands the fonts) to be able to print the score. Is it possible in this day and age to print scores on my old faithful RX80?

SCSI drive: Am I right in assuming that Bud Mills forms part of a team who have built or are building the card, Mike Maksimik provides the code for the EPROM and Tom Wilks is responsible for the software (equivalent to Myarc's MDM5) ? What will be the total cost to the user?

Technical manual: A guide to self repair of the TI 99/4A console (issue 45 page 42). Or is it the TI Technical data manual (issue 45 page 5)? Or is it Workshop manual and repair system? (issue 45 page 1); how can I get a copy?

As I am, shamefully, all thumbs, can anyone provide an RF modulator (issue 45 page 1) modified per article in issue 44 page 32 so I can see colours again on my CM-8833-II CGA monitor? I lost my colour TV to the kitchen some years ago.

Fortran: I do not understand what is meant by Line routine (issue 45 page 12) : I can not see a reference to it in LGMA'S 99 Fortran.

Is it possible for me to acquire a copy of the GPL manual (issue 45 page 14) or is it the GPL development package (issue 44 page 9)?

Fortran game: This was mentioned in issue 38 page 25. Is a copy of the source program available by now? That would be something!

I think this is enough for now.

Regards to all who hold the fort and take the trouble to keep us informed.

George Michel

T.I. EQUIPMENT FOR SALE!!!

1 x P.E. Box including 32K; RS232; Disk Controller; Dual double sided disk drives (excellent quality); printer cable; mains cable; all manuals; a comprehensive library of disk-based software (inc. the latest version of f.web; c99; games; adventures; data-base etc.)	£90.00 THE LOT!
1 x Speech synthesiser (Including booklet).	£5.00
1 x Cassette lead	£1.00
1 x Pair of Joysticks	£2.00
3 x TI 99/4a Consoles. All perfect & with power pack!	£10.00 each
Collection of cassette-based software (games etc.)	£5.00 THE LOT!

MODULES

1 x Extended Basic Module (Including Manual)	£5.00
1 x Mini Memory (Inc booklets & tape)	£3.00
1 x Terminal Emulator 2 (Make your T.I. say anything!)	£2.00
1 x Disk Manager 2	£2.00
1 x Household Budget Management (V.good for any cassette users)	£2.00
1 x Number Magic (Excellent early educational module)	£2.00
1 x Early Reading (Excellent!)	£2.00
1 x Early Learning Fun (Another good one!)	£2.00
1 x Addition & Subtraction 1 (Again, excellent.)	£2.00
1 x Demolition Division (Educational action game.)	£2.00
1 x Parsec (Including booklet)	£2.00
1 x T.I. Invaders	£2.00
1 x Indoor Soccer	£2.00
1 x The Attack (Including booklet)	£2.00
1 x Video Chess	£2.00
1 x Tombstone City	£2.00
3 x Munchman (I must like this game!)	£2.00 each
1 x Hunt The Wumpus	£2.00
1 x Adventure Module (Including tape)	£2.00

BOOKS

1 x GET MORE FROM THE TI99/4a (Garry Marshall)	£2.00
1 x THE TEXAS PROGRAM BOOK (Vince Apps)	£1.50
1 x TANTALISING GAMES FOR TI99/4a (H.Renko/S.Edwards)	£1.50
1 x USING & PROGRAMMING THE TI99/4a (Frederick Holtz)	£2.00
1 x INITIATION INTO THE TI99/4a ASSEMBLY LANGUAGE	£2.00

Assorted bundle of stuff inc.:- 1 x The TI*MES BOOK OF HARDWARE PROJECTS
2 x T.I. BASIC MANUALS
Issues 40 onwards of TI*MES.
Loads of official T.I. paperwork to help in programming etc. Included is a list of T.I. user groups world wide.

FREE to anyone interested!

If you're interested in any of the above please ring me, Scott, on:- **(0282) 459591.**

If you'd prefer to write, here's my address:-

Scott Whitley
19 Cunningham Grove
Burnley
Lancs. BB12 6DD.

POCKET CANON

This is a super little program that even the laziest of typists should be able to cope with! It runs in good old TI Basic and was written by a gentleman called S.T.Holl. It first (and probably last) appeared in an American publication called Home Computer Magazine in 1984. If it doesn't work, check for typing errors!

```
100 REM   POCKET CANON
200 DIM F(7)
300 DATA 0,0,262,196,220,165,175,131,175,196
400 READ N,V,F(0),F(1),F(2),F(3),F(4),F(5),F(6),F(7)
500 DEF M(X)=INT(X)-8*INT(X/8)
600 CALL SOUND(500,F(M(N/4)),3,1.5*F(M(N/2)),30+29*(V>31),2*F(M(N)),
    30+30*(V>63))
700 N=N+1+32*(N>31)
800 V=V+1
900 GOTO 600
```

.....and that's it! This program was unarchived and dusted off for your listening pleasure by **Mike Poskitt**.

