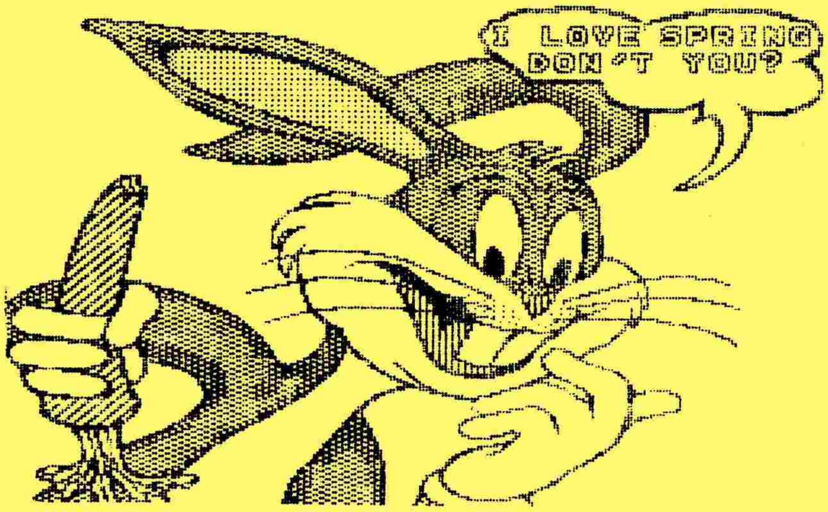
TI*MES

TI99/4A USERS GROUP (UK) CONTACTS



ISSUE NO. 40

SPRING 1993

CONTENTS:-

EDITORIAL

T.Stevens

Well another issue has arrived with you. We must apologise for the slightly late issue this quarter. Unfortunately Alan Baily our Editor has been very ill in Hospital. However with lots of rest and help and some good reading in this book he will most likely be back with us, doing the fantastic job that he does. It is no mean feat taking on and collating this little lot. I should know I have just completed this issue on Alans behalf. So from all of us Get Well Soon!!

AGM

At the rear of the book is this year map for the AGM. For full details see my artical.

DISCLAIMER

Views expressed by the contributors to this magazine are strictly their own, and do not necessarily represent those of the commitee. Contrary opinions are very welcome and will be given equal prominence if it is at all possible. Attributions not made at all, or made in error will be corrected on request.

NEXT COPY DATE

All submissions to the editor as usual. Copies to be recieved no later than 1st June 1993.

Please ensure the DARKEST print possible as light copies have to be darkened at a cost of 25p per copy so please help keep cost down. Those submitted this issue were near perfect. When producing the copies ensure width limit of 180mm and 66 lines maximum to the A4 sheet. If you wish to send in items hand written please ensure they are readable so the items can be typed up quickly. Thankyou for your support.

FROM THE CHAIRMANS CHAIR

T.STEVENS 1993

Last issue I had some problems with my trusty computer. The darn thing decided to blow up on me just before last issue, so I was stuggling with the internals to get it working. I ended up with a new ac/dc converter board inside and a tweek to a few of the other components. In the process I had few formatter hic ups which for some reason messed up the small routines in last months issue. So here are the amendments:-

```
100 CALL CLEAR :: CALL SCREE
N(13):: CALL MAGNIFY(2):: CA
LL SPRITE(#1,42,16,100,100)
110 CALL JOYST(1,X,Y):: CALL
MOTION(#1,-Y*4,X*4):: DISPL
AY AT(23,4):"Y=";Y,"  X=";X:
"-Y*4=";-Y*4,"X*4=";X*4 :: G
OTO 110
```

```
110 FOR N=1 TO 50 :: RANDOMI
ZE :: A=INT(RND*255):: B=INT
(RND*255):: CALL LOCATE(#1,A
/2+1,B+1):: NEXT N
120 FOR N=1 TO 50 :: RANDOMI
ZE :: CALL PEEK(-31808,A,B):
: CALL LOCATE(#1,A/2+1,B+1):
: NEXT N :: GOTO 110
```

So now you have the correct listings, you will now be able to take advantage of the extra power from your machine.

Now that I have got the "Bugs" out of the way, onto the A.G.M. This will be held 10am to 4pm on Saturday 1st May 1993 at the Princess Anne St Johns Training Centre, Trinity Street, Derby. This is due to the fact of price and availablity. Various venues around the country were looked at but came no where near to that of last years amenities and access with an addition of #20.00 off last years price. I do of course except that some members have to travel long distances to the AGM. Some in excess of 250 miles from the North and near on to the same distance to the South. So as far as a half way point Derby or any venue on its parallel was ideal. In reference to the letter from Mr R.G. Christian, I did infact look at two venues in Crewe and nearby Sandbach which is just off the M6 but prices were to say the least over the top. However if any one can come up with a venue in that area to beat that paid this year, with the same or better facilities I would be glad to hear from you, so that it can be considered next year. Any person wishing to get to Derby across country from the Welsh or Cheshire areas, I have traveled in recent months the A52 from Stoke on Trent, and it is now a very good road with lots of dual carrageway bar a few small spots. From the Stoke turn off (A500) to Derby. with average traffic it took me 1 hour 8mins at 60mph (Auto Route Plus) confirms.

I will now resume the little project of Extended Basic programming that I started last quarter. I hope this time the "Bugs do not appear. With some time to do some proof reading and not soldering I hope everthing will be OK. So here goes.

Last time we looked at Joystick and Randomize routines, this time we will look at the Coincidence of Sprites. TI EXTENDED does have a little routine for doing this called CALL COINC. This takes two Sprites, allows a tolerance and then returns a variable. If the variable is -1 then a coincidence, 0 if not. You can also do Sprites with Dot Row Dot Column positions with the same tolerance and numeric variable. Last but not least the CALL COINC ALL with numeric variable. Great but not very flexible is it. Now there is a formula that is very useful to the TI programer. Its called the 49 formula. This little baby is expressed as (TO-FROM)*.49 The TO and FROM being numeric variables.

To demonstrate the 49 Formula here is a little program thats puts a sprite on the screen at a random position and spots it by its placement variables then shoots another to hit it. As can be seen the whole program runs without CALL COINC. Try doing this with COINC and you will find it almost impossible to detect the sprite to a 100%, this program does!!!

```
100 CALL CLEAR :: CALL SCREE
N(2):: CALL CHAR(46,"0000001
818"):: CALL SPRITE(#2,94,16
,180,1,0,5)
110 FOR N=0 TO 25 :: RANDOMI
ZE :: CALL PEEK(-31808,Y,X):
: CALL SPRITE(#3,65+N,16,Y/2
+1,X+1):: CALL SOUND(-60,660
,8
120 CALL POSITION(#3,Y,X,#2,
R,C):: CALL SPRITE(#1,46,16,
R,C,(Y-R)*.49,(X-C)*.49):: C
ALL SOUND(476,-3,14)
130 CALL SOUND(120,110,6)::
CALL DELSPRITE(3!O:: CALL PA
TTERN(#3,35):: CALL SOUND(10
0,220,6):: NEXT N :: GOTO 11
0
```

Very neat is'nt it. I will now go through the program with you, however I will not explain the simple call Clears and screens etc. 100 sets up Sprite (#2) setting off a ^ symbol across the screen at a velocity speed of 5 from dot row 180 dot col 1. 110 then starts a for next loop and starts the Call Peek RND routine. (described last magazine). It then puts Sprite #3 at a random location with some sound, and cycles through the alphabet by adding to ASCII char 65 the loop value of N. 120 then gets the position of #3 and #2. Sprite #1 is then lauched with the Formula 49. with added sound. 130 then with call sound statements slows the routine for correct timing then loops to the next letter (For Next Loop remember). After the loop is done then it starts back at 110 and does it all again.
As you can see not a CALL COINC in sight. What you have been able to do is some very complex maths with the minimum of memory and with speed to boot. You can also use this 49 formula in chase the sprite type situations. Try this:

```
100 CALL CLEAR :: CALL SCREE
N(5):: CALL MAGNIFY(2):: CAL
L SPRITE(#1,77,16,100,100,#2
,71,16,30,30)
110 RANDOMIZE :: CALL PEEK(-
31808,Y,X):: CALL SOUND(60,1
000-Y*3,4):: CALL LOCATE(#1,
Y/2+1,X+1)
120 CALL POSITION(#2,R,C,#1,
Y,X):: CALL MOTION(#2,(Y-R)*
.49,(X-C)*.49):: CALL SOUND(
300,880-Y*3,15):: GOTO 110
```

As you can see the line 100 is the setup and starts Sprites #2 and #1. Line 110 does our friend the randomize adds a bit of sound then locates Sprite #1 with the row col variables from the RND peek with Y/2+1,X+1. Line 120 then finds the position of Sprite #2 and #1 then Formula 49 is then used in the Call Motion routine on Sprite #2, then a little sound with the value of Y altering the frequency. The sound call also acts as a delay. If you want to speed things up try the value -60 in the call sound statement in line 110. Then we jump back to 110 and do it all again.

You ask your self, how does one get to the Formula 49 in the first place. Well its simple when you know how. The .49 in fact is a limit control on your sprites. The highest legal velocity for any sprite is 125 (see your manual) You also know that sprites can go from 1 to 256 in both dot row and column how ever after dot row 193 your sprite is off your screen so you normally use 192-193 as a restraint. (see page 173 of Manual) So anything above that will bloop. So the formula that places one sprite, at a given location,in motion towards another location is (DOT ROW To, Minus DOT ROW From)*.49,(DOT COLUMN To Minus DOT COLUMN From)*.49, so if we look at this you can keep your sprites in a set area on your screen. Say you want to keep you sprites in the area dot row 17 through to say 185 and dot columns 17 through 185 then you can adjust the formula 49 to another value like this:- (185-17=168, 127/168=.7559, therefore .75*168=126.So the .49 now =.75. With this simple formula in the Call Motion statement the greater the distance the higher the velocity will be. Try adding this new formula to the above program to see it work.
Just to add a little fun to the program we will now put into the routine a joystick control.
Replace the following lines with these.

```
110 CALL JOYST(1,X,Y):: CALL
 MOTION(#1,4*-Y,4*X):: CALL
POSITION(#2,R,C,#1,Y,X):: CA
LL MOTION(#2,(Y-R)*.49,(X-C)
*.49):: CALL COINC(ALL,A)
120 IF A THEN CALL SOUND(-10
0,-2,8):: GOTO 110 ELSE CALL
 SOUND(-150,630-R-C,15):: GO
TO 110
```

As you note Call Coinc has been used but in conjuction with the Formula 49.

Just to finish off this quarters program hints and tips here is one for you to work out without me telling you how it works.

```
100 CALL SCREEN(2):: A=1 ::
FOR N=1 TO 28 :: CALL SPRITE
(#N,42,16,N*6,ABS(N+249*(A<1
)),-6,N*2*A):: A=-A :: NEXT
N
110 GOTO 110
```

NOW HERE IS A CHALLANGE TO NONE TI USERS

Can your IBM,AMIGA,ATARI ST, or whatever run a program and keep it going with a none loop as above. I think you will find TI RULES THE DAY!!!!! The reason is that a TI was designed for Sprites, that once set into motion carry on doing what they are supposed to do without any more program help. Did you know they were designed and invented by TI for the display of aircraft icons on Radar Screens. Your Texas therefore inherited this brilliant feature.

Just for Geneve owners, the Call Peeks in the above and last issue do not seem to work when run in GPL MODE. The Geneve appears to access a different memory location for the RND and possibly other Peeks too. If you have any updates for the Geneve I would be glad to hear from you.

I have spoken to our Editor Alan Baily and he is going to set up a letters page, if he gets any. So please write in to express your views and maybe we can make your club a better place. I do also wish to carry on our club for as long as possible, so new ideas such as maybe including another machine into the fold to boost members, I don't know whether it would be good thing or bad but, I leave it open for I hope strong debate. Its YOUR club so lets hear some sort of voice from out there in TI land.

A REMINDER

THE TEXAS USER GROUP AGM
1ST MAY 1993
ST JOHNS AMBULANCE TRAINING SCHOOL
TRINITY STREET
DERBY

At the show I will be showing off Pixpro. Not that new but I will be pulling up pictures like Macfiles converting them to Pix formatt, then converting to TI Artist messing about with them in TI-ARTIST Plus then converting them again into Page Pro. I will then be showing off how you can convert TI Artist Fonts to Page Pro, Plus lots more other goodies to boot. See you there!!!!!!

MARK
WILLS
3

```
%%%%  %%%%  % %    % %%%%% %      %
%   % %   % % %    % %     %      %
%   % %   % % %    % %     %      %
%   % %%%%  % %    % %%%   %      %
%   % %   % % %    % %     %      %
%   % %   % %  %  %  %     %
%%%%  %   % %   %%   %%%%% %%%%%  %  NO. 3
```

Mark Wills waffles on about things in the TI line...

Well, I'm back. Where have you been I here you ask... Well I suppose I'd better start at the beginning...

My absence initially was caused by four things : 1) The company I was working for began to get into financial trouble and thus I had to put in rather a ridiculous amount of time into trying to prop the business up as I had a vested interest in one of the products we were developing (a 24 bit graphics card for the Amiga using TMS34020's and four TMS34082's) but alas the business fell over and we were all out of a job.

2) I got a PC and spent rather an inordinate amount of time learning c which I like very much (more of that later), and as my girlfreind and I lived in a one room bedsit at the time there was only room for one computer to be set up and so the poor old TI had to be put away for a while.

3) My girlfreind (I'm very happy to say) fell pregnant and we are now expecting our first child any day - in fact it's now seven days over due (much knashing of nails and searching for my pass port!!) so, as I tend to write the material for this column over about a week, I may yet be able to report on the birth of our first son or daughter so read on!

4) In november we moved from our tiny bedsit onto a two bedroomed house and have spent until now decorating it in preperation for our new arrival.

Anyway, the decoratings nearly finished so Ive got my TI out again and now have some time to write some drivel...

As I said, Ive been learning c and very nice it is too. (By the way, it was nice to see a mention of c99 in Richards column last quarter) so I thought i'd start a c99 column this quarter, hopefully continuing each issue (depends on how much time I get to spend on the TI what with changing nappies etc). This issue, nothing particularly heavy, just covering the fundamentals of the c99 language and it's differences from other languages, why bother with C, its good points, its not so good points etc, and with a bit of luck it should develop into a tutorial type thing that should show begginners how to program in c99.

So here we go then!

c99 for the un-initiated!

What is c?

C is a lanquage developed in the sixties and seventies, its predecessors being A and B (no really, honest!), the whole purpose of which was to present to the programmer a UNIVERSAL computer lanquage that, instead of being interpreted, was compiled into the host machines native machine code lanquage for speed, whilst retaining relatively easy to understand program syntax etc. (as we progress you will notice that c99 programs look quite a lot like BASIC programs and can be read by humans quite easily and understood).
The key point though, is this: The c lanquage is absolutely universal. What I mean by this is that for example a c program to add two numbers on one make of computer is EXACTLY the same on another, regardless of the machines make, architecture and capability. Just to expand on this further, I could write a program on a IBM PC in c and run it quite happily, then send the SAME program (not modified in any way) over to the TI and run it on the TI with no modification!

Why c?

This is of course good news to the programmer who (if he's like me) is a totally lazy waste of space and doesn't want to have to write a program on one machine, only to have to start again from the beggining on another. This is why c quickly became popular for companies like Microsoft because they could write a program in c on one make of machine and run it on another make with often no extra work to do at all! And you get the extra bonus that your program will run at machine code speed!

A quick aside here... as all makes of computer are different and have different capabilities there are sometimes extra comands in c which one can utilize to take advantage of the machines specific capability. If you always stick to the standard commands that c offers however, and do not use any special machine specific libraries, your program will always run on another make of machine, often with no modification at all.

How is this done?

The fact that any standard c program will run on more than one machine is due to the fact that every machine has a compiler (or kernal) written for it in its own mother tongue (machine code) that tells that machine how to understand a c program and convert it into its own machine code. This is of course the same for all machines, and although the kernal is written in a different machine language for each different machine, they are all programmed to recognize the same c commands, thus any kernal can understand any c program.

What do I need to run c on my TI?
----

*2022 note- this assumes all versions of c are identical. They are not.

The very minimum you will need is this:

a) A disk system
b) 32k Memory expansion
c) c99 v2.1 or greater (preferably v4)
d) Editor assembler module + editor + assembler

It helps if you also have these:

a) Funnelweb v4.32 or greater to utilize the following:
   The program editor Enhanced assembler
   Script loader

Note: Funnelweb is a most exellent environment for writing c programs in, if you are lucky like me and have double sided disk drives, you can copy the editor, the assembler, the c compiler, the scriptloader, and all the c libraries and include files to one disk and you never have to change disks. My "funnelweb programmers disk" as I call it, contains fifty three files and I can edit, compile, assemble, and run all from DSK1 with my c source, assembler source, assembler object and script files on DSK2. without ever having to remove the Extended basic module. Funnelweb is available from Stevens disk library, as is the latest version of c99. c99 will cost you six quid if you send five disks (yes five!) plus a quid for postage.

Right, lets discuss a few things about c, keeping it very simple for now.

Variables:

As with a lot of languages, c has two types of variables : Global and Local. Simply put, a global variable can be accessed by any part of a c program and local variables can only be accessed by the function (thats c's name for a subroutine) in which it is declared. In c you have to delcare every variable that you use and its type before you use it.

Variable types:

In c99 there are a number of different types of variables: Integer, Floating point (a special case on the 4a version which we won't go into), Char (a character variable). ANSI c also supports Long, Double, and Void but as they arn't supported on the 4a we won't go into those.

Program Structure:

Before we can look at a c program we need to talk a little about how c programs are structured. Firstly, anyone used to basic will be surprised by the fact that c programs do not use line numbers. Different areas of a program are referred to by names that you give to them, and you can call up a different part of a c program (or a function to give it its proper term) using that name.
There is one part of a c program that is a special case, and it is called Main.
Every c program has a function called Main. Main is the starting point of every c program.

Lets have a look at a c program then. This program will put the words "Hello there world!" on the screen. I'll describe how to type it in and run it in a minute.

```
£include dsk1.stdio

main()
{
 message();
```

2022 note: although UK writers COULD use both £ and # in their documents, many didn't.
When you see £ in these pages -in a program- please replace it with a #
eg #include

```
}

message()
{
 puts("HELLO THERE WORLD");
}
```

How to type in and run

Note: ALL c PROGRAMS ARE WRITTEN IN LOWER CASE!! (The same goes for all c programs presented in this and future articles)
Type in the program using using either the editor in the editor assembler assembler, the program editor in fweb or the text edittor in fweb.

Save it using the file name DSKx.PROG;C where x is the drive number.
Note: if using the text editor in fweb the save using PF then as above for device name etc.
Run the c99 compiler as follows:

Using Editor Assembler:

Select the load and run option from the main menu
Type DSK1.c99C (assuming the compiler is on DSK1)
The compiler will then run.

Using Fweb:

Select the c COMPILER option from the menu.

In both cases:

The files c99C c99D and c99E contain the compiler.

When the compiler runs, you will be asked for the input file name. Type DSKx.PROG;C and when asked for the output file name, type DSKx.PROG;S
If all is well your disk drives should start chugging away and the compiler will run. If you get any errors while the compiler is running then you have made a typing error while typing the program in.

When then the compiler is finished, press n to cancel a re-run and the load the assembler.

When asked for the SOURCE name, type in DSKx.PROG;S and type DSKx.PROG;O for the object file name. The object file will contain the actual machine code that represents the c program you typed in.

Assuming no errors then we need to load the object file and run it:

Editor Assembler:

  Select option 3
  Type in DSKx.PROG;O
  Type in DSK1.CSUP (assuming the file CSUP is on drive 1)
  Press ENTER
  type START and press ENTER

Fweb:

```
  Select LOADERS from menu, select option 4.
  Type in DSKx.PROG;0
  Type in DSK1.CSUP
  Press ERASE (fctn 3)
  Press ENTER
```

You will see a screen full of names.
Look for the word START and move the cursor to it using the arrow keys.
If you can't see it, then press enter to bring the next screen of names up.
When the cursor is on the word START press PROCEED (fctn 6) and the program should start.

The file CSUP comes with the c compiler and contains various functions vital to the operation of c programs. Every compiled and assembled c rogram needs CSUP (on the 4a that is)

2022 note: as always with UK publications, replace the £ sign with a #. It wasn't necessary- but many writers never understood their printers.....

An Explanation of the Program:

£include dsk1.stdio

This line tells the compiler that we are going to be using some of the standard commands and functions in c (puts is an example) so the compiler makes sure it gets all the information needed to support them from the file stdio (which means standard i/o). Without this line, the program would not run. stdio is called "an include file".

main()

This defines the start point of our program. The two brackets indicate that no parameters are being passed to the function from any other point.

{

The open brace indicates that "all code from now on is part of the function called main()"

message();

This line calls the function called message.

}

The closed brace indicates the end of the main() function.

message()

Defines the start of the function called message. The two brackets mean that it won't take any parameters passed to it from the function that called it.

{

Again the opening brace, like main(), means "all code is for the

```
function called message()"
puts("HELLO THERE WORLD!");
```

Puts on the screen the string HELLO THERE WORLD! puts means PUT String. The semicolon indicates the end of that statement. Semicolons are used a lot in c and thier use will become more clear to you as we progress.

```
}
```

This indicates the end of the function called message() Control now returns back to the function called main() as main() was the calling function, at the line after the line that called message. Because the next line in main is a } which indicates the end of main() the program ends. As stated before, all programs start in main() and most end in main() as well.

Well that wasn't so bad was it. Lets expand on things a little more:

Variables:

Programs don't tend to be much use without the ability to alter things via the use of variables. In c variables are treated differently to variables in other languages such as BASIC, in that, as stated before you have to decalre them, and tell the compiler what sort of variable you are using.