FOR SALE!

OSCAR (Optical SCAnning Reader) for TI-99/4A, complete with lead to attach to console, and 8 programs in barcode format (no other hardware required). This rare collector's item from the USA is yours for **£15** incl.

Introduction to Assembly Language for the TI Home Computer
by Ralph Molesworth ..... **£6.00** incl.

International TI-Lines Vol.4, Issues 1-9..... **£3.50** incl.

TI User Magazine, Issues 1-6 (ie. all that were ever produced. Printed/edited by Galaxy of Maidstone, Kent. Remember them?)..... **£5.00** incl.

Phone Mike Poskitt on **0268-767107** (7.30-9.00pm only).

FCTN-QUIT

TI-99/4A FAULT FINDING

With the **TI-99/4a** now over ten years old it is inevitable that hardware problems will occur. Fortunately, members of the UK TI-Users Group have access to a repair facility. There are, however, some simple checks that the non-technical minded (myself included) can do, and which may obviate the need to send your computer to the repair shop. You will need a screwdriver and a multimeter:

1. Check the **fuse** in the main plug!
2. Check the power plug that connects to the console. Set the multimeter to AC range eg. 25V. The readings should be as follows: **Pins 1,2 16V~/1.6A**
   Pins 2,4 8V~/0.15A
3. If the readings are incorrect, open the PSU box and check the fuses.
4. If all is OK so far, open your TI console (go on, it's not such a big deal. Just watch out for static---discharge yourself by touching eg. radiator, and don't touch any of the components on the printed circuit boards!). In order to open the console you will need to unscrew 7 screws and unclip the on/off switch. Once open (see diagram below) you will see that the main board is encased in a metal cover. The position of the screws holding this and the keyboard and internal PSU is shown in the diagram.
5. Remove the PSU, then disconnect the fourway connector leading to the main computer board (the voltages may be marked alongside).
6. Check voltages with the multimeter (set to DC). The readings should be as follows:
   orange +5V **black ground** **white** +12V **green** -5V

   If the readings are incorrect, the fault is in the PSU.
7. If OK so far, well I'm afraid it's beyond the scope of this article! (ie. It's the limit of my knowledge). If there are any electronics wizards out there who can offer further help on the DIY front, please contribute an article to TI*MES.

Well, Gary is hoping for a page of this, to fill the inside back cover, so here goes! My news is bad because there are problems with the BBS. The BBS came on five disks, which were:

Disk 1 - BBS program disk

Disk 2 - BBS documents disk

Disk 3 - Murder Motel

Disk 4 - ANSI Tools

Disk 5 - BBS source code

The first problem was that the program disk and documents disk were DOUBLE DENSITY!!!! I had the task of pulling my system apart and putting my 4A back on line so I could read the disks with my MYARC HFDCC. I had to remove the GENEVE, MEMEX card, and TI Disk controller, and re-plug in my guitar strap, 32K card, and MYARC HFDCC. This gave me the problem of a 10 minute period of catch up learning as I worked out why my floppies didn't work! Optimistically, when I put my HFDCC away, I'd set my DIP switches on the HFDCC ready to take advantage of a new arrangement of drives, including my 1.44Meg 3.5" drive. Using my drives in their currently jumpered position meant they didn't work! Why? Well, the HFDCC allows you to alter the speed at which the card steps the drive head. I'd got them set to 8ms, so it wasn't giving my old 40 track drive time to move, so I had to pull everything apart again and change it to 16ms, giving the heads twice as long to catch up! This did the trick, but strangely, my 80 track drive wouldn't read the disk. On a MYARC controller, 80 track drives should be able to read and write 40 track disks because it detects it's a 40 track by reading the first sector, and then it causes the head to double step, but it didn't seem to do it! It would only read in the actual 40 track drive.

The program and document disks were formatted as double sided, double density, which gives 1440 sectors. I can't remember the exact size of the files on the program disk, but the documents disks contained about 900 sectors of D/V 80 files. It's a massive manual, which gives you some idea of the complexity of the program.

The only option at the time to convert the files was to save them onto 3.5" disks formatted to DSSD (720 sectors). This made no difference between the TI controller and MYARC HFDCC, because the HFDCC was setup (with the DIP switches) to make it look at the 3.5" floppy as though it was just a normal 5.25" 40 track drive. I knew I couldn't convert it to 40 track DSSD with the TEAC 80 track drive, because it gave unpredictable results while reading, and I couldn't risk using it for writing, or I'd have to swap the GENEVE over again later!!

Well, I managed to get away with archiving the files down from the 1440 sector disk to the 3.5" disk (720 sectors). The program disk archived down to just over 500 sectors, and the documents disk archived down to 421 sectors.

I was able to copy these to a standard 5.25" disk when I'd put the GENEVE back on-line. And it was a damn relief to get my system back together again. It was laborious, and very time consuming, and the problem is that I've got to go through it all again when the SCSI cards arrive, and in this case it will be much worse!!

Unfortunately when Trevor and I had a meeting to try out the BBS, we found that some of the files had become corrupted. This is either due to X-Raying in customs, or a problem when converting them. We do have the source code, and the documents describe enough of the program for us to have a go at getting things going.

Tim Tesch enclosed the source code for our Vice Chairman's eyes only, so given the documents and configuration files, Mark might be able to get something up and running to shorten the waiting time for another copy of the program arriving from the states.

What we can say about the program is that it appears to be XB based, but all the time-critical functions are LINKed assembly language routines, so there are XB assembly links for reading strings from the serial port, and file transfers!

The advantage of this is that you can make it do ANYTHING you want, and even chain programs together while a user is on-line, hence "Murder Motel"!! This is a multi-player game that can be played on-line whilst connected to the BBS!!! It's not the only game either! There are around ten games that can be added to the board, so callers can select which game they would like to play, and they can play different games whilst on-line!!!! It's not all games though! There is a program called GRAFFITTI WALL which lets users draw pictures whilst being on-line, using ANSI symbols found in TELCO, and also ANSI COLOR (yes on-line Color graphics) which is not found in TELCO. We are going to hopefully make this compatible with TE-II, so owners of the TE-II cartridge can finally make use of its long promised and almost legendary colour graphics support. GRAFFITTI WALL supports both 40 and

80 columns, so no one should feel left out.

Well, it's late, I'm tired, and I've got a doctors appointment in the morning, so I'm off!!!

Happy New Year

PS. SDT HAS WORKED ON TRUV. STEVENS SYSTEM SO THERE MUST HAVE BEEN A PROBLEM ON RICHARDS! (ED)

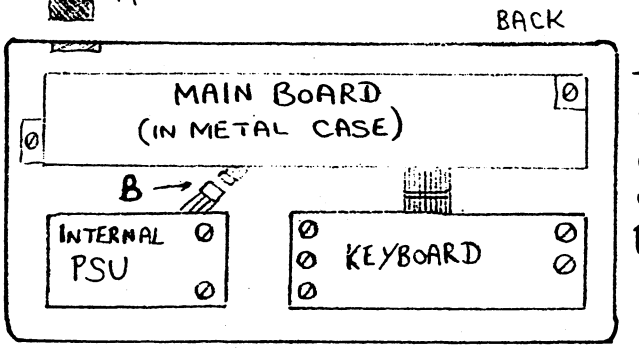
TI-99/4A FAULT FINDING

With the **TI-99/4a** now over ten years old it is inevitable that hardware problems will occur. Fortunately, members of the UK TI-Users Group have access to a repair facility. There are, however, some simple checks that the non-technical minded (myself included) can do, and which may obviate the need to send your computer to the repair shop. You will need a screwdriver and a multimeter:

1. Check the **fuse** in the main plug!
2. Check the power plug that connects to the console. Set the multimeter to AC range eg. 25V. The readings should be as follows: **Pins 1,2 16V~/1.6A**
   Pins 2,4 8V~/0.15A
3. If the readings are incorrect, open the PSU box and check the fuses.
4. If all is OK so far, open your TI console (go on, it's not such a big deal. Just watch out for static---discharge yourself by touching eg. radiator, and don't touch any of the components on the printed circuit boards!). In order to open the console you will need to unscrew 7 screws and unclip the on/off switch. Once open (see diagram below) you will see that the main board is encased in a metal cover. The position of the screws holding this and the keyboard and internal PSU is shown in the diagram.
5. Remove the PSU, then disconnect the fourway connector leading to the main computer board (the voltages may be marked alongside).
6. Check voltages with the multimeter (set to DC). The readings should be as follows:
   orange +5V **black ground** **white** +12V **green** -5V

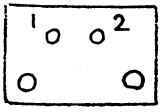
If the readings are incorrect, the fault is in the PSU.