Here is an example program, I'll describe it to you shortly.

```
001 £include dsk1.stdio
002
003 £asm
004  REF PRINTF
005 £endasm
006
007 main()
008 {
009  int c;
010  for(c=0;c<1001;c++) {
011   printf("%d ",c);
012  }
013 }
```

2022 note: Instead of £ use the # symbol- applies to all this article

You will notice that i've included line numbers so that we can refer to each line by number. This should also aid you when typing the programs into the editor or fweb.

Line 001: Tells the compiler neccessary information about c's standard functions, as before.

Line 003: Tells the compiler that all lines up to £endasm are to be treated as machine code and as such, be passed straight accross to the source file un-modified. This is handy for writing machine code functions in your c programs.

Line 004: Tells the assembler that tthe file PRINTF is also going to be loaded at run time (to support the printf instruction in line 011). If this line was ommitted the program would compile but it wouldn't assemble. Notice the leading space - it is important!

Line 007: This is the entry point of our program. All c programs start here

Line 008: "All code after this bracket is for main()"

Line 009: Tells the compiler that we are declaring an INTeger variable called c. Our machine is a 16 bit machine and therefore all integers have the range 0 to 65535 or -32768 to +32767.

Line 010: This is similar to BASICS FOR/NEXT command. It works like this:

The for command needs three peices of information to set it up:

The following should clarify things:

```
for(x;y;z)
{
 blah...
 blah...
 blah...
}
```

x = The initializer. We need to give some variable an intitial value for the start of the loop, in line 010 we are giving c an initial value of 0. If you want to use the value of a variable that is modified elsewhere in a program/function then miss out the initializer like this:

```
for( ;c<1001;c++)
```

y = The test. The loop needs to exit at some point so we need to include an expression to tell the loop when to pack its bags. In this case, we are testing c against all values less than 1001. After every iteration of the loop, the value of c will be tested against the value of 1001. If c is less than 1001 then the expression is said to evaluate to "true", and, in c when any expression in a loop test field is true, the loop will execute.
When c = 1001 then the test will evaluate to false (because 1001<1000=false) so the loop will terminate.

z = The control expression. After every iteration of a loop you want some sort of action to take place. In this case, we are incrementing c by a value of 1 each time the loop iterates (c++ means c=c+1 but you should use c++ when ever you can because it generates faster machine code.)

Notes:

The test and control expressions do not have to be associated with the variable referenced in the initializer field. You could have something completely different in them.
for example:

```
for(c=0;c<10;puts("hello"));
```

would put the word "hello" on the screen for ever. It caries on for every because each time the loop iterates, c is tested against the value of 10. Because c=0 the expression is true and so the control field executes!

In order to terminate the loop you would need to modify the value of c and one way of acheving this is as follows:

```
for(c=0;c<10;puts("hello"))
{
 c++;
}
```

This leads us on nicely to another point about loops:
If there are any peices code in curly brackets associated with the loop (like above) then that code will execute every time the test field equates to true. This allows entire routines to be enclosed inside a for expression like BASIC allows. The only confusing factor is that the formats are different. Perhaps an analagy will help clear matters as I suspect I have probably succesfully managed to confuse everyone!
These loops will do exactly the same in BASIC and c :

```
BASIC                     c
10 FOR I=1 TO 100         for(i=1;i<101;i++)
20 PRINT "HELLO"          {
30 NEXT I                  puts("HELLO\n");
                          }
```

Note: the backslash and n (\n) int the puts command tells the computer to move to the start of the next line for th next print statement. If it were absent it would print like this:

```
HELLOHELLOHELLOHELLO etc
```

the \n produces:

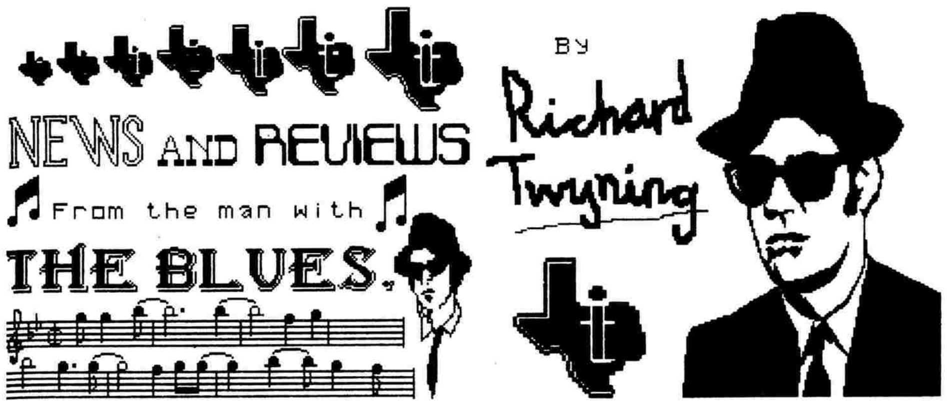
```
HELLO
HELLO
HELLO
etc
```

Well I think I'll leave it there for now, have fun with c and experiment. If you get stuck then write to me and send me your source code on disk and I will try and sort your problems out.

I'll quit now now: Next issue : DO WHILE and extended IF's (space permitting!)

Oh by the way - Amanda (my girlfreind) gave birth to a bouncing baby boy on 7th January 1993 at 4.35am weighing 8lb 9oz. I was there at the birth and I have to say that it is THE most amazing experience and totally changes your life. (Cut to the faint sounds of violins playing and Sir Harry Seacombe saying something profound and heavy!) I'm glad to say that the birth was pretty straight-forward and mother and baby (Benjamin Garry William) are both doing fine.

The date is the 14th of Feb and he is already smiling and cooing away at five weeks old, yet the book on baby development we have states they dunno what they're doing until 20 weeks so he must be one really clever dude (like father like son!! ahem!) I wonder if he'll appreciate the internal archetecture of the TI-99/4a? Hmmm... Only one way to find out....



Dear Fellow supporters of the cause,
Oops! Apologies first I'm afraid, for the errors that were in my last article!
These were caused by the exact mistake of the formatter that were mentioned in the exact same newsletter! This is due to putting numbers straight after asterisks without leaving a space.

Here are the corrections.

2022 : These are identical to the code in issue 39 except 30002 in the 16 col vn and line 200 in the ML vn where 6 becomes *256..

<u>256 colour XB & XHI version</u>

```
10 S=1.2 :: OPEN #1:"DSK1.MYART",INPUT ,DISPLAY ,FIXED 128
11 CALL LINK("MOD256"):: R=0 :: C=0
12 LINPUT #1:A$ :: A$=SEG$(A$,3,LEN(A$))
13 GOSUB 30000 :: COL=V :: GOSUB 30000 :: RL=V :: IF RL=0 THEN
   RL=256
14 CALL LINK("LINE",R/S,C/S,R/S,(C+RL)/S,COL):: C2=C :: C=C+RL
15 IF C>255 THEN R=R+1 :: C=(RL+C2)-256 :: CALL LINK("LINE",R/S,
   0,R/S,C/S,COL)
16 GOTO 13
17 CALL KEY(0,K,S):: IF S=0 THEN 17
18 CALL LINK("NORMAL"):: END
30000 IF A$="" THEN 30020
30001 V=ASC(A$):: A$=SEG$(A$,2,LEN(A$)):: RETURN
30020 IF EOF(1)THEN CLOSE #1 :: GOTO 17 ELSE LINPUT #1:A$ :: GOTO
      30001
```

<u>16 colour XB & XHI version</u>

```
10 S=1 :: INPUT "Enter MYART image name: ":N$ :: OPEN #1:N$,INPUT
   ,DISPLAY FIXED 128
11 LINPUT #1:A$ :: A$=SEG$(A$,3,LEN(A$))
12 FOR C=1 TO 16 :: R=ASC(A$):: G=ASC(SEG$(A$,2,1))::
   A$=SEG$(A$,3,LEN(A$)):: B=(R AND 7):: R=((R AND 240)/16)::
   G=(G AND 7):: CALL LINK("COLMIX",C,R,G,B):: NEXT C
111 CALL LINK("HIRES"):: R=0 :: C=0
113 GOSUB 30000 :: CALL KEY(0,K,ST):: IF K<>-1 THEN CALL
    LINK("NORMAL"):: END
114 CALL LINK("LINE",R/S,C/S,R/S,(C+RL)/S,COL):: C2=C :: C=C+RL
```

```
115 IF C>511 THEN R=R+1 :: C=(RL+C2)-512 :: CALL LINK("LINE",R/S
    ,0,R/S,C/S,COL)
116 GOTO 113
117 ! IMAGE LOADED
118 ACCEPT A$ :: CALL LINK("NORMAL") :: END
30000 IF A$="" THEN 30020
30001 COL=ASC(A$):: RL=ASC(SEG$(A$,2,1)):: A$=SEG$(A$,3,LEN(A$))
30002 RL=RL+((COL AND 15)* 256):: COL=((COL AND 240)/16)+1 ::
      RETURN
30020 IF EOF(1)THEN CLOSE #1 :: GOTO 117 ELSE LINPUT #1:A$ :: GOTO
      30001
```

Missing Link version

```
1 S=1 :: CALL LINK("PENHUE",2,16):: CALL SAY("ENTER+NAME"):: CALL
  LINK("INPUT",7 0,20,N$):: OPEN #1:N$,INPUT ,DISPLAY ,FIXED 128
  :: LINPUT #1:A$
2 I=ASC(SEG$(A$,2,1)):: A$=SEG$(A$,3,LEN(A$)):: IF I=250 OR I=122
  OR I=158 OR I= 255 THEN 100
12 R=0 :: C=0
13 GOSUB 30000 :: COL=V/16 :: GOSUB 30000 :: RL=V :: IF RL=0 THEN
   RL=256
14 CALL LINK("LINE",R/S,C,R/S,(C+RL),COL):: C2=C :: C=C+RL
15 IF C>255 THEN R=R+1 :: C=(RL+C2)-256 :: CALL LINK("LINE",R/S,
   0,R/S,C,COL)
16 CALL KEY(0,K,ST):: IF K<>-1 THEN CLOSE #1 :: END ELSE 13
17 CALL KEY(0,K,S):: IF S=0 THEN 17 ELSE END
100 A$=SEG$(A$,33,LEN(A$))! Remove palette data
110 R=0 :: C=0
120 GOSUB 180 :: CALL KEY(0,K,ST):: IF K<>-1 THEN CLOSE #1 :: END
130 CALL LINK("LINE",R/S,C/2,R/S,(C+RL)/2,COL):: C2=C :: C=C+RL
140 IF C>511 THEN R=R+1 :: C=(RL+C2)-512 :: CALL LINK("LINE",R/S,
    0,R/S,C/2,COL)
150 GOTO 120
160 ! IMAGE LOADED
170 CALL KEY(0,K,S):: IF S=0 THEN 170 ELSE END
180 IF A$="" THEN 210 190 COL=ASC(A$):: RL=ASC(SEG$(A$,2,1))::
    A$=SEG$(A$,3,LEN(A$))
200 RL=RL+((COL AND 15)* 256):: COL=((COL AND 240)/16)+1 :: RETURN
210 IF EOF(1)THEN CLOSE #1 :: GOTO 160 ELSE LINPUT #1:A$ ::
    GOTO 190
30000 IF A$="" THEN 30020
30001 V=ASC(A$):: A$=SEG$(A$,2,LEN(A$)):: RETURN
30020 IF EOF(1)THEN CLOSE #1 :: GOTO 17 ELSE LINPUT #1:A$ ::
      GOTO 30001
```

To disable the image type checking you can stick a GOTO in line 2 just before the IF statement. To force it to load the image as a 256 colour picture, then do a GOTO 12. To force it to load a 16 colour picture, do a GOTO 100.

I'm afraid that this article will be a bit shorter than my previous ones. I've been a bit depressed lately and have had to get eight assignments out of the way!

There's bad news from Alexander Hulpke. He has heard only rumours about the PC emulator for the 4A.

I've left a message on **CompuServe** for Ron Walters for him to give me any information on upgrading the address decoding on the RAVE 99 speech adapter, and I've also requested any information on his rumoured colour printing program that will print MYART images on a colour printer. I'm tired of having to use the Amiga to print out pictures I've drawn with YAPP.

I could never use Deluxe Paint to draw anything seriously. It's only used to manipulate pictures before transferring them to the GENEVE, or for printing YAPP pictures in full colour.

The Amiga to me is just a peripheral that's connected to port 2 of my RS232 and allows me to rip data off from PC's and Amiga's and get it onto my real computer.

To relieve my need to type a bit for this article, I have decided to include a program listing. This is produced using my 28 column printing program. It's for Commando for Extended BASIC, **but this is the totally upgraded** version to which I have **added two new units and a couple** more guards! The units I have **added are a tank and helicopter!**

<u>COMMANDO COMMANDS</u>

A - Move up	E - Fire Up
B - Move Left	F - Fire Left
C - Move Right	G - Fire Right
D - Move Down	H - Fire Down
I - Set Bomb	K - Change last move

<u>Helicopter ONLY Commands</u>

M - Take off   L - Land helicopter.

The indicator ALT indicates if the helicopter is currently on the ground, or in the air. 1 for flying and 2 for landed. When the helicopter is in the air it cannot fire or set bombs, but it also cannot be destroyed by a bomb, or shot by a guard!

The program needs a CALL FILES(1) and NEW typing in command mode before loading and running it.

I've noticed that my 28 column print program has spaced some line erroneously in a few places, but that'll add to the fun of typing it in. You will also need to create a high score by just opening a file of the correct type and printing a number to it. If you run the program on a GENEVE it will display the correct time in the top right corner of the screen! This line can be taken out totally on a normal 4A, or just changed to read a Zeno board clock etc.

I've had a bit of information from Gary Smith on the Texas Instruments TMS320C40 digital signal processor. It's been out since 1991 and chews on 250MIPS. The Intel 80586 **P5** has not been released yet, but when it is, it will only do 100MIPS! Two P5's in parallel will still only give you 190MIPS!

News from Norman (Simon Sloane, a friend of mine and Gary's, also from Newark) is that systems based on the P5 will set you back TEN THOUSAND POUNDS!!!! Tsunami (TI SuperSPARC) will start at $5000 and will hopefully totally kill it if they attract all the new software to it. The daft idea with the P5 is that it contains a 486 chip, and a RISC chip on the same bit of silicon which means that for any software to get any faster it will have to be totally re-written for RISC. All existing software will run on the 486 half of the chip.

Therefore, if the software's got to be re-written, then they might as well buy a Tsunami and start from scratch with a decent machine after IBM and Intel have totally wrecked innovation over the last decade.

Hopefully, there'll be more in the next article.

I hope to see you all at the AGM. Gary will be there and we'll hopefully have both our systems. I'll be demonstrating graphics and sound and whatever else comes to mind.

All for now from Richard Twyning.

```
1 CALL INIT :: CALL CLEAR ::
 PRINT "TI-Extended BASIC.":
" ":"EXPANDED VERSION  C
OMMANDO 4":"":"" :: CALL SAY
("COMMAND+O FOUR")
4 CALL SCREEN(3):: OPEN #1:"
DSK1.COMMANDO2",INTERNAL ::
INPUT #1:HSC :: CLOSE #1
 :: CALL SCREEN(3)
30 CALL CLEAR :: PRINT TAB(9
);"COMMANDO  I V": : : :: PR
INT "GRAPHICS BEING DEFI
NED.": : :: PRINT "  PLEASE
 WAIT.": : :
40 CALL CHAR(115,"0070FF78FC
FFAA7CBD4A89F99F9152BD"):: C
ALL CHAR(114,"8090BF9080
000000")
110 CALL CHAR(128,"081808080
8081C",129,"1C22020408103E")
120 CALL CHAR(130,"384404180
44438",131,"FF9999FFFFE7E7E7
")
130 CALL CHAR(136,"00101898F
E981810",137,"0030389CFF9C38
30")
140 CALL CHAR(138,"7878FCFFF
C787800",139,"FFBDDBE7E7DBBD
FF",125,"DB7E81FFAB7E7E9
9")
150 CALL CHAR(117,"010001000
10001AA",118,"FFFFFFFFFFFFFF
FF",119,"0000001818")
156 CALL COLOR(13,2,8,14,9,1
2,11,3,16,12,2,16)
190 CALL DELSPRITE(ALL):: PR
INT " IN THIS GAME YOU WILL
GIVE": :"THE ORDERS TO 5
 UNITS ";CHR$(128);CHR$(129)
;CHR$(130);CHR$(32);CHR$(116
);CHR$(115): :
191 BN,R,C,GN,X,W,N,BS,TS,D,
GC,K,A,T,S,M,P=0 :: W$="" ::
 PRINT "IN THEIR RAID ON
 AN ENEMY": :"INSTALLATION."
: :
200 PRINT "THERE ARE THREE D
IFFERENT": :"TARGETS -1 THE
FUEL DUMP": :"        -2
 THE TANK DEPOT ": : :: Z,BS
,UN,ALT,Q,GUARDS,SC=0 :: RAN
DOMIZE :: PRINT "
 -3 THE AIRPORT."
230 PRINT "1 HAS FEW GUARDS
3 HAS MOST.": :
240 INPUT "WHICH TARGET ,1 2
 OR 3 ?":TN
250 IF TN>3 THEN 240
260 CALL SAY("FINE"):: PRINT
 : :" YOU CAN GIVE YOUR UNIT
S ": :"ORDERS FOR UP TO
30 MOVES": :"AHEAD.": :
270 INPUT "HOW MANY MOVES AT
 A TIME ?   (1 TO 30) ? ":M
280 IF M>30 THEN 270 ELSE CA
LL SAY("O+K")
290 PRINT : :"KEY:": :" ";CH
R$(136);"  ";CHR$(137);"  PL
ANES": : :: PRINT " ";CH
R$(138);"   TANK",CHR$(139);
" FUEL DUMP": :" ";CHR$(131)
;"  BUILDING",
300 PRINT CHR$(125);"   GUAR
D": :" YOU SCORE BY BLOWING
UP ": :"PLANES,TANKS AND
 FUEL DUMPS": : :: PRINT " O
R BY SHOOTING GUARDS.": :
310 PRINT "   THE LOSS OF A
UNIT WILL": :" REDUCE YOUR S
CORE.": : :: PRINT " **
ONE MOMENT PLEASE **": : : :
: RESTORE 350
350 OPTION BASE 1
355 RANDOMIZE
360 DIM B$(18,32)
365 DIM B(70,3)
370 DIM G(150,3)
371 FOR R=1 TO 150 :: FOR C=
1 TO 3 :: G(R,C)=0 :: NEXT C
 :: NEXT R :: FOR R=1 TO
 70 :: FOR C=1 TO 3 :: B(R,C
)=0 :: BN=0 :: NEXT C :: NEX
T R
380 DIM U(5,2)
385 GN=0
390 DIM O(5,30)
395 UN=5
396 DISPLAY AT(24,1):"OLD GA
ME (Y/N):" :: ACCEPT AT(24,1
6)VALIDATE("YN")BEEP:A$
:: IF A$="Y" THEN 7500
400 FOR R=2 TO 17 :: FOR C=2
 TO 32 :: DISPLAY AT(24,1):R
;" ";C :: B$(R,C)=CHR$(1
17):: NEXT C :: NEXT R
401 B$(3,14),B$(4,13),B$(4,1
5),B$(5,14),B$(6,27),B$(7,26
),B$(7,28),B$(8,27)=CHR$
(118)
402 B$(8,8),B$(9,7),B$(9,9),
B$(10,8),B$(1,22),B$(2,21),B
$(2,23),B$(3,22)=CHR$(13
9)
403 B$(10,28),B$(11,27),B$(1
1,29),B$(12,28),B$(14,29),B$
(15,28),B$(15,30),B$(16,
29)=CHR$(139)
404 B$(14,6),B$(15,5),B$(15,
7),B$(16,6),B$(16,17),B$(15,
19),B$(16,21)=CHR$(118)
405 B$(15,18),B$(15,20),B$(1
5,23),B$(14,21),B$(14,22),B$
(14,23),B$(13,22),B$(13,
24),B$(12,23)=CHR$(139)
406 B$(12,12),B$(12,14),B$(1
3,11),B$(13,13),B$(13,15),B$
(14,12),B$(14,14)=CHR$(1
18)
407 B$(13,25),B$(13,26),B$(1
3,27),B$(14,24),B$(14,28),B$
(15,24),B$(16,25),B$(16,
26),B$(16,27)=CHR$(118)
408 B$(2,2),B$(2,4),B$(2,6),
B$(4,2),B$(4,4),B$(4,6),B$(6
,2),B$(6,4),B$(6,6)=CHR$
(139)
409 B$(2,29),B$(2,31),B$(4,2
9),B$(4,31),B$(6,29),B$(6,31
)=CHR$(139)
450 ON TN GOTO 460,570,680
460 FOR N=1 TO 40 :: DISPLAY
 AT(24,1):N :: C=INT(RND*29)
+3 :: R=INT(RND*16)+3 ::
 B$(R,C)=CHR$(131):: NEXT N
:: FOR R=2 TO 15 :: DISPLAY
AT(24,1):R :: B$(R,3)=CH
```

```
R$(139)
470 B$(R,7)=CHR$(139):: B$(R
,11)=CHR$(139):: B$(R,13)=CH
R$(139):: B$(R,15)=CHR$(
139):: B$(R,5)=CHR$(139):: B
$(R,9)=CHR$(139):: NEXT R ::
 GOTO 780
570 FOR N=1 TO 50 :: DISPLAY
 AT(24,1):3;N :: R=INT(RND*1
6)+2 :: C=INT(RND*29)+3
:: B$(R,C)=CHR$(131):: NEXT
N :: FOR N=1 TO 43 :: DISPLA
Y AT(24,1):2;N :: R=INT(
RND*15)+2 :: C=INT(RND*29)+3
580 B$(R,C)=CHR$(138):: NEXT
 N :: GOTO 780
680 FOR N=1 TO 73 :: DISPLAY
 AT(24,1):1;N :: R=INT(RND*1
6)+1 :: C=INT(RND*30)+2
:: B$(R,C)=CHR$(131):: X=136
-(RND>.5)
690 R=INT(RND*16)+1 :: C=INT
(RND*29)+3 :: IF R+C>45 THEN
 690
700 B$(R,C)=CHR$(X):: NEXT N
780 FOR R=1 TO 18 :: DISPLAY
 AT(24,1):18-R :: B$(R,2)=CH
R$(118):: B$(R,1)=CHR$(3
2):: B$(R,32)=CHR$(118):: NE
XT R :: FOR C=1 TO 32 :: B$(
1,C)=CHR$(118):: B$(18,C
)=CHR$(118)
851 CALL CLK :: NEXT C :: B$
(4,25),B$(5,24),B$(5,26),B$(
6,25),B$(9,26),B$(8,9),B
$(9,24)=CHR$(139)
852 B$(8,23),B$(10,25),B$(7,
24),B$(7,26),B$(8,4)=CHR$(13
8):: B$(8,27),B$(6,23),B
$(6,27)=CHR$(131)
860 FOR N=1 TO TN*50 :: DISP
LAY AT(24,1):(TN*50)-N;" GUA
RDS" :: GUARDS=N
870 R=INT(RND*10)+2 :: C=INT
(RND*29)+3 :: IF B$(R,C)<>CH
R$(117)THEN 870
880 B$(R,C)=CHR$(125):: G(N,
1)=R :: G(N,2)=C :: G(N,3)=I
NT(RND*4)+1 :: CALL CLK
```

```
:: NEXT N
950 C=INT(RND*8)+4 :: FOR N=
1 TO 5 :: DISPLAY AT(24,1):"
 UNIT ";N :: B$(17,C+N)=
CHR$(127+N):: U(N,1)=17 :: U
(N,2)=C+N :: NEXT N :: B$(U(
5,1),U(5,2))=CHR$(115)
960 R=INT(RND*10)+2 :: C=INT
(RND*29]+3 :: IF B$(R,C)<>CH
R$(117)THEN 960 ELSE U(4
,1)=R :: U(4,2)=C
990 INPUT "PRESS ENTER TO ST
ART GAME ":A$
1000 CALL SCREEN(12):: IF U(
4,1)=0 OR U(4,2)=0 THEN 1010
 ELSE B$(U(4,1),U(4,2))=
CHR$(116)
1010 CALL HCHAR(16,2,116,768
):: CALL VCHAR(1,1,32,32)::
CALL HCHAR(19,1,32,6*32)
:: DISPLAY AT(24,1)BEEP:"ALT
=";ALT
1015 DISPLAY AT(19,1)BEEP:"
HIGH SCORE=";HSC :: DISPLAY
AT(24,12):GUARDS;" GUARD
S."
1016 IF M=1 THEN DISPLAY AT(
20,18)BEEP:M;" MOVE " ELSE D
ISPLAY AT(20,18)BEEP:M;"
 MOVES "
1018 DISPLAY AT(19,1)BEEP:"S
C=";INT((TN*M*(TS*10)+(GN*5)
)/(6-UN));"  HI.SC=";HS
C
1025 FOR R=1 TO 18 :: CALL C
LK :: FOR C=1 TO 32 :: X=ASC
(B$(R,C)):: CALL HCHAR(R
,C,X):: NEXT C :: NEXT R ::
CALL VCHAR(1,1,118,18)
1070 CALL SAY("ARE+YOU+#READ
Y TO START#"):: GOSUB 30000
:: CALL MAGNIFY(2)
1080 @=0 :: FOR D=1 TO 10 ::
 CALL LOCATE(#D,D+@,D+@):: @
=@+14 :: CALL MOTION(#D,
8,8):: NEXT D :: CALL SOUND(
100,900,0)
1191 CALL SPRITE(#20,137,INT
(RND*9)+2,96,124,(INT(RND*10
```

```
)-INT(RND*10))+1,(INT(RN
D*10)-INT(RND*10))+1)
1200 CALL HCHAR(21,1,32,128-
32):: CALL HCHAR(24,1,32,2)
1201 W$="GAME LEVEL "&STR$((
TN-1)*6+M):: R=20 :: C=3 ::
GOSUB 6000
1210 FOR T=1 TO 5
1215 IF M=1 THEN DISPLAY AT(
20,18):M;" MOVE " ELSE DISPL
AY AT(20,18):M;" MOVES "
1220 IF U(T,1)=0 THEN 1430
1230 W$="ORDERS FOR UNIT "&S
TR$(T):: R=23 :: C=3 :: GOSU
B 6000
1245 DISPLAY AT(19,1):"SC=";
INT((TN*M*(TS*10)+(GN*5))/(6
-UN));"HI.SC=";HSC;"B=";
100-BN
1246 DISPLAY AT(24,1)SIZE(10
):"ALT=";ALT :: DISPLAY AT(2
4,12)SIZE(11):GUARDS;"GU
ARDS"
1270 CALL HCHAR(21,1,32,32):
: CALL HCHAR(21,2,127+T):: I
F T=4 THEN CALL HCHAR(21
,2,116)
1271 IF T=5 THEN CALL HCHAR(
21,2,115)
1275 DISPLAY AT(24,2)SIZE(10
)BEEP:"          "
1280 FOR N=1 TO M
1290 W$="MOVE "&STR$(N):: R=
24 :: C=5 :: GOSUB 6000
1330 CALL SOUND(500,500,1)
1340 CALL KEY(3,K,S):: CALL
CLK
1341 IF T=4 AND U(T,1)<>0 AN
D K=ASC("L")THEN CALL HCHAR(
21,2+N,K):: O(4,N)=K-64
:: GOTO 1420
1342 IF T=4 AND U(T,1)<>0 AN
D K=ASC("M")THEN CALL HCHAR(
21,2+N,K):: O(4,N)=K-64
:: GOTO 1420
1350 IF S=0 THEN 1340
1360 IF (K<65)*(K>7)THEN 133
0
1370 O(T,N)=K-64
```

```
1380 CALL HCHAR(21,2+N,K)
1390 IF K<75 THEN 1420
1400 N=N-1-(N=1)
1410 GOTO 1290
1420 NEXT N
1430 NEXT T
1440 FOR T=1 TO 5
1450 FOR P=300 TO 600 STEP 6
0
1460 CALL SOUND(40,P,1)
1470 NEXT P
1480 NEXT T
1485 FOR R=1 TO 5 :: FOR C=1
 TO M :: IF O(R,C)<1 THEN O(
R,C)=10
1486 NEXT C :: NEXT R
1487 CALL HCHAR(23,1,32,64):
: FOR R=1 TO 3 :: IF U(R,1)=
0 THEN 1489
1488 FOR C=1 TO M :: CALL HC
HAR(R+20,2,127+R):: CALL HCH
AR(R+20,C+2,O(R,C)+64)::
 NEXT C
1489 NEXT R :: IF M>14 THEN
1500
1490 CALL HCHAR(21,18,116)::
 CALL HCHAR(22,18,115):: FOR
 R=4 TO 5 :: IF U(R,1)=0
 THEN 1492
1491 FOR C=1 TO M :: CALL HC
HAR(R+17,C+18,O(R,C)+64):: N
EXT C
1492 NEXT R :: DISPLAY AT(23
,16):"COMMANDO 4"
1500 FOR N=1 TO M
1505 DISPLAY AT(19,1):" MOVE
 ";N;" SC=";INT((TN*M*(TS*10
)+(GN*5))/(6-UN));"B=";7
0-BN
1506 DISPLAY AT(24,1)SIZE(9)
:"ALT=";ALT :: DISPLAY AT(24
,12)SIZE(11):GUARDS;"GUA
RDS"
1510 FOR T=1 TO 5
1520 IF U(T,1)=0 THEN 1850
1555 IF O(T,N)+64=ASC("M")TH
EN 16000
1556 IF O(T,N)+64=ASC("L")TH
EN 15000
1557 IF ALT>0 AND T=4 AND O(
T,N)=9 THEN CALL SAY("YOU+CA
N+NOT+SET+THE1+TIME+WHEN
+A+L+T+IS+ONE"):: GOTO 1850
1558 IF ALT>0 AND T=4 AND O(
T,N)>4 AND O(T,N)<9 THEN CAL
L SAY("YOU+CAN+NOT+FIRE+
WHEN+A+L+T+IS+ONE"):: GOTO 1
850
1560 ON O(T,N)GOTO 1570,1590
,1610,1630,1670,1700,1730,17
60,1790,1850
1570 IF T=4 THEN 16570 ELSE
IF B$(U(T,1),U(T,2)-1)=CHR$(
117)THEN 1571 ELSE 1640
1571 B$(U(T,1),U(T,2))=CHR$(
117):: CALL HCHAR(U(T,1),U(T
,2),117):: U(T,2)=U(T,2)
-1
1572 IF T=5 THEN B$(U(T,1),U
(T,2))=CHR$(115):: CALL HCHA
R(U(T,1),U(T,2),115)ELSE
 B$(U(T,1),U(T,2))=CHR$(127+
T):: CALL HCHAR(U(T,1),U(T,2
),127+T)
1573 GOTO 1640
1590 IF T=4 THEN 16590 ELSE
IF B$(U(T,1),U(T,2)+1)=CHR$(
117)THEN 1591 ELSE 1640
1591 B$(U(T,1),U(T,2))=CHR$(
117):: CALL HCHAR(U(T,1),U(T
,2),117):: U(T,2)=U(T,2)
+1
1592 IF T=5 THEN B$(U(T,1),U
(T,2))=CHR$(115):: CALL HCHA
R(U(T,1),U(T,2),115)ELSE
 B$(U(T,1),U(T,2))=CHR$(127+
T):: CALL HCHAR(U(T,1),U(T,2
),127+T)
1593 GOTO 1640
1610 IF T=4 THEN 16610 ELSE
IF B$(U(T,1)-1,U(T,2))=CHR$(
117)THEN 1611 ELSE 1640
1611 B$(U(T,1),U(T,2))=CHR$(
117):: CALL HCHAR(U(T,1),U(T
,2),117):: U(T,1)=U(T,1)
-1
1612 IF T=5 THEN B$(U(T,1),U
(T,2))=CHR$(115):: CALL HCHA
R(U(T,1),U(T,2),115)ELSE
 B$(U(T,1),U(T,2))=CHR$(127+
T):: CALL HCHAR(U(T,1),U(T,2
),127+T)
1613 GOTO 1640
1630 IF T=4 THEN 16630 ELSE
IF B$(U(T,1)+1,U(T,2))=CHR$(
117)THEN 1631 ELSE 1640
1631 B$(U(T,1),U(T,2))=CHR$(
117):: CALL HCHAR(U(T,1),U(T
,2),117):: U(T,1)=U(T,1)
+1
1632 IF T=5 THEN B$(U(T,1),U
(T,2))=CHR$(115):: CALL HCHA
R(U(T,1),U(T,2),115)ELSE
 B$(U(T,1),U(T,2))=CHR$(127+
T):: CALL HCHAR(U(T,1),U(T,2
),127+T)
1640 IF U(4,1)=0 THEN 1850 E
LSE CALL HCHAR(U(4,1),U(4,2)
,116)
1660 GOTO 1850
1670 D=4
1680 GOSUB 3000
1690 GOTO 1850
1700 D=2
1710 GOSUB 3000
1720 GOTO 1850
1730 D=1
1740 GOSUB 3000
1750 GOTO 1850
1760 D=3
1770 GOSUB 3000
1780 GOTO 1850
1790 IF BN=70 THEN 1810
1800 GOSUB 3500
1805 GOTO 1850
1810 W$=" YOU HAVE RUN OUT O
F BOMBS" :: R=23 :: C=2 :: G
OSUB 6000
1845 CALL SAY("NO MORE LEFT"
)
1850 IF BS=0 THEN 1900
1860 GOSUB 4000
1900 NEXT T
1910 FOR T=1 TO TN*50 :: DIS
PLAY AT(24,7)SIZE(3):150-T
1920 IF G(T,1)=0 THEN 2300
1921 CALL SOUND(-4,110,10)
```

```
1930 FOR Z=1 TO 5 :: CALL CL
K
1935 IF Z=4 AND ALT>0 THEN 2
100 ELSE IF U(Z,1)=0 THEN 21
00
1940 IF RND>.7 THEN 2100
1950 IF G(T,2)<>U(Z,2)THEN 2
020
1970 IF ABS(G(T,1)-U(Z,1))>6
 THEN 2100
1980 D=1
1990 IF G(T,1)>U(Z,1)THEN 20
80
2000 D=3
2010 GOTO 2080
2020 IF G(T,1)<>U(Z,1)THEN 2
100
2040 IF ABS(G(T,2)-U(Z,2))>6
 THEN 2100
2050 D=4
2060 IF G(T,2)>U(Z,2)THEN 20
80
2070 D=2
2080 GOSUB 3030
2090 Z=5
2100 NEXT Z
2110 B$(G(T,1),G(T,2))=CHR$(
117)
2120 CALL HCHAR(G(T,1),G(T,2
),117)
2130 ON G(T,3)GOTO 2140,2170
,2200,2230
2140 IF B$(G(T,1)-1,G(T,2))<
>CHR$(117)THEN 2270 ELSE G(T
,1)=G(T,1)-1 :: GOTO 229
0
2170 IF B$(G(T,1),G(T,2)+1)<
>CHR$(117)THEN 2270 ELSE G(T
,2)=G(T,2)+1 :: GOTO 229
0
2200 IF B$(G(T,1)+1,G(T,2))<
>CHR$(117)THEN 2270 ELSE G(T
,1)=G(T,1)+1 :: GOTO 229
0
2230 IF B$(G(T,1),G(T,2)-1)<
>CHR$(117)THEN 2270 ELSE G(T
,2)=G(T,2)-1 :: GOTO 229
0
2270 G(T,3)=G(T,3)+1 :: G(T,
3)=G(T,3)+(4*(G(T,3)>4))
2290 B$(G(T,1),G(T,2))=CHR$(
125):: CALL HCHAR(G(T,1),G(T
,2),125)
2300 NEXT T :: DISPLAY AT(24
,9)SIZE(3):"   "
2310 NEXT N :: IF UN THEN 23
40 ELSE 5000
2340 W$="Q=QUIT. G=GO ON. S=
SAVE GAME." :: R=24 :: C=1 :
: GOSUB 6000 :: CNT=0
2345 IF U(4,1)<>0 THEN CALL
HCHAR(U(4,1),U(4,2),116)
2371 CALL SAY("WHICH ONE PLE
ASE")
2380 CALL KEY(3,K,S):: CALL
SPRITE(#27,114,8,137,16,#28,
114,16,137,INT((3000-CNT
)/12.5)+16):: CALL CLK
2390 IF K=81 THEN 5000
2391 IF K=ASC("M")THEN ACCEP
T AT(24,1):M :: GOTO 1200
2395 IF K=ASC("S")THEN E=0 :
: GOTO 7550
2396 CNT=CNT+1 :: DISPLAY AT
(21,11):" ";3000-CNT :: IF C
NT>3000 THEN E=1 :: GOTO
 7550
2400 IF K=71 THEN 1200
2410 GOTO 2380
3000 R=U(T,1)
3010 C=U(T,2)
3020 GOTO 3100
3030 R=G(T,1)
3040 C=G(T,2)
3100 FOR S=1 TO 6 :: R=R+(D=
1)-(D=3):: C=C+(D=4)-(D=2)::
 IF B$(R,C)=CHR$(117)THE
N 3300
3140 IF (B$(R,C)=CHR$(131))+
(B$(R,C)=CHR$(118))THEN 3350
3145 IF B$(R,C)=CHR$(115)OR
B$(R,C)=CHR$(116)THEN 3160
3150 IF (B$(R,C)<CHR$(128))+
(B$(R,C)>CHR$(130))THEN 3280
3160 FOR A=1 TO 5
3170 IF (U(A,1)=R)*(U(A,2)=C
)THEN 3190
3180 GOTO 3270
3190 IF A=4 AND ALT>0 THEN 3
270 ELSE U(A,1)=0
3200 FOR X=1 TO 5 :: CALL HC
HAR(R,C,127+A):: CALL SOUND(
50,200*X,1):: CALL HCHAR
(R,C,117):: CALL SOUND(50,11
0*X,1):: NEXT X :: CALL SAY(
"UHOH")
3255 B$(R,C)=CHR$(117):: UN=
UN-1
3265 IF UN=0 THEN 5000
3270 NEXT A
3280 IF B$(R,C)<>CHR$(125)TH
EN 3350
3290 GOSUB 4500
3295 GOTO 3350
3300 CALL HCHAR(R,C,119):: C
ALL SOUND(50,-1,1):: CALL SO
UND(20,-1,1):: CALL HCHA
R(R,C,117)
3340 GOTO 3400
3350 S=6
3400 NEXT S
3410 RETURN
3500 BN=BN+1 :: B(BN,1)=U(T,
1):: B(BN,2)=U(T,2)
3530 W$="CHARGE SET.TIMER ST
ARTED" :: R=23 :: C=8 :: GOS
UB 6000 :: CALL SAY("SET
 AND START TIME"):: CALL SOU
ND(500,760,1)
3580 BS=BS+1
3590 RETURN
4000 FOR Z=1 TO 70 :: CALL C
LK
4010 IF B(Z,1)=0 THEN 4410
4020 B(Z,3)=B(Z,3)+1
4021 DISPLAY AT(24,23):"TR="
;60-B(Z,3)
4030 IF B(Z,3)<60 THEN 4410
4050 FOR R=B(Z,1)-2 TO B(Z,1
)+2 :: FOR C=B(Z,2)-2 TO B(Z
,2)+2
4070 IF (R<2)+(R>17)THEN 436
0
4080 IF (C<2)+(C>31)THEN 436
0
4090 CALL GCHAR(R,C,GC):: CA
LL SCREEN(7):: CALL SOUND(30
,-1,1):: CALL HCHAR(R,C,
118):: B(Z,3)=0 :: CALL SOUN
D(30,-5,1):: CALL SCREEN(12)
:: CALL SCREEN(7)
4100 CALL HCHAR(R,C,32):: CA
LL SOUND(30,-8,1):: CALL HCH
AR(R,C,117):: CALL SCREE
N(12)
4160 IF GC<>125 THEN 4240
4170 FOR W=1 TO TN*50 :: CAL
L CLK
4180 IF (G(W,1)=R)*(G(W,2)=C
)THEN 4200
4190 GOTO 4230
4200 G(W,1)=0
4210 GN=GN+1 :: CALL SAY("GO
T+ONE"):: GUARDS=GUARDS-1 ::
 DISPLAY AT(24,12)SIZE(1
1):GUARDS;"GUARDS"
4220 B$(R,C)=CHR$(117)
4230 NEXT W
4240 IF (GC>135)*(GC<140)THE
N 4260
4250 GOTO 4275
4260 B$(R,C)=CHR$(117)
4270 TS=TS+1
4275 IF GC=115 OR GC=116 THE
N 4290
4280 IF (GC<128)+(GC>130)THE
N 4350
4290 FOR W=1 TO 5
4300 IF (U(W,1)=R)*(U(W,2)=C
)THEN 4320
4310 GOTO 4340
4320 IF W=4 AND ALT>0 THEN 4
335 ELSE U(W,1)=0 :: CALL SA
Y("UHOH")
4330 UN=UN-1
4335 IF UN=0 THEN 5000
4340 NEXT W
4350 B$(R,C)=CHR$(117)
4360 NEXT C
4370 NEXT R
4390 BS=BS-1
4400 B(Z,1)=0
4410 NEXT Z
4420 RETURN
4500 FOR A=1 TO TN*50
4510 IF (G(A,1)=R)*(G(A,2)=C
```

```
)THEN 4530
4520 GOTO 4620
4530 G(A,1)=0
4540 FOR X=1 TO 6 :: CALL HC
HAR(R,C,125):: CALL SOUND(50
,-X,1):: CALL HCHAR(R,C,
117):: CALL SOUND(50,110*X,1
):: NEXT X
4600 GN=GN+1 :: CALL SAY("GO
T+ONE"):: GUARDS=GUARDS-1 ::
 DISPLAY AT(24,12)SIZE(1
1):GUARDS;"GUARDS"
4610 B$(R,C)=CHR$(117)
4620 NEXT A
4630 RETURN
5000 CALL SAY("THE1+GAMES+FI
NISHED+NOW"):: IF UN THEN 50
40
5010 CALL SOUND(1000,220,1)
5020 CALL SOUND(1000,110,1)
5030 CALL SAY("A+#NICE TRY#.
.. BUT . #I WIN#"):: GOTO 50
50
5040 CALL SOUND(2000,294,1,3
70,1,440,1)
5045 CALL HCHAR(22,1,32,96):
: IF GN=150 THEN CALL W(1000
):: CALL SAY("#YOU WIN#"
)
5050 W$="  FINAL SCORE "&STR
$(INT((TN*M*(TS*10)+(GN*5))/
(6-UN)))&"          " ::
 R=23 :: C=2 :: GOSUB 6000 :
: PRINT
5055 SC=MAX(HSC,INT((TN*M*(T
S*10)+(GN*5))/(6-UN)))
5056 IF SC>HSC THEN GOSUB 70
00
5057 HSC=MAX(SC,HSC)
5058 CALL SOUND(100,900,0)
5081 GAME=GAME+1 :: CALL KEY
(0,K,S):: IF K=5 THEN GAME=0
5082 IF GAME=5 THEN RUN "DSK
1.STOP"
5090 W$="  ANOTHER GAME (Y/N
)? "
5100 R=23
5110 GOSUB 6000 :: PRINT ::
IF MOVES$="AUTO" THEN GOSUB
```

```
9900 :: RESTORE 1100 ::
GOTO 190
5120 CALL SOUND(150,1397,1)
5130 CALL KEY(3,K,S)
5140 IF S=0 THEN 5130
5150 IF K=ASC("Y")THEN RESTO
RE 40 :: GOSUB 9600 :: GOTO
190
5151 IF K=ASC("N")THEN GOSUB
 9600 :: GOTO 5180
5154 GOTO 5090
5160 IF K=78 THEN 5180
5170 GOTO 5121
5180 CALL SAY("GOODBYE+UNTIL
+NEXT+TIME"):: END
6000 FOR Q=1 TO LEN(W$):: X=
ASC(SEG$(W$,Q,1)):: CALL HCH
AR(R,Q+C,X):: NEXT Q ::
RETURN
7000 OPEN #1:"DSK1.COMMANDO2
",INTERNAL :: PRINT #1:SC ::
 CLOSE #1 :: RETURN
7500 ACCEPT AT(24,3)BEEP:A$
:: OPEN #1:A$,INTERNAL,VARIA
BLE 200
7501 FOR R=1 TO 18 :: FOR C=
1 TO 32 :: INPUT #1:B$(R,C):
: NEXT C :: NEXT R
7502 FOR R=1 TO 70 :: FOR C=
1 TO 3 :: INPUT #1:B(R,C)::
NEXT C :: NEXT R :: FOR
R=1 TO 150 :: FOR C=1 TO 3 :
: INPUT #1:G(R,C):: NEXT C
7503 NEXT R :: INPUT #1:BN,G
N,BS,TS,GC,M,UN
7504 INPUT #1:ALT,GUARDS ::
FOR R=1 TO 5 :: FOR C=1 TO 2
 :: INPUT #1:U(R,C):: NE
XT C :: NEXT R
7505 INPUT #1:SC,TN,HSC :: C
LOSE #1 :: GOTO 990
7506 CLOSE #1 :: GOTO 396
7550 ACCEPT AT(24,2)BEEP:A$
:: OPEN #1:A$,INTERNAL,VARIA
BLE 200
7551 FOR R=1 TO 18 :: FOR C=
1 TO 32 :: PRINT #1:B$(R,C):
: NEXT C :: NEXT R
7552 FOR R=1 TO 70 :: FOR C=
1 TO 3 :: PRINT #1:B(R,C)::
NEXT C :: NEXT R :: FOR
R=1 TO 150 :: FOR C=1 TO 3 :
: PRINT #1:G(R,C):: NEXT C
7553 NEXT R :: PRINT #1:BN,G
N,BS,TS,GC,M,UN
7554 PRINT #1:ALT,GUARDS ::
FOR R=1 TO 5 :: FOR C=1 TO 2
 :: PRINT #1:U(R,C):: NE
XT C :: NEXT R
7555 PRINT #1:SC,TN,HSC
7556 CLOSE #1 :: IF E=0 THEN
 2340 ELSE END
9600 FOR D=1 TO 28 :: CALL S
PRITE(#D,125,INT(RND*12)+2,9
6,124,INT(RND*50)-INT(RN
D*50),INT(RND*50)-INT(RND*50
)):: CALL MAGNIFY(1)
9601 FOR C=1 TO 20 :: NEXT C
 :: CALL MAGNIFY(2):: NEXT D
 :: RETURN
9900 FOR D=1 TO 100 :: PRINT
 "Y ";:: NEXT D :: RETURN
15000 CALL SAY("DOWN"):: IF
ALT<1 THEN ALT=0 :: GOTO 185
0
15001 IF ALT=1 THEN 15002 EL
SE ALT=ALT-1 :: GOTO 1850
15002 IF B$(U(4,1),U(4,2))=C
HR$(117)THEN ALT=0 :: B$(U(4
,1),U(4,2))=CHR$(116)::
GOTO 1850 ELSE 15003
15003 CALL SAY("YOU+CAN+NOT+
PUT+IT+DOWN+THERE"):: CALL S
AY("THERE+IS+SOME+THING+
IN+THE1+WAY"):: GOTO 1850
16000 CALL SAY("UP"):: IF AL
T=0 THEN 16001 ELSE ALT=ALT+
1 :: GOTO 1850
16001 ALT=1 :: B$(U(4,1),U(4
,2))=CHR$(117):: GOTO 1850
16570 IF ALT=0 THEN 16571 EL
SE 16573
16571 IF B$(U(4,1),U(4,2)-1)
=CHR$(117)THEN B$(U(4,1),U(4
,2))=CHR$(117):: CALL HC
HAR(U(4,1),U(4,2),117):: U(4
,2)=U(4,2)-1 :: GOTO 16572 E
LSE 1640
16572 CALL HCHAR(U(4,1),U(4,
2),116):: B$(U(4,1),U(4,2))=
CHR$(116):: GOTO 1640
16573 U(T,2)=U(T,2)-1 :: IF
U(T,2)<2 THEN U(T,2)=2
16574 CALL HCHAR(U(T,1),U(T,
2)+1,ASC(B$(U(T,1),U(T,2)+1)
)):: GOTO 17000
16590 IF ALT=0 THEN 16591 EL
SE 16593
16591 IF B$(U(4,1),U(4,2)+1)
=CHR$(117)THEN B$(U(4,1),U(4
,2))=CHR$(117):: CALL HC
HAR(U(4,1),U(4,2),117):: U(4
,2)=U(4,2)+1 :: GOTO 16592 E
LSE 1640
16592 CALL HCHAR(U(4,1),U(4,
2),116):: B$(U(4,1),U(4,2))=
CHR$(116):: GOTO 1640
16593 U(T,2)=U(T,2)+1 :: IF
U(T,2)>32 THEN U(T,2)=32
16594 CALL HCHAR(U(T,1),U(T,
2)-1,ASC(B$(U(T,1),U(T,2)-1)
)):: GOTO 17000
16610 IF ALT=0 THEN 16611 EL
SE 16613
16611 IF B$(U(4,1)-1,U(4,2))
=CHR$(117)THEN B$(U(4,1),U(4
,2))=CHR$(117):: CALL HC
HAR(U(4,1),U(4,2),117):: U(4
,1)=U(4,1)-1 :: GOTO 16612 E
LSE 1640
16612 CALL HCHAR(U(4,1),U(4,
2),116):: B$(U(4,1),U(4,2))=
CHR$(116):: GOTO 1640
16613 U(T,1)=U(T,1)-1 :: IF
U(T,1)<1 THEN U(T,1)=1
16614 CALL HCHAR(U(T,1)+1,U(
T,2),ASC(B$(U(T,1)+1,U(T,2))
)):: GOTO 17000
16630 IF ALT=0 THEN 16631 EL
SE 16633
16631 IF B$(U(4,1)+1,U(4,2))
=CHR$(117)THEN B$(U(4,1),U(4
,2))=CHR$(117):: CALL HC
HAR(U(4,1),U(4,2),117):: U(4
,1)=U(4,1)+1 :: GOTO 16632 E
LSE 1640
16632 CALL HCHAR(U(4,1),U(4,
2),116):: B$(U(4,1),U(4,2))=
CHR$(116):: GOTO 1640
16633 U(T,1)=U(T,1)+1 :: IF
U(T,1)>18 THEN U(T,1)=18
16634 CALL HCHAR(U(T,1)-1,U(
T,2),ASC(B$(U(T,1)-1,U(T,2))
)):: GOTO 17000
17000 CALL HCHAR(U(4,1),U(4,
2),ASC(B$(U(4,1),U(4,2)))):: 
 GOTO 1640
30000 RESTORE 30500
30001 SP=0 :: FOR D=1 TO 10
:: READ W$ :: CALL SPRITE(#D
,ASC(W$),INT(RND*9)+2,SP
+6,133,5,0):: SP=SP+15 :: NE
XT D
30002 RESTORE 200
30003 RETURN
30500 DATA "C","O","M","M","
A","N","D","O"," ","4"
32000 RETURN
32700 ! COMMANDO 4 UPDATE
   TI-Extended BASIC.
(C) 1992 RICHARD TWYNING
32767 SUB CLK :: CALL PEEK(-
32750,S,@,M,],H,_):: DISPLAY
 AT(1,20)SIZE(8):STR$(H+
_*10-176)&":"&STR$(M+]*10-17
6)&":"&STR$(S+@*10-176):: SU
BEND
```

```
1 ! Richard Twyning's little
 typewriter program
2 CALL KEY(0,K,S):: IF S=0 T
HEN 2 ELSE PRINT CHR$(K);::
IF K=15 THEN CALL SAY("F
INISHED"):: END ELSE IF K=13
 THEN OPEN #1:"PIO.LF" :: PR
INT #1:CHR$(10):: CLOSE
#1 :: PRINT :: CH=0 :: GOTO
2
3 OPEN #1:"PIO.LF" :: PRINT
#1:RPT$(" ",CH)&CHR$(K):: CL
OSE #1 :: CH=CH+1 :: IF
CH=79 THEN CH=0 :: OPEN #1:"
PIO" :: PRINT #1:"" :: CLOSE
 #1
4 GOTO 2
```