7. If OK so far, well I'm afraid it's beyond the scope of this article! (ie. It's the limit of my knowledge). If there are any electronics wizards out there who can offer further help on the DIY front, please contribute an article to TI*MES.



TI-99/4A WITH OUTER CASING REMOVED.

A



B

+5	o	ORANGE
GR.	o	BLACK
+12	o	WHITE
-5	o	GREEN

Mike Poskitt 45

FOR SALE: HARDWARE, SOFTWARE, MANUALS and PERIPHERALS AS ONE LOT

HARDWARE: TI99/4A
EXPANSION BOX with
32K CARD (TI)
RS232 CARD (MYARC)
DISK CONTROLLER CARD (MYARC)
2 x MPI ½HEIGHT 5¼ inch DRIVES
UPRATED POWER SUPPLY
SPARE TRANSFORMER

PERIPHERALS:
JOYSTICKS (DOUBLE)
CASSETTE LEADS
SPEECH SYNTHESISER

MODULES: EXTENDED BASIC
TI WRITER
TI MULTIPLAN
EDOTOR/ASSEMBLER
MINI-MEMORY
TERMINAL EMULATOR II
HOUSEHOLD BUDGET MANAGEMENT
VIDEO CHESS
MUSIC MAKER
DIAGNOSTIC (very rare)

(all with handbooks or manuals)

CASSETTES: LINE-BY-LINE ASSEMBLER
PROGRAMMING AIDS I
TEACH YOURSELF BASIC
TEACH YOURSELF EXTENDED BASIC
PLUS 18 FURTHER CASSETTES WITH GAMES, UTILITIES AND MUSIC (FROM MAGAZINES ETC)
A FEW OTHER CASSETTES OF ODD BITS AND PIECES

) SOME CASSETTES AND DISKS HAVE
) P.D. PROGRAMS, SO ALL CASSETTES
) AND DISKS OTHER THAN COMMERCIAL
) ARE FREE OF CHARGE, BUT ARE
) NOT AVAILABLE SEPARATELY!
) SEVERAL BOOKS ALSO, HANDBOOKS
) AND 'HOW TO' BY PETER BROOKS
) AND GARRY MARSHALL - ALL IN,
) NOT SEPARATELY.

DISKS:
DISK MANAGER III (MYARC)
DISK MANAGER II (TEXAS)
DM 1000
TI WRITER
TI WRITER II
TI WRITER XB
TI WRITER 4.1
MULTIPLAN
MULTIPLAN ENHANCEMENTS
MULTIPLAN CONVERSION
MULTIPLAN V.4
FUNWEB 4.12
FUNWEB 40
FUNWEB 40 A/B/C
TI FORTH
TIFORTH SOURCE CODE
TI FORTH DISASSEMBLER (DASSM)
FORTH MODS
C 99 (1)
C 99 (2) A & B
C (IDEAS/STUFF)
EDITOR ASSEMBLER A & B
MIXED GAMES - VARIOUS SOURCES

TI DIAGNOSTIC
TI TESTS (VARIOUS METHODS)
ARCHIVER
TI MAILING LIST
SIDEWRITER
FASTERM
MODULE TO DISK
SORGAN
FRACTAL (more than one disk)
4TH TUTORIAL
PILOT 99 (prog. language)
PROGRAMMING AIDS II & III
FSKU4.12M
IBMTOTI
TITOIBM
PC-XFER
TE3C
LIBRARY DISK B
DM1000 DOCS
TE II MANUAL & PROTOCOL
TI MANUAL (2 DISKS)
MANY OTHER DISKS FROM LIBRARY

MANUALS: TI WRITER
EDITOR/ASSEMBLER
PRBASE
TE II
C99
ENHANCED BASIC

MULTIPLAN
TI FORTH
FASTERM
TE3C4
DM1000
TECH. MANUALS FOR ALL THE BITS

TI CASSETTE RECORDER/PLAYER EXTRA £12.50
CITIZEN 120 D PRINTER EXTRA £30.00

ALL THE ABOVE AS ONE LOT £215.00
(NO SPLITTING)

Dr. D. BAINES, 15, WINKFIELD ROAD, WINDSOR, BERKSHIRE, SL4 4BA.
BY POST ONLY PLEASE. ALL LETTERS ANSWERED. MUST BE COLLECTED