TUR-MITEs....

This is taken from ALGORITHM 4.1, Jan-Mar 1993, and comes from Odd Arild Olsen of Norway. Unlike other graphics programs I have given, the joy of this one is not from the end result (which has its own limited attractions) but rather from the dynamics of watching it appear on screen. The plot is entirely deterministic, the end result depending on only three variable elements.

Picture if you will a tur-mite going for a walk - not entirely in a straight line! If he walks on a piece of ground without a footprint, he will leave one and turn ninety degrees in one direction. If he walks on a bit of ground with a footprint, he will scuff it out, and turn in the other direction.

Got that? OK, now imagine TWO tur-mites going for a wander around your tv screen and you watch as they each leave footprints or scuff out existing ones.

The result of their behaviour can be:
...both wander around a contained "nest" area.
...a path may be created and then erased continuously in a closed cycle.
...one stays in the nest while the other heads in a more or less straight line to the edge of the screen- his footprints leave a spiral type line.
...BOTH leave the nest and head off to the screen edge, often in directions at right angles to each other, sometimes in opposing directions, and once in a while in the same direction.
...Both trace out an ever increasing shape, roughly diamond shaped, which increases in size as they trudge around.
...and within these general classifications there are some oddities to be found!

The result is tied into the original direction of the second tur-mite relative to the first (same, 90 degrees, opposing) and the difference in the starting column and line (and -2 is NOT the same as +2). If they start far apart they may never meet of course!

The algorithm first, for a single tur-mite:

```
DIR=original direction tur-mite is moving in
GETPIXEL(X,Y) *is screen position on or off?
IF PIXEL OFF THEN DIR=DIR+1 :: TURN PIXEL ON
             ELSE DIR=DIR-1 :: TURN PIXEL OFF
IF DIR>3 THEN DIR=0
IF DIR<0 THEN DIR=3
IF DIR=0 THEN Y=Y+1
IF DIR=1 THEN X=X+1
IF DIR=2 THEN Y=Y-1
IF DIR=3 THEN X=X-1
go back to getpixel
```

You will notice we need to determine if a particular pixel is on or off. In the Basic family of languages we have, I only know of Myarc XB which allows this, so the program overleaf requires the Myarc XB (which requires a Myarc 512k ram card). The algorithm above may help you translate it into other languages.

```
100 ! TUR-MITES for Myarc XB
110 ! from Odd Arild Olsen
120 CALL DCOLOR(16,1)
130 CALL GRAPHICS(3)
140 CALL SCREEN(3)
150 RANDOMIZE
160 X=90+INT(RND*50-RND*50)
170 Y=110+INT(RND*50-RND*50)
180 X2=X+INT(RND*23-RND*23)
190 Y2=Y+INT(RND*23-RND*23)
200 IF RND<.1 THEN X2=X :: Y
2=Y ! BOTH START IN SAME PLA
CE ONE TENTH OF THE TIME
210 DIR=INT(RND*4) :: DIR2=I
NT(RND*4) :: IF RND>.3 THEN
DIR2=DIR
220 CALL GCHAR(X,Y,S)
230 IF S=0 THEN DIR=DIR+1 EL
SE DIR=DIR-1
240 S=(S=0)*-1 ! IF PIXEL OF
F THEN S=-1*-1=1 AND NEXT PI
XEL WILL BE PLOTTED ON- ELSE
 OFF.
250 CALL POINT(S,X,Y) ! S=1
TURNS PIXEL ON, S=0 TURNS IT
 OFF.
260 IF DIR>3 THEN DIR=0 ELSE
 IF DIR<0 THEN DIR=3
270 IF DIR=0 THEN Y=Y+1 ELSE
 IF DIR=1 THEN X=X+1 ELSE IF
 DIR=2 THEN Y=Y-1 ELSE IF DI
R=3 THEN X=X-1
280 X=MIN(X,190) :: Y=MIN(Y,
190) :: X=MAX(X,1) :: Y=MAX(
Y,1) ! STAY ON SCREEN
290 CALL GCHAR(X2,Y2,S2)

300 IF S2=0 THEN DIR2=DIR2+1
 ELSE DIR2=DIR2-1
310 S2=(S2=0)*-1

320 CALL POINT(S2,X2,Y2)

330 IF DIR2>3 THEN DIR2=0 EL
SE IF DIR2<0 THEN DIR2=3
340 IF DIR2=0 THEN Y2=Y2+1 E
LSE IF DIR2=1 THEN X2=X2+1 E
LSE IF DIR2=2 THEN Y2=Y2-1 E
LSE IF DIR2=3 THEN X2=X2-1
350 X2=MIN(X2,190) :: Y2=MIN
(Y2,190) :: X2=MAX(X2,1) ::
Y2=MAX(Y2,1)
360 CALL KEY(5,T,I) :: IF T=
32 THEN CALL LINK("DUMP",0,1
7)
370 GOTO 220
```

ALGORITHM is US$24 for 4 issues from:
P O Box 29237, Westmount Postal Outlet,
785, Wonderland Road, LONDON, Ontario, CANADA, N6K 1M6

The CALL LINK in line 360 uses a machine code utility for the Myarc XB which is available from the disk library.

....

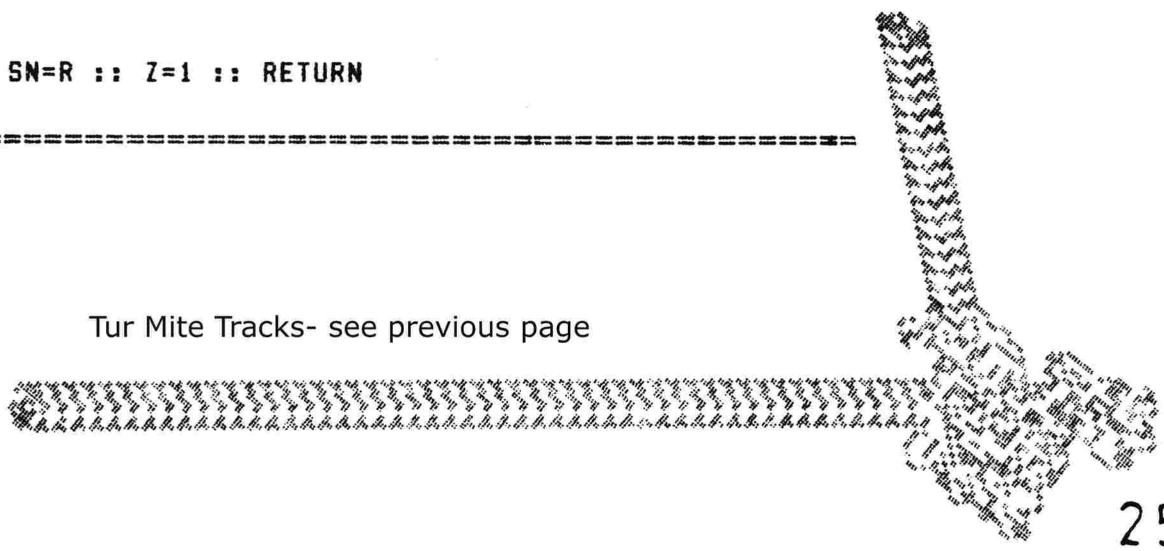
TEST TI*MES ISS:38 P.31 . J.Murphy for DOR-TIG

Only one submission to produce a solution to the above test (thanks!) - can you solve it faster? Send in your code!

```
100 CALL CLEAR :: CALL SCREEN(5):: FOR C=1 TO 12 :: CALL COLOR(C,16,1):: NEXT C
110 INPUT "  REQUIRE INFORMATION? Y/N":IN$
120 IF IN$="Y" OR IN$="y" THEN 130 ELSE 190
130 PRINT : : :"1/ THE NUMBERS YOU INPUT      WILL BE THE BOTTOM RIGHT    HAND
SIDE AND LEFT HAND      SIDE STEPS."
140 PRINT :"2/ ON SELECTING PRINTER THEN   EVERY TIME A REPEAT         NUMBER IS
 FOUND THIS WILL   BE PRINTED BY PRINTER,     ELSE PRINTED ON SCREEN."
150 PRINT :"3/ HOLDING DOWN 'C' WILL       PRINT,TO PRINTER, A LIST,   OF HIGHES
T NUMBER OF         STEPS FOUND UNTILL NO      HIGHER NUMBER OCCURS."
160 PRINT :"4/ TO STOP PROGRAM,HOLD DOWN   KEY 'Q'.": :"  PRESS ANY KEY TO CONTI
NUE"
170 CALL KEY(0,K,S):: IF K<1 THEN 170
180 !
190 !SHAW TEST TI*MES ISS:38 P.31 . Saved as 'NUMSTEPS'. J.Murphy for DOR-TIG
200 DIM A$(100),B$(100),NS(20),NA$(20),NB$(20),NF1(20),NF2(20)
```

```
210 CALL CLEAR
220 INPUT "PRINT RESULTS? Y/N ":PR$
230 IF PR$="Y" OR PR$="y" THEN PL=1 :: OPEN #PL:"PIO" :: GOTO 250
240 PL=0
250 INPUT "LHS 1st NUMBER ":LHS :: INPUT "RHS 1st NUMBER ":RHS
260 F1=LHS :: F2=RHS :: F3=F1
270 X=0 :: T=1
280 FOR I=1 TO 20
290 IF I=1 THEN 300 ELSE 310
300 A$(I)=STR$(F1):: B$(I)=STR$(F2):: GOTO 360
310 A$(I)=SEG$(STR$(VAL(A$(I-1))*VAL(B$(I-1))),LEN(STR$(VAL(A$(I-1))*VAL(B$(I-1)
))),1)
320 B$(I)=SEG$(STR$(VAL(B$(I-1))+VAL(A$(I))),LEN(STR$(VAL(B$(I-1))+VAL(A$(I)))),
1)
330 FOR CK=I-1 TO 1 STEP -1
340 IF A$(CK)=A$(I)AND B$(CK)=B$(I)THEN 380
350 NEXT CK
360 NEXT I
370 GOTO 460
380 PRINT #PL:" REPEAT FOUND NUMBER OF       STEPS= ";I-1:" REPEAT NUMBERS ARE ";
A$(CK);" & ";B$(CK)
390 NS(T)=I-1
400 IF NS(T)>=NS(T-1)THEN 410 ELSE 420
410 NS(T)=I-1 :: NA$(T)=A$(CK):: NB$(T)=B$(CK):: NF1(T)=F1 :: NF2(T)=F2 :: T=T+1
420 CALL KEY(0,K,S)
430 IF K=67 THEN GOSUB 540 ELSE IF K=81 THEN GOTO 520
440 PRINT #PL:" START NUMBERS WERE";F1;" & ";F2: :
450 IF PL=1 THEN 470
460 FOR D=1 TO 500 :: NEXT D
470 CALL CLEAR :: X=X+1 :: F2=F2+1 :: PRINT #PL:" NUMBER OF CHECKS= ";X
480 F3=F3+1
490 IF F3=F1+11 THEN 500 ELSE 510
500 F1=F1+1 :: F3=F1 :: F2=RHS
510 GOTO 280
520 IF PL=1 THEN CLOSE #1
530 END
540 OPEN #2:"PIO"
550 IF Z>0 THEN 580
560 PRINT #2:TAB(4);"NUMBER";TAB(15);"REPEAT";TAB(28);"LHS START";TAB(38);"RHS
TART"
570 PRINT #2:TAB(4);"OF STEPS";TAB(15);"NUMBERS";TAB(30);"NUMBER";TAB(39);"NUMBE
R" :: SN=1
580 FOR R=SN TO T-1
590 PRINT #2:TAB(6);NS(R);TAB(16);NA$(R);TAB(19);NB$(R);TAB(32);NF1(R);TAB(40);N
F2(R)
600 NEXT R
610 CLOSE #2 :: SN=R :: Z=1 :: RETURN
```

==


Tur Mite Tracks- see previous page

WHAT HAPPENED TEN YEARS AGO.....

NOVEMBER 1982
===============

-UK magazine Electronics Today Inter- national publish d-i-y computer project using 9995 CPU, 9929 VDP and TI Power Basic. Computer known as CORTEX.
99er Hall of Fame includes Cheryl Regena Whitelaw, Munchman 178,950 and Stephen Shaw Pinball (Video Games 1) 10,028,010.
-99er Magazine goes monthly

DECEMBER 1982
================

-Microsoft Multiplan for the 99/4A becomes a reality when the final bugs in the 4A version of the program are worked out. First actual showings of the program don't begin until March 1983 though.

JANUARY 1983
================

TI produces a little known 220 page catalog listing virtually every piece of software available for the 99/4A to that date, from both TI and other sources. The publication, which is authored by a team of seven TI employees, gives detailed descriptions of each program and the required hardware. The very same catalog was the source of information for the much valued and excellently presented Unisource Encyclopedia/Catalog. Verbatim descriptions from the TI catalog can be seen throughout the Unisource publication.

-TI announces the CC-40 (compact computer 40) on January 6, 1983.

COMPUTE! magazine begins coverage of the TI-99/4A with a single article that is written by C. Regena. It covers the hardware, software and miscellaneous resources of our computer.

TIHOME (UK User group) membership 700.

TI UK announce UK release of Parsec now, and UK release of TI99/2 in July 1983.
Rumours of TI modules from Thorn-EMI.
No TI99/4As on sale as TI fail to meet UK market demand.

FEBRUARY 1983
=================

The TI-99/2 computer is introduced to the world. Photographs of it and it's peripherals can be found in the February and March issues of 99er Home Computer Magazine.
-Ad in Compute! magazine lists TI-99/4A as a computer that will have ZAXXON for it by April.
DataSoft advertises for TI programmers in Compute! magazine to write games for the 99/4A line.
-C. Regena's "Programmer's Reference Guide..." book released by the Compute! Books Company.
-Thorn EMI announce proposed release of TI Modules Submarine Commander and River Rescue in Spring 1983.
-PC World announce that in 1982 TI sold 20,000 TI99/4As in UK.
-99er Mag become 99er Home Computer Mag
-TI reports 50 US User Groups

MARCH 1983
==============

Microsoft Multiplan is released for the 99/4A.

Harry Allston
10300 Kings River Road #57
Reedley, CA 93654-3622

November 09, 1990

The disks now added to your user group disk library hold the redefinition of alpha A-Z. I took the TIPS definition and traced to waxed paper then redefined them with TI-ARTIST PLUS!.

I got tired of the 'stepstair' ASC banners. The procedures to use the redefined characters are a little cumbersome and do require constant attention to the printer, BUT, they sure do make a nice looking banner. (an unbiased opinion)

1. OPEN TI-ARTIST to PRINT.
2. Make Printer ready.
3. Insert disk that includes the FIRST letter of the banner.
4. In PRINT mode, press 1.
   ENTER 'DSK(n).ALPHA-(n).
   Press R (rotate),
   Press V (full width),
   Press P (print).
   The R and V need not be changed in the course of printing the banner.
5. I find it takes 12 passes of the printer prior to printing.
6. When the last print pass takes place, press FCTN 4 (CLEAR). This stops the program and printer.
7. Roll the printer to the space desired between letters.
   DISENGAGE the friction feed!!!
   TURN OFF PRINTER.
   TURN ON PRINTER.
   If you do not initialize the printer you will print a 'do-hickey.
8. **PRESS 1 and enter new filename for next character. Make sure the proper disk is in the drive. Press P to print.**
9. **Place finger on top of printer to prevent false forward motion of paper. Continue to hold paper until you receive the FIRST PRINT PASS. QUICKLY release the friction enable lever. You are now on your second character.**

I know this sounds like a lot of bother but you will like the results.

I hope if a use is found for the program(s) that any improvements on the graphics will be passed on to me.
Enjoy.

==========================

PLAYING AMERICA IN XBASIC

By Earl Raguse

The following little XBASIC music program plays AMERICA. It illustrates one way that you can just copy the note frequencies you need from the TI XB Manual, and incorporate them into the program then forget them. Thereafter you just use the musical note names.

I first listed AMERICA in 28 column format, and there was so much wasted space, because so many lines exceed 28 columns by only 1 or 2 characters, I redid it!

```
100 ! SAVE DSK1.AMERICA
110 !By Earl Raguse 5/12/88
120 DISPLAY AT(6,11)ERASE ALL:"AMERICA"
130 X,Y,Z=30000 :: L=360 :: D=1 :: XV,YV,ZV=0
140 ME=330 :: MF=349 :: MG=392 :: MA=440 :: MBF=466 :: HC=523 :: HD=587
150 DISPLAY AT(8,0):"My Coun-try,"
160 X=MF :: D=2 :: GOSUB 1000
170 X=MF :: GOSUB 1000
180 X=MG :: GOSUB 1000
190 DISPLAY AT(8,13):"'tis of thee,"
200 X=ME :: D=3 :: GOSUB 1000
210 X=MF :: D=1 :: GOSUB 1000
220 X=MG :: D=2 :: GOSUB 1000
230 DISPLAY AT(10,0):"Sweet Land of"
240 X=MA :: GOSUB 1000
250 X=MA :: GOSUB 1000
260 X=MBF :: GOSUB 1000
270 DISPLAY AT(10,15):"Lib-er-ty"
280 X=MA :: D=3 :: GOSUB 1000
290 X=MG :: D=1 :: GOSUB 1000
300 X=MF :: D=2 :: GOSUB 1000
310 DISPLAY AT(12,0):"Of thee I"
320 X=MG :: GOSUB 1000
330 X=MF :: GOSUB 1000
340 X=ME :: GOSUB 1000
350 DISPLAY AT(12,11):"Sing."
360 X=MF :: D=6 :: GOSUB 1000
370 DISPLAY AT(14,0):"Land where my"
380 X=HD :: D=2 :: GOSUB 1000
390 X=HD :: GOSUB 1000
400 X=HD :: GOSUB 1000
410 DISPLAY AT(14,15):"fa-thers died,"
420 X=HD :: D=3 :: GOSUB 1000
430 X=MBF :: D=1 :: GOSUB 1000
440 X=MA :: D=2 :: GOSUB 1000
450 DISPLAY AT(16,0):"Land of the"
460 X=MBF :: GOSUB 1000
470 X=MBF :: GOSUB 1000
480 X=MBF :: GOSUB 1000
490 DISPLAY AT(16,13):"Pil-grim's pride"
495 X=MBF :: D=3 :: GOSUB 1000
500 X=MA :: D=1 :: GOSUB 1000
510 X=MG :: D=2 :: GOSUB 1000
520 DISPLAY AT(18,0):"From ev-'ry"
530 X=MA :: GOSUB 1000
540 X=MBF :: D=1 :: GOSUB 1000
550 X=MA :: GOSUB 1000
560 X=MG :: GOSUB 1000
570 X=MF :: GOSUB 1000
580 DISPLAY AT(18,13):"Moun-tain side, "
590 X=MA :: D=3 :: GOSUB 1000
600 X=MBF :: D=1 :: GOSUB 1000
610 X=HC :: D=2 :: GOSUB 1000
620 DISPLAY AT(20,0):"Let--Free-dom"
630 X=HD :: D=1 :: GOSUB 1000
640 X=HC :: D=.5 :: GOSUB 1000
650 X=MBF :: GOSUB 1000
660 X=MA :: D=2 :: GOSUB 1000
670 X=MG :: GOSUB 1000
680 DISPLAY AT(20,15):"ring."
          ! continued--->
```

UK Mains - IS it 240V AC from the socket?

I have had some problems recently with console lock ups, which did not seem to go away when cleaning module or side connector contacts.
Then I found a text advising that the console was very prone to lock up if the mains supply varied by more than about 5%.
That was my problem! Too low a voltage- keeps the bills down but not too good for computers. The solution was to remove all high wattage devices from the console mains ring when using the computer- the resulting small increase in supply voltage did the trick.
Over at my parents house we measured a supply voltage more than 5% OVER the standard 240v, and this was confirmed by the electricity supplier who put in a recording meter- high voltages put up bills and burn things out. Their supply voltage has now been adjusted to 240v.
So, any problems with electrical things, check your mains voltage- our privatised suppliers do not seem too worried about the voltage they feed to us these days!
stephen.

```
690 X=MF :: D=6 :: GOSUB 1000
990 END
1000 !**PLAY IT FUZZY**
1010 Y=X*.99 :: Z=X*1.01 :: YV,ZV=XV+8
1020 CALL SOUND(L*D,X,XV,Y,YV,Z,ZV)
1030 RETURN
================
```

Lets take a look at what goes on here. In line 130, several defaults are set; they don't really matter much in this program, but its good practice to make sure that you don't have loose ends.

Line 140: Here we equate the frequency values from the XB manual to the note names A,B,....F,G. We designate the octave starting with middle C by prefix M, and the octave above that by prefix H. In months to come we will designate an octave one lower or higher by adding prefixes L or H, then LL and HH for two octaves up or down. Flat notes are postfixed with F and sharp notes with S.

Most books make the octaves from C to C' rather than A to A' as TI did. The actual names given the octaves by musicians are strange words, and since we don't need them, I will never mention them except for the middle C octave which is called One Line and abbreviated C'. I will always use MC, less ambiguous.

In Line 150, I have DISPLAYed the lyrics just prior to coding and playing the measure associated with it. This makes it easy for the beginners, like me, to find a note sequence, if any corrections need to be made. I read text much better than music notation. I like to code a measure at a time, that way I can count measures on the staff and the program to find out where I am.

Displaying the lyrics this way has definite disadvantages however; as in this case, verse two frequently has exactly the same notes, but of course different words. If the words and music are kept separate, they can be reused as subroutines.

When I do that, I print out one verse of the lyrics, then play the music which I write as a subroutine. Then I print the next verse followed by the music subroutine, etc etc. Always carefully examine a piece of music for repeating sequences of notes. There is almost always some in a longer piece. Write these only once as subroutines and call them at the appropriate times.

Lines 160-180 demonstrate the technique used in this program. This is the way most experts do it, it is very versatile, and needs only one CALL SOUND subroutine (ie 1000). Take a look at it now, and note what the variables are.

In this case I need to initiate only one frequency variable X, as the other two variables (Y and Z) are mathematically related and taken care of in the PLAY IT FUZZY subroutine (1000). The other variable which must be specified, if changed, before calling the subroutine is D, the duration variable which is used to multiply the length L (1/8 note length).

One could also specify the volume variables XV,YV and ZV to get special effects. Later, I will show you a technique using READ and DATA statments, but this is only efficient with certain kinds of music.

⟶

Looking at the subroutine 1000, "PLAY IT FUZZY", note line 1010, this line sets up frequency variables Y and Z a little lower and a little higher in frequency than the main frequency and their respective volume variables (attenuation) YV and ZV somewhat higher (ie lower vol), all as a function of X and XV.

This saves a lot of code in the above line statements. The result here is to make the music frequency a little less pure, in other words Fuzzy. If you are a purist, and think you don't like this effect, just use a (!) to inactivate line 1010.

Now for the lesson of the day, with AMERICA typed, loaded and tried, including inactivation of 1010, type and MERGE the short program called QUAVERY. It too has a statement 1030 which you should try inactivating with (!).

```
1 ! SAVE DSK1.QUAVERY,MERGE
1000 !**PLAY IT QUAVERY**
1010 ! By E Raguse , 5/12/88
1020 !TRY COMMENTING OUT 1030
1030 Y=X*.99/2 :: Z=X*1.01/2 ::YV=XV+9 :: ZV=20 1040 FOR W=1 TO 2*D
1050 CALL SOUND(-L*D,X,XV,Y,YV,Z,ZV)
1060 CALL SOUND(-L*D,X/1.01,XV+2,Y,YV+4 ,Z,ZV+4)
1070 NEXT W
1080 CALL SOUND(-1,X/2,XV)
1090 RETURN
```

==

```
*********************************
**                             **
**  CONSOLE ROM RANDOM NUMBER  **
**      GENERATOR ROUTINE      **
**       AT >027A IN ROM       **
**  PLACE MODULO VALUE IN R13  **
**     PLACE SEED AT >83C0     **
**      BY MACK MCCORMICK      **
*********************************

RANDOM LI   R4,>6FE5     CONSTANT
       MPY  @>83C0,R4    >83C0 IS RND NUMBER SEED
       AI   R5,>7AB9
       MOV  R5,@>83C0    MOVE IT BACK OUT TO SEED
       MOVB *R13,R6      LOAD MODULO VALUE
       SRL  R6,8         SHIFT TO LOW ORDER
       INC  R6
       CLR  R4           SET UP R4,R5 FOR DIV
       SWPB R5           SWAP BYTES
       DIV  R6,R4        PERFORM DIVISION
       MOVB @WS+11,@>8378 MOVB R5LB TO >8378 RANDOM NUMBER
       RT
* YOU CAN ALWAYS SINGLE STEP THRU IT IN CONSOLE ROM TO SEE
* EXACTLY HOW IT WORKS. BY MACK MCCORMICK 74206,1522
* ANOTHER MEANS IS TO TAKE BYTES FROM ROM IN SEQUENCE. THEY CLOSELY
* APPROXIMATE A RANDOM NUMBER.
```

==============================

PLAYING CHARGE IN XBASIC
By Earl Raguse

I have chosen the attached program called CHARGE, it plays the bugle call of the same name. Notice, that it uses the standard CALL SOUND statement for each note using the frequency expressed as a number. This method is rather tedious for a larger program, but for this one it is relatively efficient. I have used a variable L for the note length, because it is used many times. Notice that I used a delay loop to do a rest in line 180. Rests can also be done using CALL SOUND(R, 30000,30) where R is the rest period. As an aside, a CALL SOUND statement can be used instead of a delay loop, in non-sound programs, provided the delay is not greater than about 4.25 seconds.

```
100 ! SAVE DSK1.CHARGE
110 ! By Earl Raguse 5/12/88
120 DISPLAY AT(12,11)ERASE ALL:"CHARGE"
130 L=125
140 CALL SOUND(L,1047,0)
150 CALL SOUND(L,1175,0)
160 CALL SOUND(L,1319,0)
170 CALL SOUND(L,1568,0)
180 FOR T=1 TO 125 :: NEXT T
190 CALL SOUND(L*2.5,1319,0)
200 CALL SOUND(L*6,1568,0)
210 END
```

===================================

Does anyone in the UK have a MIDI-MASTER interface? Dolores of Harrison Software has started a Midi user group and newsletter. The Casio 700 is not considered usable as it only allows one single channel for Midi.

===================================

BITS and PIECES

by Col Christensen
Brisbane User Group
Australia

SPRITELY SPRITES

Just a few little items in Extended Basic first. This one is a good old favourite called 3D Sprites we used to show to those poor souls with inferior computers to make their mouths gape. Just one program line and great things happen. If you don't know the routine, try it and see.

```
     100 CALL CLEAR :: CALL SCREE
     N(2):: CALL MAGNIFY(2):: FOR
      X=1 TO 26 :: CALL SPRITE(#X
     ,X+64,INT(X/2)+3,10,40,30,10
     4-X*8):: NEXT X :: INPUT A$
```

Here is a little demo program I made many years back to demonstrate the priority of sprites and illustrates the effect of a sprite of higher priority masking out any of lower order when they coincide on the screen.

```
100 CALL CLEAR :: CALL SCREE
N(2):: CALL MAGNIFY(2):: C=3
 :: RANDOMIZE
110 CALL CHAR(42,"FFFFFFFFFF
FFFFFF")
120 CALL CHAR(36,"00001F1131
F1FF42")
130 FOR X=1 TO 27 STEP 2
210 C=C+3 :: IF C>16 THEN C=
C-14
140 CALL SPRITE(#X,42,C,180-
X*4,220-X*6)
150 NEXT X
160 C=4
170 FOR X=2 TO 28 STEP 2
180 SP=INT(RND*12+5)
190 C=C+3 :: IF C>16 THEN C=
C-14
200 CALL SPRITE(#X,36,C,180-
X*4,220-X*6,0,-SP)
210 NEXT X
220 GOTO 220
```

Remember those messages picked out with monster LEDs that move across a display (the message not the LEDs). This is another demo routine used way back in the early 1980s simply by using the DISPLAY AT statement and was incorporated in an XBasic LOAD program .

```
100 CALL CLEAR
110 A$="
          THE TI-99/4A IS A C
OMPUTER MILES ABOVE ITS CLAS
S AND CAN RUN RINGS ROUND IT
S POOR COMPETITORS."
120 DISPLAY AT(22,1)SIZE(28)
:A$
130 A$=SEG$(A$,2,LEN(A$)-1)&
SEG$(A$,1,1)
140 GOTO 120
```

CONCATENATION IN BASIC AND TI-BASE

TI BASIC AND TI-BASE command files allow the printing or display to the screen of character and numeric variables and literals. How you type in these instructions varies though. A basic program line is stored in memory as one long string of code but appears on the screen wrapped around on up to 4 screen lines.
e.g. An XBasic program line might apear on the screen as:

```
160 RANDOMIZE :: FOR I=1 TO
32 :: X=INT(RND*100)+1 :: N
BR(I)=X :: NEXT I :: CALL CL
EAR :: CALL MAGNIFY(2)
```

TI-BASE is different in that it stores its command files on disk as D/V 80 records and each record in that file is taken from just one screen line which is 40 characters in length. So it appears that this situation places a limit on the length of TIB's printing and display capabilities.

TIB (here short for TI Base!) overcomes this limitation by providing for, like Basic, a form of concatenation. This is a horrendous word but simply it means the construction of a larger string from a number of smaller components. Basic provides the & symbol for this purpose. e.g.

```
100 A$="THE ANSWER IS"&NUMB
ER&"SQUARE"&UNIT$
110 PRINT A$
```

TIB interprets concatenation if a ¦ (FCTN/A) is placed between the bits to be joined. If a splillover from one screen line to the next occurs then it is necessary to place a semicolon (;) at the end of each but the last line to indicate that more concatenation is to follow. The following line needs to begin with a ¦. See TI-BASE Manual pages 1-2, 3-13 and 5-9/5-10

```
LOCAL STRING C 80
LOCAL NUMBER N 4 0
LOCAL UNITS C 8
REPLACE STRING WITH "THE ANSWER IS";
  ¦NUMBER¦" SQUARE "¦UNITS
DISPLAY STRING
```

In operation, both of the above could give a readout of

```
THE ANSWER IS 156 SQUARE METRES
```

TIB can also handle long quotes in a REPLACE WITH command using any number of screen lines. Note the (;) at the end of each line and the (¦) at the beginning of the next.

```
REPLACE STRING WITH "Release the alpha";
  ¦ "lock key and use the joystick con";
  ¦ "trollers"
DISPLAY STRING
```

I had problems when I tried to send a long concatenated string to the printer with the TIB PRINT statement though. The concatenation shown above only worked following a REPLACE WITH and a PRINT (or DISPLAY) the variable as shown above. I wanted simply to imitate the basic pending print separator (;) function that can be used in a program line such as ;

```
PRINT "THE ANSWER IS";NUMB
ER;"SQUARE ";UNIT$.
```

It was a nuisance and a waste of good memory space to have to use the REPLACE WITH statement in a TIB command file before each PRINT.

The TIB Manual makes no mention of possible ways make a print statement extend beyond the one screen line. But finally and recently, I accidently discovered how it is done and so simple too. How could I ever have been so stupid. All one has to do is to use the semicolon symbol (;) at the end of each line on the screen. So a piece of TIB command file to send a long string to the printer could look like:

```
     PRINT "Release the alphalock key ";
      "and use the E, S, D, and X keys";
      " or Joystick 1."
```

Notice that there is no (|) needed at the beginning of each subsidiary line like in the concatenation following a REPLACE WITH statement. But if quoted text and variables are to be mixed in the one print statement, at least one space must separate each.

```
     PRINT "THE ANSWER IS" NUMBER ;
      "SQUARE" UNITS "WHICH IS ";
      "THE AREA OF THE SQUARE"
```

PEB POWER SUPPLY 1

Garry gave me a call one day to say that his system had shut down on him. There were no card lights lit up and the PEB seemed to be dead. His tests with a multimeter indicated that for the +16v rail less than a volt was showing, and on the negative supply he measured around -50v. Weird. Some slow thinking (I'm not prone to quick thinking) gave me the impression that the DC earth connection was open circuit or even the transformer centre tap was not at earth potential. Without an earth connection, I reasoned, the "earth" would float somewhere between the positive supply and the negative supply.

Its position would be governed by the amount of current drawn in each half of the supply. As heavy current is used in the +5v in the cards and only relatively low current in the negative supply, the earth point should float at a point very near to the positive supply leaving about double the normal voltage below the earth point. This was what the symptoms pointed to. When he had time, Garry opened the PEB and sure enough, the earth connection on the card that the connector plugs into had overheated and cooked. He soldered the wire directly to the board, powered up and all worked again.

PEB POWER SUPPLY 2

Another instance of PEB failure occurred in one of Chas' boxes after a card whose connector was a slightly sloppy fit in the PEB backplane socket was inserted. The nominal +16v and the -16v supplies (actually they come out at around 23v each) are on adjacent connectors and a misaligned card could short the two together.

This seemed to have been the cause because voltages in the PEB were as dead as doornails. Fortunately, all the cards still worked perfectly in another expansion box so no damage had been done to them. Continuity tests on the transformer primary windings showed an open circuit. Once before (Bugbytes, April 1987) I reported that the manufacturer of the transformer saw fit to put a fuse INSIDE the tranny and cover it all over with tape and a nylon cover under which no one would know it lurked. I soon uncovered the fuse only to find that it already had a jumper wire connected from one end to the other. --->

There had to be a break somewhere else in the winding. My plan now was to unwind the primary winding down till I came to the open circuit, repair it then rewind the wire back on. I stripped the iron core from the windings, no easy task, and proceeded to remove the rest of the tape from the primary winding.
Lo and behold, there was A SECOND FUSE, blown of course. Bridging it with a wire brought back continuity into the primary winding so I removed both fuses, connected together the wiring ends that were joined to the fuses and insulated them. The next thing to do was to replace the hefty 1.5A slow-blow mains fuse at the back of the PEB with a standard .5A one though I would have preferred to try a .25A slow blow type to give better protection but I didn't have one at the time. If a fuse goes now, it can be easily replaced.

==================================

ACCESSING THE DISK DRIVE WITH ASSEMBLY LANGUAGE

ASSEMBLY PROGRAMMING- ADVANCED LEVEL

ACCESSING THE DISK DRIVE DIRECTLY

Written by Mack McCormick.

The subject is disk DSR routines at the disk ROM and direct access to the controller chip level. In reviewing the information I have on the subject I find that the entire scope is covered in five books for a total of about 2,000 pages. Obviously, I must limit my discussion.

The disk DSR is developed on three levels:

Level 1 - Basic disk functions. Sector Read/Write, head control, drive selections, track formatting and buffer allocation

Level 2 - The "file" concept. Each file is assessible by its name and an offset of a 256-byte block relative to the beginning of the file.

Level 3 - Extension to the user level. Fixed or variable length records or files.

One other level which you won't find documented is direct access to the controller chip in the controller card.

I intend to confine my discussion to level 1 and chip level routines. Due to length, this tutorial will discuss sector I/O.

There are three different controller chips contained in the three different controller cards on the market (TI, CorComp, MYARC). The chips are all made by Western Digital. They are the WD 1771, WD 2793, and the WD 1770 respectively. Once again I will limit this tutorial to the TI controller card and its chip.

Everything in this tutorial will pertain to all three cards except direct access to the controller chip and its associated commands.

First lets review the TI controller card features and ROM. As you know it can control up to 3 DS/SD drives. There are 40 tracks per drive and 9 sectors per track. Each sector is 256 bytes in length. Track 0 is closest to the outside and track 39 nearest the center of the disk.

There is a built in DSR ROM which contains 6 level one routines which may be executed by branching to them. These will accomplish almost all we need to do except things like track I/O, Volume Information Block update and others. To get at these routines you must access the Floppy Disk Controller (FDC) chip directly. To accomplish this we need to know how the FDC chip accesses the drive and build from there.

These are some of the features of th WD 1771 chip: Automatic track seek with verification, in the read/write mode single/multiple sector read/write with automatic sector seek. Writes entire track for formatting. Programmable track to track step times.

Six registers:

Data shift register - Assembles serial data from the disk read and transfers during write.
Data Register - 8 bit holding register during read/write operations. During a seek command it contains the desired track position.
Track Register - 8 bit register that contains the track number of the current read/write head position. Incremented by one as the head steps in toward track 39 and decremented by one towards track 00. Contents are compared with the disk track number in the ID field during read, write and verify.
Sector Register - 8 bit register for holding the desired sector position. Contents compared with the disk sector ID field during read and write operations.
Command Register - 8 bit register for the command to be executed.
Status Register - 8 bit register to hold drive status.

There are eleven commands available:

		Bits							
Type	Command	7	6	5	4	3	2	1	0
I	Restore	0	0	0	0	h	V	r1	r0
I	Seek	0	0	0	1	h	V	r1	r0
I	Step	0	0	1	u	h	V	r1	r0
I	Step In	0	1	0	u	h	V	r1	r0
I	Step Out	0	1	1	u	h	V	r1	r0
II	Read Command	1	0	0	m	b	E	0	0
II	Write Command	1	0	1	m	b	E	a1	a2
III	Read Address	1	1	0	0	0	E	0	0
III	Read Track	1	1	1	0	0	1	0	s
III	Write Track	1	1	1	1	0	1	0	0
IV	Force Interrupt	1	1	0	1	I3	I2	I1	I4

Plug in the appropriate values by type command:
(see next page)

Type I
h = Head load flag. 1-beginning. 2-not beginning.

V = Verify. 1-verify on last track. 0-no verify.

r1r0 = Stepping motor rate. 0 0 - 6ms. 1 0 - 10ms. 1 1 - 20ms.

u = Update flag. 1-update track register. 0-no update.

Note: Head step times are based on the 1 MHz clock contained in the controller card.

Type II
m = multiple record. 0-single. 1-multiple.

b = Block length flag. 1-IBM format (256 Byte). Other flags only if need to know.

a1a0 = Data Address Mark 00->FB(Data Mark)

Type III
s = Synchronize Flag. 0-Single density.

Type IV (interrupt condition flags)

I0 = 1, not ready to transisition.

I1 = 1, Ready to not ready transisition

I2 = 1, Index Pulse

I3 = 1, Immediate Interrupt

E = Enable head load and 10 msec delay
    1-delay. 0-head already loaded no delay.

This all seems confusing now but before its all over you should have a better understanding.

Head loading means the read/write heads are placed in contact with the disk (the click you hear when the drive activates) and data may be transfered. The head stays loaded until a command is received to unload or until timeout occurs (2 disk revolutions).

I suppose this is the best place to cover the disk format. Have you ever wondered what's in between the data fields (256 bytes), well here it is. (ON NEXT PAGE!)

Number of Bytes	Whats There
12	Index Gap. >FF
6	Sync >00
* Sector begins here. Repeat 9 times *	
1	ID Single density >FE
1	Track Address >00->27
1	Side >00
1	Sector Address >00->08
1	Sector Len >01
2	Cycle Redundancy Check >F7
11	Data Separator >FF
6	Sync >00
1	Data Address Mark >FB
256	File Data
2	CRC >F7
* Sector ends here *	
36	Data Separator >FF
240	End of track fill >FF

From this you can see there are 3177 bytes per track but only 2304 are actual data bytes.

There are three ways to perform a sector I/O. You may use the DSRLNK, access the disk ROM without DSRLNK, or access the controller chip directly.

Lets examine the first two methods.

Sector I/O is commonly refered to as subprogram 010. All arguments for the I/O are passed thru the FAC block in CPU RAM (>834A). Here's how it maps out:

>834A-4B	{Address of actual sector accessed when complete.}
>834C	Disk drive 1,2, or 3.
>834D	Read/Write 0=write. <>0=read
>834E-4F	VDP Buffer Address (256 byte size).
>8350-51	Sector Number

Error codes returned at >8350 after operation. 0=no error. 1=error.

```
****************************************
*                                      *
*   SECTOR I/O ROUTINE DEMO USING      *
*               DSRLNK                 *
*   ACCOMPANIES SECTOR I/O TUTORIAL    *
*    BY MACK MCCORMICK 74206,1522      *
*                                      *
****************************************
       DEF  SECTOR
       REF  VMBW,VMBR,DSRLNK

PABI   DATA >0110        SUBPROGRAM 010
CPUBUF BSS  256          CPU BUFFER


SECTOR LI   R0,>F80      ADDRESS OF PAB
       LI   R1,PABI      PAB
       LI   R2,2         TWO BYTES
       BLWP @VMBW        WRITE PAB TO VDP
       LI   R1,>0101
       MOV  R1,@>834C    /DISK DRIVE 1, <>0=READ
       LI   R1,>1000
       MOV  R1,@>834E    /VDP BUFFER START ADDRESS
                         / AT LEAST 256K
       CLR  R1
       MOV  R1,@>8350    /LOOK AT SECTOR 0
       LI   R1,>F80
       MOV  R1,@>8356    POINT TO THE PAB AT >8356
       BLWP @DSRLNK      ACCESS THE DISK
       DATA >A           USE DISK DSR SUBROUTINES
* NORMALLY YOU WOULD CHECK FOR ERRORS AT >8350 HERE
* YOU COULD ALSO CHECK >834A FOR ACTUAL SECTOR READ

*------------------------------*
*  PUT IT UP ON THE SCREEN     *
*------------------------------*
       LI   R0,>1000     VDP BUFFER ADDRESS
       LI   R1,CPUBUF    CPU BUFFER ADDRESS
       LI   R2,256       MOVE 256 BYTES DOWN
       BLWP @VMBR
*THIS WOULD BE THE PLACE TO MANIPULATE DATA BEFORE WRITING
                         IT BACK UP
       CLR  R0           SIT POSITION 0
       BLWP @VMBW        WRITE UP TO SCREEN IMAGE TABLE
*------------------------------*
*  WRITE BACK OUT TO DISK      *
*------------------------------*
       LI   R1,>0100     /DISK 1, WRITE/
       MOV  R1,@>834C
       BLWP @DSRLNK      WRITE IT BACK OUT
       DATA >A
       JMP  $            YOU WOULD EXIT THE PROGRAM HERE
       END
* YOU CAN SEE HOW EASY IT IS TO WRITE A SECTOR COPIER JUST FROM
* THIS SHORT CODE ADD A FEW WHISTLES AND BELLS AND YOU HAVE A
* FIRST CLASS PRODUCT
```

!!!

```
****************************************
*             2D EXAMPLE               *
*   SECTOR I/O ROUTINE DEMO USING      *
*        DIRECT ROM ACCESS             *
*   ACCOMPANIES SECTOR I/O TUTORIAL    *
*    BY MACK MCCORMICK 74206,1522      *
*                                      *
****************************************
       DEF   SECTOR
       REF   VMBW,VMBR,GPLWS

SUBR   DATA >0110          SUBPROGRAM 010
CPUBUF BSS   256           CPU BUFFER
MYREG  BSS   >20           MY WORKSPACE


SECTOR LWPI  GPLWS

       LI    R1,>0101
       MOV   R1,@>834C     /DISK DRIVE 1, <>0=READ
       LI    R1,>1000
       MOV   R1,@>834E     /VDP BUFFER START ADDRESS/
                           /AT LEAST 256K
       CLR   R1
       MOV   R1,@>8350     /LOOK AT SECTOR 0
       LI    R12,>1100     SET CRU REGISTER TO BASE ADDRESS OF
                                 >1100 DSK DSR ROM
       SBO   0             PAGE IN THE DISK DSR ROM TO >4000

* Of course you could eliminate the next five instructions and
* manually scan the DSR ROM for the word which immediately
* proceeds >0110 and loaded R9 with that value which is >56DC
* in the case of the CorComp card and BL directly to it.  I
* scanned the link table so this program could be used with
* other DSR subroutines and with all controller cards.
       LI    R9,>4000      BEGINNING OF DISK DSR ROM
NEXT   C     *R9+,@SUBR    SEARCH LINK TABLE FOR ENTRY POINT
       JNE   NEXT
       AI    R9,-4         SUBTRACT 4 FOR ENTRY POINT
       MOV   *R9,R9        GET THE ENTRY POINT ADDRESS
       BL    *R9           BRANCH TO THE ROUTINE

* NORMALLY YOU WOULD CHECK FOR ERRORS AT >8350 HERE
* YOU COULD ALSO CHECK >834A FOR ACTUAL SECTOR READ

*-------------------------------*
*  PUT IT UP ON THE SCREEN      *
*-------------------------------*
       NOP                 NOP IS REQUIRED HERE BECAUSE THE DSR
                           ROUTINE INCT's THE
*                          RT ADDRESS
       LI    R0,>1000      VDP BUFFER ADDRESS
       LI    R1,CPUBUF     CPU BUFFER ADDRESS
       LI    R2,256        MOVE 256 BYTES DOWN
       BLWP @VMBR
*THIS WOULD BE THE PLACE TO MANIPULATE DATA BEFORE WRITING
                                 IT BACK UP
       CLR  R0            SIT POSITION 0
       BLWP @VMBW         WRITE UP TO SCREEN IMAGE TABLE
*-------------------------------*
*  WRITE BACK OUT TO DISK      *
*-------------------------------*
       LI   R1,>0100      /DISK 1, WRITE/
       MOV  R1,@>834C
       BL   *R9
       SBZ  0             PAGE OUT DISK DSR
       JMP  $             YOU WOULD EXIT THE PROGRAM HERE
       END

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

*****************************************
***                                   ***
*** This program does a sector read   ***
*** at the FDC level FD1771. This     ***
*** tutorial is part of the continuing ***
*** series for advanced programmers.  ***
*** Will only work with TI card as    ***
*** written. Must INV commands to work ***
*** with CorComp. TI card is on an INV ***
*** data bus. CorComp is not.         ***
***                      Mack         ***
***                                   ***
***                                   ***
*****************************************
       DEF  SECTOR

WS     EQU  >83E0         LETS USE SOME HIGH SPEED RAM FOR OUR WS

***************************************
*       FD1771 DEFINITIONS            *
***************************************
FDS    EQU  >5FF0         READ STATUS
FDRT   EQU  >5FF2         READ TRACK REGISTER
FDRD   EQU  >5FF6         READ DATA REGISTER
FDC    EQU  >5FF8         COMMAND REGISTER
FDWT   EQU  >5FFA         WRITE TRACK REGISTER
FDWS   EQU  >5FFC         WRITE SECTOR REGISTER
FDWD   EQU  >5FFE         WRITE DATA REGISTER

***************************************
*       REGISTER DEFINITIONS          *
***************************************
VALUE  EQU  0             GENERAL
VALUE1 EQU  1             GENERAL
RAMPNT EQU  2             VDP RAM POINTER
COUNT  EQU  6             GENERAL PURPOSE COUNTER
TEMP   EQU  7             USED TO STORE RT ADDR
TEMP1  EQU  8             USED TO STORE RT ADDR
CRUBAS EQU  12            CONTAINS CRU ADDRESS
VDP    EQU  15            CONTAINS ADDR OF VDPWA

***************************************
*            CRU DEFINITIONS          *
***************************************
MOTBIT EQU  1             MOTOR ON BIT OFFSET
WAIBIT EQU  2             WAIT LOGIC ENABLE
HLTBIT EQU  3             HEAD LOAD TIMING BIT
DS1BIT EQU  4             FIRST DRIVE SELECT BIT OFFSET
```

```
****************************************
*            VDP DEFINITIONS           *
****************************************
VDPWA  EQU  >8C02          VDP WRITE ADDRESS
VRD    EQU  >8800-VDPWA    READ DATA
VWD    EQU  >8C00-VDPWA    WRITE DATA
VRS    EQU  >8802-VDPWA    STATUS

****************************************
*          MISC DEFINITIONS            *
****************************************
T75M   EQU  75*40          75 MILLISECONDS
T1000M EQU  40000          ONE SECOND
NSEC   EQU  9              # SECTORS PER TRACK
SECLEN EQU  256            # BYTES PER SECTOR
MEMSTA EQU  0              VDP LOCATION FOR DATA
                            (SCREEN IMAGE TABLE HERE)

*-- PROGRAM STARTS HERE --*

SECTOR
       LWPI WS
       LI   CRUBAS,>1100 CRU BASE ADDRESS
       LI   VDP,VDPWA      VDP WRITE ADDRESS IN R15
       SBO  0              PAGE IN THE ROM SO WE CAN
                                   GET AT THE FDC

****************************************
*        SELECT THE DRIVE              *
****************************************
       LI   RAMPNT,>0100 DRIVE 1, >02 DRV #2, >04 DRV #3.
       AI   CRUBAS,2*DS1BIT   SET UP THE CRU ADDRESS FOR THE
                                                  PROPER LINE
       LDCR RAMPNT,3       SELECT THE DRIVE
                           (BYTE ADDR SOURCE OPERAND)
       AI   CRUBAS,-2*DS1BIT BACK OUT THE OFFSET
       LI   VALUE,T75M     WAIT 75 ms
DRIVE  SRC  TEMP,4         OK TO ROLL TEMP
       SRC  TEMP,4         JUST WASTING THE PROPER TIME
       DEC  VALUE
       JNE  DRIVE

****************************************
*  SEEK SECTOR                         *
* This routine calculates the track and*
*  sector numbers from the logical     *
*  sector number and seeks the proper  *
*  track. Drive is restored to track 0.*
****************************************

*-- RESTORE THE DRIVE --*

       BL   @LCMD          ISSUE A RESTORE CMD
       DATA >F500

       BL   @BUSY1         WAIT FOR RESTORE TO COMPLETE

       SETO VALUE          INVERT 0 BYTE
       MOVB VALUE,@FDWT    OUTPUT THE CURRENT TRACK NUMBER (TRK 0)
```

```
*-- COMPUTE THE CORRECT TRACK NUMBER FROM THE LOGICAL SECTOR --*
       LI    VALUE1,0       THE NUMBER HERE IS THE SECTOR YOU
                            WANT TO READ
       CLR   VALUE          DIVIDEND HIGH
       LI    TEMP,NSEC      PUT NUMBER OF SECTORS IN TEMP
       DIV   TEMP,VALUE
       SWPB  VALUE          QUOTIENT IS TRACK #
       INV   VALUE          INV IT FOR FDC CHIP
       MOVB  VALUE,@FDWD    LOAD FD1771 DATA REGISTER
       SWPB  VALUE1         REMAINDER IS SECTOR #
       INV   VALUE1         FDC CHIP LIKES INV VALUES
       MOVB  VALUE1,@FDWS   LOAD FD1771 SECTOR REGISTER
       CB    VALUE,@FDRT    SAME TRACK AS LAST TIME?
       JEQ   SEEK           YUP...GO ON AND DO IT
       BL    @LCMD          SEEK AND VERIFY THE SECTOR ADDRESS
       DATA  >E100
       BL    @BUSY1         WAIT FOR COMMAND TO FINISH

* NOTE HERE THAT SINCE THE DATA MUST BE MOVED FROM CPU RAM TO
* VDP RAM HOW MUCH TIME TI COULD HAVE SAVED BY GIVING US A CPU
* WRITE OPTION LIKE MYARC DID.  INSTEAD IF WE WANT TO USE IT IN
* CPU WE MUST MOVE IT BACK DOWN WHEN USING THE DISK ROM.


*-- READ THE SECTOR --*

SEEK   LI    RAMPNT,MEMSTA MEMSTA IS VDP RAM LOCATION TO WRITE TO
       BL    @VDPLAW        PREPARE FOR VDP WRITE

       BL    @LCMD          FINALLY WE CAN READ THE SECTOR
       DATA  >7300          REMEMBER THIS IS INV
       SBO   WAIBIT         ENABLE WAIT LOGIC
       LI    COUNT,SECLEN BYTES PER SECTOR
RSECT  MOVB  @FDRD,VALUE    GET THE BYTE
       INV   VALUE          IT WAS INVERTED SO REINVERT IT.
       MOVB  VALUE,@VWD(VDP) PUT IT IN VDP
       MOVB  @FDRD,VALUE    THESE INST ARE REPEATED BECAUSE
                                SPEED IS CRITICAL
       INV   VALUE
       MOVB  VALUE,@VWD(VDP)
       DECT  COUNT          WE'VE WRITTEN TWO BYTES
       JNE   RSECT          FINISHED?
       BL    @BUSY          DISABLE WAIT LOGIC AND WAIT FOR
                               MOTOR TIME OUT

       SBZ   0              PAGE OUT THE ROM
       LIMI  2              ENABLE INTERRUPTS
       JMP   $              THATS IT FOLKS. FCTN QUIT TO EXIT.

****************************************
*SUBROUTINE-TURN MOTOR ON & ISSUE CMD  *
*     INPUT: CMD IN MSB OF DATA        *
*      NOTE: ALL COMMANDS MUST BE INV  *
****************************************
LCMD   MOV   *R11+,VALUE    PICK UP COMMAND BYTE
       SBZ   MOTBIT         TOGGLE MOTOR ON CLOCK
       SBO   MOTBIT
       LI    COUNT,T1000M ONE SEC TIME OUT
MOTOR  SRC   TEMP,4         WASTE TIME
       SRC   TEMP,4           "    "
       DEC   COUNT
       JNE   MOTOR
       MOVB VALUE,@FDC   LOAD THE COMMAND
       SBO  HLTBIT       SET HLT
       SRC  TEMP,8       WASTE MORE TIME (MOTORS ARE SLOW-
       SRC  TEMP,8       COMPUTERS ARE FAST <grin>)
       RT

****************************************
*      BUSY ROUTINE                    *
* Waits for the current command to be  *
* completed. No error on motor time out*
****************************************
BUSY   SBZ  WAIBIT       DISABLE WAIT LOGIC

BUSY1
       MOVB @FDS,VALUE   GET THE STATUS OF THE DRIVE
       INV  VALUE        INVERT FOR DATABUS LOGIC
*      JLT  NODISK       THIS IS WHERE YOU CHECK FOR NO DISK
       SRC  VALUE,9      CONTINUE WAIT LOOP?
       JOC  BUSY1        YUP...BEEN A CARRY
       RT

****************************************
* VDP READ/WRITE SET UP ROUTINE        *
* Address in vdp is RAMPNT             *
****************************************
VDPLAW ORI  RAMPNT,>4000 SET WRITE BIT
       ANDI RAMPNT,>7FFF STRIP MSB
       JMP  VDPLA1

VDPLAR ANDI RAMPNT,>3FFF STRIP OFF 2 MSB
VDPLA1 SWPB RAMPNT       ALWAYS WRITE LSB FIRST
       MOVB RAMPNT,*VDP  LOAD TO VDP
       SWPB RAMPNT
       MOVB RAMPNT,*VDP  NOW WRITE THE MSB
       ANDI RAMPNT,>3FFF GET RID OF WRITE BIT
       RT
NINE   DATA 9
       END
```

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!eof!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

02/19/92 20:53

Extended BASIC AVERAGE - A Tinygram

By: Andy Frueh, Lima UG

I don't know how many teachers or bowlers we have out there. Face it, sooner or later, everyone has to average out SOMETHING. Prices, scores, grades, whatever. This program is simple. You can do the same thing very easily on a calculator. But it does show beginners how the TI can keep track of how many INPUTs were made, and compute answers based not only on WHAT was input, but how many items were input. It also shows how you can take a string from INPUT and turn it into a numerical value to be computed.

Enter E to calculate average of the values entered.

```
100 I=0::DISPLAY AT(13,1)ERASE ALL:"Input value"
200 I=I+1::DISPLAY AT(23,1):I::ACCEPT AT(23,3)VALIDATE(DIGIT,"E"):I$
300 IF I$="E" THEN 400 ELSE V=VAL(I$)::T=T+V::GOTO 200
400 PRINT "Average Value is: ";T/I:"of";I;"items."
```
==

REVIEWS OF TI EDUCATION MODULES

by Phillis Peyton, grade 5 classroom teacher
reprinted from IUG NEWSLETTER September 15, 1982

READING ROUNDUP

The Reading Roundup module provides instruction and practice in three reading skills: figures of speech, work meanings, and idioms. While the stories are written simply enough to be read by a student in third grade, the skills are sophisticated enough to challenge a sixth grade student. The module could be used to provide remedial help for a student in junior High School without making him feel that he must read juvenile material.

[BB&P EDITOR'S NOTE: The next two paragraphs provide a good description of the general structure of most of the READING modules, including the 1983 "rare" cartridges.]

Two activities are provided for each skill. The "Study it" activity provides instruction through examples presented in a colorfully illustrated story. The student's rate of reading will not be a factor contributing to his success or failure at learning the skill. He is allowed to pace himself, pressing "enter" when he has had ample time to finish reading the material on the screen. In the "Study It" activities for each skill, the student is given opportunities to respond, but scores are not tallied.

A correct response causes an appropriate signal such as "Right" to flash on the screen. The signal is accompanied by a catchy melody. Incorrect responses result in an opportunity to make a second choice or, by pressing "Aid", to reread the material and then try again. a second incorrect response causes the correct answer to be shown to the student. The student is allowed to learn to improve his reading skills without fear of failure when he makes a mistake or is learning by trial and error.

The "Try It Out" activity for each skill contains paragraph length stories and allows the student to type in a character's name, thus personalizing the stories for him. The name will appear in each of the stories in the activity. At the conclusion of the activity, a score is shown.

The figure of speech taught in Activities 1 and 2 are similes and metaphores. Both are used to show comparisons and are common literary techniques used to cause the reader to form a mental picture. Similies use the word LIKE or AS. Examples of similies are:

1. He is as gentle as a newborn deer.
2. The kite soared like an eagle.

Metaphores form the comparison without the words LIKE or AS. An example of a metaphore is: What a railroad engine of an ox!

The student is expected to learn to tell what two things are being compared and how they are similar. The words "simile" and "metaphor" are not used in the module.

In activities 3 and 4 the student is shown how to use context clues to determine the meanings of words that are unknown or have multiple meanings. It is not always possible or even desirable to stop and use a dictionary every time we encounter an unfamiliar word or a familiar word used in a new and different way. The ability to use context clues is an invaluable aid to reading with comprehension.

Activities 5 and 6 give the student an opportunity to learn the meanings of some common idioms that our English language is so full of. "Sitting on pins and needles" is an example of an idiom used to indicate nervousness. Many adults do not realize the difficulty a child can have in understanding the figurative language that he hears and reads.

The young child's language is completely literal -- He means what he says, and he says what he means. Since our language makes use of so many idioms, the knowledge of their meanings can result in higher comprehension scores for the student.

All three skills are combined in activity 7 as a culminating activity.

The four lengthy stories all follow a "Western" theme accompanied by appropriate music in keeping with the title of the module -- Reading Roundup. I highly recommend its use for improving reading skills, particularly by the student in the intermediate grades (4-6).

[available from the disk library to load with MYARC XB or any module which has the Ed/As grom plus ram at >6000 eg Super Space]

DIVISION 1

Division 1 command module created by Scott Foresman and Company for Texas Instruments will be an invaluable aid in the classroom as well as in the home. Because it is a complete text on division facts, its use will cover a wide range of ages and levels of ability. Division is commonly introduced in grade three, and the facts are reviewed through grade six. Grades three through six, then, are the levels at which this module will be used most extensively. It will also be useful to challenge a younger gifted student and as a remedial tool for those students above sixth grade who have not achieved mastery of division facts.

The nine activities available for selection are:

1. Meaning of Division
2. Divisors of 1, 2, and 3
3. Divisors of 4, 5, and 6
4. Divide using !-- (the division sign)
5. Practice and Paint
6. Divisors of 7, 8, and 9
7. How many boxes?
8. Divide With a Remainder
9. Make a Picture

The activities proceed in sequence from the least difficult to most difficult. Each activity may be worked independently of all others. However, the ability to work successfully at each activity depends upon the mastery of skills that have been introduced in the preceding activities.

By working through Activity 1 the student will receive an excellent explanation of what actually happens during the division process. This writer has known students who had memorized division facts and still lacked an understanding of the concept of division. The explanation on the module is made without using the words "divide" or "division", and without using either of the signs normally used to work division problems.

In Activities 2 and 3 the use of the sign / (the "division sign") and the number sentence form are introduced. When the working form is introduced in Activity 4, using the vertical format and the sign !---, the transition is make simple by showing both forms and actually moving each number from the number sentence to its proper position in the new format.

The relationship between multiplication and division is stressed in Activity 6 by showing a "check" in which the divisor and the quotient are multiplied. an incorrect answer causes the complete multiplication table for that divisor to be displayed on the screen.

The concept of remainders is illustrated in Activity 7 by evenly grouping and having "leftovers". The word "remainder" is used in Activity 8 and the working form is shown. The student learns to give the quotient and the remainder.

At the onset of each activity the student may choose to see one or more excellent teaching examples. Exceptions to this are Activities 5 and 9 because they were designed to be checkup activities.

The illustrations on the computer screen are more effective than even very attractive textbook illustrations. The book's pictures are stationary, while items on the screen may actually be repositioned to show the grouping process.

The learner receives a simulation of using manipulatives, a concrete approach required by many children efore they can proceed to more abstract learning. Through the use of the voice synthesizer the student hears the equation as he sees what is taking place. The result is that he is receiving information in three modes: visual, auditory, and kinesthetic. By involving all of these senses in the learning process, retention chances are much greater. DIVISION 1 is sure to be a popular and enjoyable aid to learning.

[Anyone got this on disk for XB load????]

READING FUN

Reading Fun is Scott Foresman's reading skills module for the younger child in the primary grades. The module contains four illustrated stories accompanied by musical background. At the onset of each story three words that are possibly new ones for the student are shown on the screen. The child may, by pressing the number next to any of the words, hear it pronounced and see it used in a sentence. When the word appears later in the story, he may receive the same help by pressing "Aid".

The first three stories provide instruction in one skill each. After the child has had several opportunities to respond to questions in a non threatening way, he is invited to try out what he has learned. he is then given a series of ten questions over some short passages of reading. At the conclusion of the activity, the child's score is shown. If he responded correctly on the second try, he is given credit for a correct answer. He has the opportunity to look at the text of the story again before he attempts to correct his answer.

The first story deals with problems and how people solve them. The child learns to identify the problem from a list of three possibilities. He also selects the solution that was used in the story.

The second skill is labeled "Why things happen." Educators usually refer to it as the ability to distinguish cause and effect.

Thirdly, the child learns to watch for clues that tell how characters feel. He must know the meanings of some common words that describe people's feelings, moods, and emotions. Some of the words used are: tired, happy, angry, and upset. The answers to some questions are stated directly in the story. Other questions such as, "How did Ann probably feel?" require that the child draw some conclusions or use some inference skills.

The fourth story allows the child to use what he has learned about all three skills. Questions asked are:

What is the problem?

What caused the problem?

At this point the child is asked to pick one of the three main characters to solve the problem. He is given a choice of three different actions that character might take in attempting to solve the problem. After his choices have been made, he is told, "Now let's see what happens next." the text of the story continues according to the child's choices and he can then see for himself whether or not he has chosen wisely. He may try as many of the nine possible solutions as he desires. By choosing possible solutions to these problems, a child can begin to learn to predict the outcomes when certain courses of action are taken, and to think about the possible consequences for actions that people take.

These and the other skills dealt with in the module are feferred to as reading skills. having mastered them, a child will almost certainly become a better reader. They might be more appropriately named, however, as thinking and living skills.

[Available from the disk library to load with Myarc XB or with any module which has the Ed/As grom plus ram at >6000, eg Super Space]

===========================

NEVER RELEASED OFFICIAL TI MODULES

described by Charles Good
Lima Ohio User Group

The Lima User Group has obtained some module software that we have never seen actually offered for sale. This software was developed in house by Texas Instruments or was created by third parties under an official TI license. Most of this software has a Texas Instrument copyright statement on the title screen. In some cases these modules were released to very limited circulation. Most of the modules were never released at all, mainly because of TI's withdrawal from the home computer market. This time I will talk about utility and application software.

DISK MANAGER 3

The title screen says "copyright Texas Instruments" but gives no date. This apparently was the disk manager that was to accompany the TI double sided double density disk controller. Two of these controllers, formatting to 1280 sectors DSDD, were sold by one of the dealers at the 1989 Chicago faire. DM3 looks very much like the well known DM2. However DM3 offers you the choice of drives 1-4, whereas DM2 only lets you use 1-3. Also, with DM3 the defaults on the screen for "initialize disk" are "double sided" and "double density." In all other respects DM3 appears identical to DM2. I successfully initialized disks with DM3 and my CorComp controller SSSD and DSSD. I was not able to make DM3 successfully initialize in DSDD format, probably because I think DM3 tries to make 1280 sector (8 sectors per track) disks in double density. I have no trouble initializing 1440 sector DSDD disks on my CorComp controller using the DM2.

CLASS DATA RECORDER

Copyright 1981 by Scott Foresman and Company. This is one part of the "School Management Applications" series developed by Scott Foresman, a large publisher of school text books as well as some better known TI educational software. An entire school system could be managed with this software. Packages for inventory control, payroll, salary planning and analysis, student scheduling, and analysis of student grades within a class and throughout the entire school system were supposed to be included in this series.

This complete School Management Applications series is described in the booklet "Texas Instruments Home Computer Program Library" (copyright TI 1982, numbers "CL581C" and "1043605-001" on the back cover) that was packaged in the box with my first 99/4A. CLASS DATA RECORDER is given the Scott Foresman identification number 30406 in this booklet. The "School Management Application" series is also described in a 1982 article by Dr. Tom Hansen published in vol 1, #5 of 99er magazine. Of the 15 separate parts of this series, only CLASS DATA RECORDER and SCHOOL MAILER are available to me.

CLASS DATA RECORDER is for use by teachers to keep track of and analyze classroom student grades. The module is a hybrid of GPL and TI BASIC code, resembling in this respect the Personal Record Keeping module. When using CLASS DATA RECORDER with a Gram device it is necessary to have TI BASIC (Groms 1 and 2) on line.

It is necessary to have a newly initialized SSSD disk, or a disk which has previously been used with CLASS DATA RECORDER in drive 1 when you select the software from the main title screen. If there are any files on the disk that CLASS DATA RECORDER does not recognize, you will get an error message. There is alot of disk activity when using CLASS DATA RECORDER. Apparently the program stores most of its data on disk rather than in memory, allowing the classroom teacher to manipulate large amounts of data.

You are first asked for the date.

On first use with a clean disk you are then asked:
Course Title
Period of the day
Term of the school year
Teacher name
Will assignments be weighted?

You then get to enter the names of all the students in the class. Then you enter the assignments, each with the following information:
Name (of the assignment, such as "Quiz 1")
Is it pass/fail?
Weight (only used if assignments are weighted)
Total Points (the point value of this assignment)

Later, when you again boot this data disk you are again asked for the date, and then given the choice:
1. Enter/Edit data
2. Print Reports.

Selecting #1 gives you a chance to enter grades or manipulate data. You are, from #1 above, given this menu:
1. New assignment and scores.
2. Assign gradelines
3. Edit records
4. Add a student.
5. Assign final grades.
6. Add a new course.

If you choose to PRINT REPORTS, you are given the following choices:
1. Class list
2. Individual Student Summary
3. Cumulative Class Averages
4. Class Averages/assignment
5. Rank List
7. Histogram.

Printing is done to RS232. If you have a Gram device, you can change this to PIO.

As an experiment I have recently used CLASS DATA RECORDER to keep track of student data in one of the classes I teach. It is very user friendly. Although I don't have access to the original documentation I have had no problems figuring out how to use CLASS DATA RECORDER except for the initial first time startup procedure (SSSD disk with no files on it in DSK1.). This can be a useful piece of software to any teacher IF a way can be found to direct output to a printer.

SCHOOL MAILER

Copyright 1981 by Scott Foresman and Company. This is also part of the "School Managment Applications" series. The module is a combination of GPL code and TI BASIC and requires two drives. First time use requires a blank SSSD in both DSK1. and DSK2.

This software is used to generate address labels for the students in a single school or an entire school system. Printing output is to RS232. Labels can be printed based on grade, teacher, building, zip code, etc. The program serves as a data base for student and parent names addresses and phone numbers.

On first use you are asked the following information about each student in the data base:

Name
Grade
Room
Sex;
Parent or guardian name
Street Address
City
State
Zip code (up to 9 digits)
Option field 0-9 (I havn't figured out the meaning of this yet.)

Upon subsequent booting of a data disk you are given this menu:

1. Enter information (as above)
2. Edit/Display information
3. Print Reports
4. Upgrade all students (indicate that they have been promoted to the next grade).
5. Delete a grade

DISK DUPLICATOR -release 1.0

There is no date or copyright notice. The powerup menu has these choices:

1. Duplicate Pascal Disks
2. Duplicate Basic disks
3. Compare Disks
4. Diskette Quality Test
5. Catalog Disk.

In reverse order to save the most interesting for last, #5 is identical to disk catalogs of the DM2 and DM3 modules.

#4 offers you the choice of "Destructive test (YN)".

The Compare Disks routine (#3) will terminate the first time a difference is found in a sector by sector comparison. However the exact nature of the difference and the sector location of the difference are not given.

I was not able to check out PASCAL disk duplication (#1). Duplicating BASIC disks (#2) is the most interesting feature of this module. Duplication only works with a SSSD master. You can make two copies of the master disk onto two copy disks with only one keypress. You put the master disk in DSK1. and the backup disks into DSK2. and DSK3.

First the copy disks are initialized, one at a time. Next, sector by sector information is read into memory from the master disk and then output to the first copy disk and then to the the second copy disk. Since this is sector by sector copying (something DM2 and DM3 don't do) copying is rather slow compared to what is possible with a track copier. I suspect that track copiers were unknown when DISK DUPLICATOR was created.

DIAGNOSTIC TESTS

Copyright 1979 by Texas Instruments. This is designated is PHM3000, the lowest numbered module in TI's official module number system.

Available for XB load on disk from the Group disk library.

It was designed to test the 99/4 (no A) console. No equivalent for the 99/4A was ever produced. This module, like the dealer demonstration module PHM3001, was not sold widely, if at all, to the general public. The DIAGNOSTIC TESTS module is mentioned in the documentation "ADDENDUM" that was packaged in the box with the first 99/4A I purchased in 1982. A console diagnostic module is a good idea. Diagnostic tests on disk, such as those released by TI to user groups a few years ago, require a more or less working console as well as a functioning disk controller and drive. Some module problems would prevent testing from disk.

The title screen and main menu of DIAGNOSTIC TESTS use the large console character set, the character set used from the powerup "Press 1 for TI BASIC" etc menu. This is somewhat unusual. The main menu gives these choices:

1. Automatic test
2. Keyboard test
3. RAM test
4. Video display
5. Sound test
6. Calculation test
7. Cassette test
8. Handset test
9. Maintenance test

Choices #1 and #3 put various multicolored patterns on the screen. I don't know what these patterns mean.

The keyboard test (choice 2) lets you press any key and have its character displayed on the screen. This includes the arrows, which are displayed as arrows. Lower case characters, as well as non arrow FCTN characters give meaningless displays.

Video Display (choice 4) lets you view a bit map mode display (called pattern mode by the module) that includes sprites and an interrupt driven count down clock. You can also view 40 column "text mode" and "multicolor mode". The latter is rarely used in TI software, and includes squares composed of 4 pixels with the color of each square independent of any other. The "multicolor mode" test display is interesting.

Choice 5, sound test, automatically tests all aspects of the sound chip. Some of the generated test sounds are quite pleasing. This is not a speech test. TI speech was not around in 1979.

The calculation test, choice 6, automatically checks addition, subtraction, multiplication, division, log functions, trig functions, and "miscellaneous" functions. You don't see much on the screen.

Choice 7 exercises CS1 and CS2. You have to verify that the cassettes are doing what the computer says they are supposed to be doing.

The biggest surprise in this module is the handset (joystick) test. The screen display indicates that FOUR JOYSTICKS can be tested. Tests include all 8 joystick positions plus the fire button. Does anyone know if the 99/4 had provisions for four joysticks? [stephen here... no it did not, but hand held units were originally designed. This test picks up TWO joystick positions for each direction, testing for values of 4 or 7. stephen].

The maintenance test (choice 9) brings up a display that says "for repair technician only." You can check groms 0,1, and 2. You also test sound, VDP RAM, and XML. I don't know what "XML" is [check your editor assembler manual-stephen]. The screen display for most of these tests simply says "test in progress." You do hear something in the sound test, and a ghosty immage flashes across the screen in the VDP RAM test.

Released or Not? A Review of "Shanghai"
By: Andy Frueh, Lima UG

With the recent coverage of "lost" TI items, I thought it would be a good time to review this "Concept Education Game." It was produced by Funware. I have seen about five of their products advertised. These include St. Nick, Ambulance, Pipes, Driving Demon, and Rabbit Trail. I have seen a few more that were never advertised, and I have never seen the advertised "Pipes." There are supposedly several more programs out there by Funware.

"Shanghai" starts out with the typical Funware title screen. The name of the program is in hugh blocky letters, and a graphic having to do with the game is in the middle. This is not impressive, but it is the actual game that counts. It teaches basic principals of having a business: making money and staying in business.

The action is somewhat slow, but is sufficient to keep people busy for a while without wanting to quit. The concepts involved are basic, but very young children may have difficulty. Be sure they know or you explain the idea of selling at higher cost than production costs to make a profit.

Pressing any key starts the game. This is the scenario: You are a merchant sailing down a river. You have to buy and sell items as quickly as you can, before your expenses exceed your revenue. An overhead view showing your ship and other ships, and the coastline is on the left of the screen.

You have a total of three boats. You only loose one if you crash into one of the large ships that occasionally cruise by. You are faster than they are, and getting out of their way is easy. Crash three times and your game ends. Your game can also end as soon as your expenses are greater than the money your making (in other words, once your profit is $0).

On the right side of the screen is an information display. It shows your revenues and expenses. Profit is shown by subtracting the later from the revenues. Below this is your current cargo with a maximym of 5 items, and how many ships are left.

At the bottom of this display is the direction of the wind, which influences your control of the boat. Also is a box for the high score. In the middle of this side of the screen is a list of the six different commodities or items that can be bought and sold, along with current buy and sell prices. The whole idea is to buy things with a higher re-sell value.

Items are scattered on the coastline. To buy an item, move the ship so the nose is touching the item and press fire. If an item is blinking, then that marks a point where you can sell all of that particular item. For example, if the item looking like a duck was blinking, you can sell any "ducks" you have there, but no other items can be sold.

Buying and selling instantly affects your profit. Of course, you loose all your current cargo if you crash. That means if you spent 15 "dollars" buying an item and crash before you can sell these, you have lost that money. Occasionaly, there are black pirate boats that try to take all of your cargo. The same thing applies here as with crashing. These ships are very difficult to shake off.

If a prices suddenly is in a red box, that means that the price is going to change. It seems like they usually just drop, but they may also rise. This also keeps you on your toes.

It does a very good job of teaching its concepts, and is fun for an educational game.

Shanghai is available on disk for XB+32k from the disk library.

WHY TI?
(THE HISTORY OF TEXAS INSTRUMENTS)

[BB&P editor's note: The original source of this article is unknown. It was found by Bill Gaskill among his piles of TI stuff.]

1930-- Commercialized its invention, the Reflection seismograph, and revolutionized petroleum exploration. corporate name was "Geophysical Service", abbreviated as GSI.

1946-- Diversified by adding electronic systems manufacturing.

1952-- Entered the transistor business with a new corporate name: "Texas Instruments Incorporated".

1954-- Became the first company to mass produce germanium radio stansistors; DEVELOPED THE FIRST COMMERCIAL TRANSISTOR RADIO; Introduced the first commercial silicon transistor - the type required in space and military systems.

1958-- Announced TI INVENTION OF INTEGRATED CIRCUIT, which provides the basis for virtually all modern developments in electronics.

1961-- Invented the semiconductor thermal printer.

1967-- Introduced TI invention: the world's first electronic handheld calculator.

1969-- Announced first data terminal to use thermal printing, The Silent 700.

1970-- Invented the "Single-Chip-Microprocessor" which today is the "brain" of a wide range of products.

1971-- Commercially introduced the microcomputer, or "Miracle Chip", a TI invention that includes all the elements of a complete computer, including memory, in one integrated circuit.
1972-- Entered consumer end-user market with the DATAMATH handheld calculator, initially priced at $149.95
1973-- Introduced a "4K-Bit Random Access Memory" (RAM), setting new industry standard.
1975-- Introduced first 16-bit mocroprocessor family to use memory-to-memory architecture, increasing performance. Introduced "3D" seismic data gathering and processing.
1976-- Developed "Solid State Software" plug in modules for pocket calculators.
1977-- Received patents for "Closed-Loop" solar energy system for converting sunlight to electricity.
1978-- Introduced revolutionary SPEAK&SPELL learning aid using synthetic speech chip. [BB&P editor's note: In Sept 1992 you can still buy this product at my local Toys-R-Us.]
1979-- Introduced first "64K Erasable Programmable Random Access memory" (EPROM); It was preceded by TI introduction of first 16K EPROM in 1977 and first 32K EPROM in 1976. Introduced a Home Computer.
1980-- Produced first commercial single-chip, 16 bit microcomputer, the TMS9940.
1981-- Introduced "TI LOGO", the first microcomputer language enabling children to use computer to solve problems. Introduced the TMS 7000 family of 8-bit single-chip mocrocomputer circuits [BB&P editor's note: These chips are used as CPUs in the CC40 and TI74]. Achieved volume production of leadership 64K bit dynamic RAMs.

This takes us into 99/4 history, more to follow...

THE NEW FUNNELWEB v5 TEXT EDITOR- 80 columns only so far...
described by Charles Good
Lima Ohio User Group

Tony McGovern has released a "completely rewritten from source code" Funnelweb version 5 80 column editor dated Dec 15, 1992. A similar 40 column version will follow . These v5 editors are designed to run from the Funnelweb v4.4 environment. So far, the only "version 5" parts of Funnelweb are the text editors. They are fully multi-lingual and compare favorably with Asgard's new FIRST DRAFT word processor. New features, added or revised since the v4.4 editor are summarized below.

HOW TO OBTAIN THIS SOFTWARE:

The 80 column v5 editor files are available from your User Group disk libraran, two disks for the Editor and another two disks for a compatible formatter and other bits.

HELP SCREENS AND MULTIPLE FILES IN MEMORY:

FIRST text in memory - the edit buffer:

SECOND text in memory - the help screens:
When the 80 column editor first boots it loads into memory up to four help screens. These can be viewed from the command line by pressing H(elp). Each screen is 26 lines by 80 columns and they pop up on screen immediately because they are already in VDP memory. The Program Editor loads one set of help screens relating to assembly language coding. The Word Processing editor loads another set of screens more appropriate for help with text editing and formatting.

You can move back and forth from one help screen to another by pressing the Q and A keys. FCTN/9 returns you back to the edit buffer. Sets of useful help screens are provided, and the user can also create personalized help screens. A utility is provided to convert the first 26 lines of any DV80 file into an 80 column help screen. {The 40 column editor will have 28 line by 40 column help screens. An unlimited number of these screens can be loaded into memory one at a time from disk by pressing H(elp) from the 40 column command line.}

THIRD text in memory - screen viewing:
As in the v4.4 text editor, one can do a S(how) D(irectory), move the cursor next to a DV80 file name, press a key, and display a screen of that file. Subsequent presses of the same key window down through the entire file. There is no limit to the size of the file that can be viewed one screen at a time in this manner. This file isn't really stored in memory, just displayed on screen.

FOURTH text in memory - the V(iew) buffer:
From S(how) D(irectory) you can put the cursor next to the name of a DV80 file, press CTRL/V and load the whole file (or any part of it) into a 64K memory buffer for instant recall any time during the editing process. Once loaded into the V(iew) buffer the file can be scrolled one line at a time or windowed up and down very rapidly. Pressing <enter> from within S(how) D(irectory) or V from the editor command line pops this text into view. This V(iew) buffer can hold very large text files. It is in fact the same memory area as Funnelweb's Disk Review "V" text buffer. You can load some text into the 80 column editor's V(iew) buffer, exit the editor to a central menu, and from there go to Disk Review. After performing some disk management functions from Disk Review, you can go back to the 80 column text editor and the V(iew) buffer text will still be there! You can also load ANY KIND OF FILE into Disk Review's V(iew) buffer. Then if you exit Disk Review and go to the 80 column v5 text editor this text will be waiting for you in the editor's view buffer. Just press "V" from the editor's command line to see the text you loaded into Disk Review! {Because 40 column systems have only a limited amount of VDP memory, this V(iew) buffer feature is not available from the 40 column v5 editor.}

FIFTH text in memory - the ST(ore) buffer:
From the editor command line you can press ST(ore) and move the contents of the edit buffer into temporary storage in VDP RAM. You can then load another file into the edit buffer, edit the second file and save it back to disk, then press RE from the command line to RE(call) the ST(ore)d text back into the edit buffer. The ST(ore) buffer acts as a temporary ramdisk, but is much faster. Text files are saved and loaded to horizon ramdisks one record (line of text) at a time. This is fast but definately not instantaneous. Large files take 10s of seconds to load and save with a horizon. The ST(ore) buffer response time is immediate! It is too bad you can't exchange text between the edit and store buffers. Tony says this trick would eat up lots of memory and that's why such a featurehas not been included. {ST and RC to and from VDP RAM is not available from the 40 column v5 editor. There isn't enough memory.}

NEW FILE SAVING AND PRINTING OPTIONS:

These are available in both the 40 and 80 column editors and are accessed via the P(rint)F(ile) command. You can configure the editor with printer codes. Then every time you insert a "P" in front of the printer name (such as PF <enter>, P PIO) the editor will send these preconfigured codes to the printer before any text. I have my v5 editor set up to send the "print all the following in emphasized print" command. If I also use a "Q" with PF the editor will send a printer reset code to the printer after all text has been printed (PF <enter>, P Q PIO).

You can append the contents of the text buffer to the end of an existing disk file by specifying the disk file as the printer device preceeed by an "A" (PF <enter>, A DSKx.FILENAME). DV80 files of unlimited size can be created this way. I build multiple choice exams for my students this way, one question at a time taken from question lists I have stored as DV80 files.

You can also use PF to create DF128 text files readable directly by MS-DOS and Unix software.

NEW POWERUP OPTIONS:

Normally the v5 editor boots in either Word Processing or Program Editing mode depending upon which of Funnelweb's central menus is used to select the editor. However, if you hold the space bar down as the editor loads you get a list of choces. The editor can be pre-configured to always give you this list of choices without pressing the space bar or to automatically boot as any one of the choices uness you press the space bar.

1. Word Processing
2. Program Editor

Then you get these choices:

1. Default 7-bit
2. National 7-bit
3. TI Euro Writer
4. All Chars.

If you want the resulting disk file of your document to be readable by someone else on another computer using anything except Funnelweb v5 (such as an earlier version of Funnelweb, or TI Writer) then select items 1 or 2 from this menu. Items 3 and 4 from the above menu do some fancy stuff (more about this later), but produce disk files that can only be read and displayed on screen properly with Funnelweb v5.

After you chose one of these options, you are given the following choice of languages, comparable to what is suggested by the TI Writer module:

1. Australia---My 40 column beta test editor lists this as "default". This is the one USA English users would choose. It uses character sets C1 or C2, the same character sets used by the rest of Funnelweb. This is the only option that does not load in additional character and command sets from disk.
2. British---Choosing this loads in a separate character set that redefines SHIFT/3 as the British pound sterling symbol. In all other respects "2. British" is the same as "1. Australia".
3. France
4. Deutschland
5. Italia
6. Sverge
7. Nederland
8. Espania

Choosing a non English language loads in foreign language character sets that redefine little used keyboard characters such as FCTN/A, FCTN/F, FCTN/W, etc as appropriate foreign characters. Most of these foreign characters are vouls with accent symbols over them such as umlaut, grave, acute, or circumflex. These character sets and their ASCII values correspond to some of the international character sets 1-9 found on most modern printers. This means that if you send the a control code to set your printer for the appropriate foreign character set then the foreign characters you see on screen will be printed properly. From the editor the epson compatible printer key sequence with no spaces between keypresses is CTRL/U FCTN/R CTRL/U R CTRL/U SHIFT/A thru I CTRL/U where A-I specify the particular character set (1-9) desired. For Genimi 10X and SG10 printers substitute 7 for R in the above key sequence.

Non-English languages also load appropriate foreign text into the command line and change the command abbreviations to reflect the foreign language. For example, in French "Imprimer Fichier" means Print File, and you use the command IF, not PF, to print stuff. The Swedish version has "Lagra Filer" for Save File. The command LF in the Swedish text editor will save (not load) a file. This can be disasterous for English speakers who don't know Swedish.

Not all the foreign commands and command line text are finished. English, German, and Swedish are complete. French and Dutch are almost complete. Spanish hasn't been started. Sample source code and a utility that creates foreign commands and command line text are included for those interested in expanding Funnelweb v5's multilingual capabilities.

EURO-WRITER:

For Europe, TI prepared a multilinual version of TI Writer (TIW v2) with some special features. By selecting EURO-WRITER from the powerup menus, the Funnelweb v5 editor provides all the features of the TI Writer v2 editor; specifically an intuitive way of adding accent marks to vouls.

When in Funnelweb's EuroWriter mode you have access to the foreign character set of the language you select from the powerup menus, and these character sets include some, but not all, accented vouls. But there is another intuitive way to create accented vouls that lets you put ANY ACCENT over ANY VOUL. Type a voul, either upper or lower case. Then backspace to put the cursor back over the voul, type any of four FCTN/key or CTRL/key combinations, and an umlaut grave acute or circumflex mark will appear on screen over the voul! The only problem is that these voul/backspace/accent screen displays are coded with high ASCII numbers above 127 and don't normally print properly. You need to print text files with these accented vouls using the European formatter (also multingual), the formatter that TI included with TI Writer v2. You need to use special transliteration files that rededine ASCII codes greater than 127 as accented vouls. This formatter with its auxiliary language and transliteration files is not part of the Funnelweb v5 editer package, but the files can be obtained by anyone from the Lima user group. Unfortunately, the European formatter REQUIRES use of the TI Writer module. It hasn't yet been modified to run easily out of the Funnelweb environment using something other than the TIW module to boot Funnelweb. Also, transliterations to print some of the accented vouls are less than ideal. Accentd vouls you see clearly on screen with Funnelweb's Eurowriter mode may look strange when printed.

ALL CHARS MODE:

Our 99/4As normally can directly type ASCII 0-127 with ASCII characters below 32 accessed from CTRL/U "special character mode". But our 8 bit computer is capable of generating codes 0-255. When high ASCII codes <127 are sent to a printer during text printing the printer will print graphic symbols. A common standard for these high ASCII graphics is the IBM character set #2 found on most printers. High ASCII codes sent to a printer with IBM graphics #2 enabled print line shapes somewhat comparable to the "lines" font of Page Pro that prints those neat borders and page dividing lines. Check your printer's manual to see what these graphics look like.

[SPECIAL NOTE FOR STAR SG10 PRINTER OWNERS: There is an undocumented software method of switching from STAR mode to the IBM character set #2. You don't need to use a dip switch. The code with no spaces between keypresses is CTRL/U FCTN/R CTRL/U w CTRL/U SHIFT/A CTRL/U. To switch from IBM set #2 back to STAR mode use this code: CTRL/U FCTN/R CTRL/U w CTRL/U SHIFT/2 CTRL/U. The w in these codes is lower case.]

Selecting All Chars mode with the Funnelweb v5 editor allows you to directly type on screen and print to the printer ASCII 0-254 of the IBM character set #2. This includes all the normal upper and lower case letters numbers and keyboard symbols, plus the graphic symbols coded by high ASCII numbers. To type the graphic symbols type CTRL/, (control and comma simultaneously) and then each keypress will produce a graphic symbol. To return to the keyboard normal letters type CTRL/, again. Normal letters and graphic symbols remain on screen as you use CTRL/, to toggle the keyboard back and forth beteen graphics and normal. Graphics and text print normally using PF (PrintFile). You don't need a formatter to print these graphic symbols.

SOME OF THE OTHER NEW FEATURES:

--You can move text up and down from within the command line. This is very handy for M(ove lines)), D(elete lines), and C(opy lines) operations. You don't have to remember line numbers. Go to the command line and type M, D, or C. Then use the arrow or up/down screen keys to display the first line number and last line number so that you will enter the proper numbers to M, D, or C.

--From the various fixed modes with an open box cursor (Program editor or WP with word wrap off) you can break lines at the cursor, insert text, then rejoin with the next text line. This means you can insert text into the middle of a paragraph from a fixed mode without losing existing test off the end of the right margin, something no other version of TI Writer will allow.

--Typing a number in a blank command line followed by <enter> will put that line at the top of the screen and return to edit mode (S before line number not necessary). <Enter> from a blank command line returns to edit mode (E prior to <enter> not necessary).

--You can freeze the display beginning with the line below the cursor while continuing to scroll, window, and edit from the cursor line to the top of the screen. This means you can simultaneously display two parts of the edit buffer with full editing capabilities for one of these displayed parts.

--You can put a bookmark (mark the text) at any line number from either command mode or edit mode. Later you can put the cursor on this text with FCTN/= even if the text has been edited since marking.

--You can display the contents of any hard drive path from the command line similar to doing a SD. Enter "HD" from the command line, then type a path name and press enter. The resulting display of that directory's file names resembles the SD display, and you can mark DV80 files for loading into the editor. This should be great for hard drive users who have trouble cataloging their hard drives with existing software.

--From SD or HD two diffent files can be marked, the regular and "temporary" file. These can be loaded into the editor with LF (regular) and LT (loads the "temporary" file).

--A user definable wild card character can be used the string searches with FS and RS.

--The SD display shows the number of bytes remaining in the edit buffer.

--When you load, print, or save files an incrementing number in the upper right of the screen shows the current line being loaded into or out of memory.

CONFIGURING THE EDITOR --
WHICH v5 EDITOR FILES ARE REQUIRED?

Just the 80 column editor, without other parts of Funnelweb, comes on an almost full unarchived 2 SSSD disk set. There are lots of files, mostly foreign char sets and command line text.

If you want only word processing in English then put ED, EE, and HELP00 10 20 and 30 on your Funnelweb v4.4 working disk or directory. If you want to play around with All Chars graphics add file CHAR@1 to this list.

You should use INSTALL/ED and a modified CONFIG/ED to configure the Print File "P" and "Q" printer codes into your ED file, but these two files don't have to be kept on the Funnelweb work disk. Funnelweb's v5 editor can be configured with CONFIG/ED to immediately boot to any of the available languages, or to boot to the powerup menu selections. No matter how the editor is configured, if you hold down the space bar as the editor boots you will get the powerup menu selections.

If you want use the full multilingual capabilities of the v5 editor then you need all the CHARxx and F8TXxx files on your Funnelweb work disk.

The various utilities for configuring the editor, making your own help screens, making your own character sets, etc have an undocumented feature. They respond to FCTN/7 (AID) by invoking the Quick Directory if files QD and QF from Funnelweb v4.4 are on the Funnelweb work disk.
+++

PROGRAMMING MUSIC THE EASY WAY

Part 3

by Jim Peterson

PROGRAMMING MUSC
THE EASY WAY
PART 3
BY JIM PETERSON

In Part 1 of this series, I showed you the simple routine to set up a musical scale, and showed you how easy it was to merge in various routines to create different effects in single-note music. In Part 2 I showed you how to key in single-note melodies from sheet music. Now, we will get into 3-part harmony.
But first, there are a few more things I should have told you about reading music. You will often see curved lines arching over two or more notes. If the notes are not all the same, ignore those lines - they call for phrasing which you cannot really accomplish. But, if the line curves over two or three of the same note, you will get a better effect if you add all their duration values together and program them as a single note. For instance, if your chart gives a whole note a value of 8 and a half-note a value of 4, and the music has a curved line over a whole note followed by a half-note, just program one note with a duration of 12.

You may find a heavy black bar at the beginning of a measure, with a colon to its right, and somewhere later in the music will be a heavy bar with a colon at its left. This means that the notes between those bars are to be played through twice - and naturally you will want to save time by programming them in a GOSUB as I showed you in Part 2. It can get more complicated than that, but generally you can follow the lyrics to decipher what to do.

Rather rarely, you may find three notes, usually joined together, with a 3 above them. These are called a triplet, and all three of them are to be played, with the same duration for each, in the length of time it would normally take to play one of them. These can create a problem under any method of music programming. The best method is to divide the duration of the note by three and write individual CALL SOUNDs in your music, rather than a GOSUB to a routine, to handle those notes.

Now, let's get on to 3-part harmony. It is just the same as keying in single note music, except that you must also give frequency values to B and C - and, as before, you have to give those values only when they change.
So, load the SCALE routine from the first lesson, and key in this bit of music to experiment with. Notice that I found three repeating phrases and put them in subroutines in 500, 600 and 700 to make this shorter.

```
110 GOSUB 500 :: T=4 :: A=15
 :: B=11 :: C=9 :: GOSUB 100
0 :: T=8 :: A=18 :: GOSUB 10
00 :: T=2 :: A,B,C=0 :: GOSU
B 1000 :: T=2 :: A=23 :: B=1
8 :: C=15 :: GOSUB 1000 :: G
OSUB 600
120 T=2 :: A=21 :: B=18 :: C
=15 :: GOSUB 1000 :: A=23 ::
 GOSUB 1000 :: T=12 :: A=20
:: B=16 :: C=11 :: GOSUB 100
0
130 T=2 :: A,B,C=0 :: GOSUB
1000 :: GOSUB 500 :: T=4 ::
A=21 :: B=16 :: C=13 :: GOSU
B 1000 :: T=10 :: A=25 :: GO
SUB 1000
140 T=2 :: A=28 :: GOSUB 100
0 :: GOSUB 600
150 T=2 :: A=27 :: B=23 :: C
=18 :: GOSUB 1000 :: A=30 ::
 GOSUB 1000 :: T=10 :: A=28
:: B=23 :: C=20 :: GOSUB 100
0
160 T=2 :: A,B,C=0 :: GOSUB
1000 :: T=3 :: A=28 :: B=23
:: C=20 :: GOSUB 1000 :: T=1
 :: A=27 :: GOSUB 1000 :: GO
SUB 700
170 T=6 :: A=25 :: B=21 :: C
=9 :: GOSUB 1000 :: T=2 :: A
=23 :: B=18 :: C=15 :: GOSUB
 1000
180 T=10 :: A=20 :: B=16 ::
C=11 :: GOSUB 1000 :: T=2 ::
 A,B,C=0 :: GOSUB 1000
190 T=3 :: A=28 :: B=23 :: C
=20 :: GOSUB 1000 :: T=1 ::
A=27 :: GOSUB 1000 :: GOSUB
700
200 T=4 :: A=25 :: B=21 :: C
=16 :: GOSUB 1000 :: A=21 ::
 B=18 :: C=15 :: GOSUB 1000
210 T=14 :: A=20 :: B=16 ::
C=11 :: GOSUB 1000 :: T=2 ::
 A,B,C=0 :: GOSUB 1000 :: ST
OP
500 T=2 :: A=23 :: B=20 :: C
=16 :: GOSUB 1000 :: A=28 ::
 GOSUB 1000 :: A=27 :: GOSUB
 1000 :: A=28 :: GOSUB 1000
:: A=27 :: GOSUB 1000
510 A=28 :: GOSUB 1000 :: A=
23 :: B=20 :: C=16 :: GOSUB
1000 :: A=20 :: B=16 :: C=11
 :: GOSUB 1000 :: A=16 :: B=
11 :: C=8 :: GOSUB 1000 :: R
ETURN
600 T=2 :: A=27 :: B=23 :: C
=18 :: GOSUB 1000 :: A=23 ::
 B=18 :: C=15 :: GOSUB 1000
:: A=21 :: GOSUB 1000 :: A=2
3 :: GOSUB 1000
610 A=27 :: GOSUB 1000 :: A=
23 :: GOSUB 1000 :: RETURN
700 T=4 :: A=27 :: B=21 :: C
=16 :: GOSUB 1000 :: T=8 ::
A=25 :: GOSUB 1000 :: T=3 ::
 A=27 :: B=23 :: C=18 :: GOS
UB 1000
710 T=1 :: A=21 :: GOSUB 100
0 :: T=4 :: A=25 :: B=21 ::
C=16 :: GOSUB 1000 :: T=8 ::
 A=23 :: B=20 :: C=16 :: GOS
UB 1000
720 T=3 :: A=25 :: B=21 :: C
=16 :: GOSUB 1000 :: T=1 ::
A=23 :: GOSUB 1000 :: T=2 ::
 A=23 :: B=18 :: C=15 :: GOS
UB 1000
730 A=21 :: GOSUB 1000 :: A=
20 :: GOSUB 1000 :: A=21 ::
GOSUB 1000 :: RETURN
```

Save that under the filename ROSES, clear the memory with NEW, and key this in -

```
1000 CALL SOUND(D*T,N(A),V1, N(B),V2,N(C),V3):: RETURN
```

Save that by SAVE DSK1.PLAIN3,MERGE .

Load ROSES again and merge it in by MERGE DSK1.PLAIN3 .
Add a line -
105 D=200 and RUN it.

Sounds rather raw and harsh, doesn't it? Try changing that line 105 to -
105 D=200 :: V2=5 :: V3=8

Try it again. Sound better? The first time, all 3 voices were being played at the loudest volume. Usually computer music will sound better if the harmony notes are given a lower volume. Experiment and find the volumes you like best. Is the music too slow for you? Just change the value of D. Is it not in your singing key? Just change the value of F in line 100, as I showed you before.

But, does the music still have too strong a beat for your taste? Clear the memory again and key this in -

```
1000 CALL SOUND(-4250,N(A+Z)
,V1,N(B+Z),V2,N(C+Z),V3):: G
OSUB 1010 :: RETURN
1010 FOR W=1 TO T*D :: NEXT
W :: RETURN
```

Save that as NEG3,MERGE because it uses negative duration for 3 voices. Then load ROSES again and merge it in. This time, try line 105 with D=50 and with V2 and V3 as you wish. Sound smoother?

In lines 110, 130, 160, 180 and 210 of ROSES, you will find A,B,C=0 . That makes all three voices silent, because in line 100 N(0) is given a frequency of 40000 which is above the range of human hearing. This is how I programmed those silent pauses, the "rests" which were written in the music.
On a piano or guitar, the strings continue to vibrate during a rest, so that the sound gradually fades out. However, the electronically generated tones of a computer stop very suddenly. That is why I often add the duration of the rest to the duration of the preceding note, and play it right on through. Some people think that doesn't sound right, so here is another solution. Clear memory again and key this in -

```
2000 FOR W=2 TO 8 STEP 8 ::
CALL SOUND(-999,N(A+Z),V1+W,
N(B+Z),V2+W,N(C+Z),V3+W):: G
OSUB 2010 :: NEXT W :: RETUR
N
2010 FOR Y=1 TO T*D/4 :: NEX
T Y :: RETURN
```

Save that as REST,MERGE. Load ROSES again, merge in SCALE and NEG3 (this will not work well with PLAIN3) and merge in REST. Now go to lines 110, 130, 160, 180 and 210, delete the A,B,C=0 :: and change the GOSUB 1000 after it to GOSUB 2000. Add line 105, run it and see if you like that better.
Anyway, keep it for now because we will use it again.

You will probably want to have the music play through more than once. Just add :: FOR J=1 TO 4 to the end of line 105 (if you want it to play 4 times) and change the end of line 210 to read NEXT J :: STOP .

I said that you could change the key of the music just by changing the value of F in line 100. There is also a way to change it while the music is playing. After the FOR J=1 TO 4 in 105 put ::
Z=Z-(J=2)*3-(J=3)*1+(J=4)*4
That is somewhat complicated but it just means to play the second time three whole keys higher, the third time one key higher still (I know the *1 is unnecessary!) and drop back 4 keys for the 4th time, so you can take it from there and modify it as you wish. If you want to use that routine with silent rests, change the GOSUB after each rest to 3000 instead of 1000, and add this line -

```
3000 CALL SOUND(-4250,N(A),V
1,N(B),V2,N(C),V3):: GOSUB 1
010 :: RETURN
```

This tune happens to end in a rest, which is unusual. If you key in another tune and it seems to end too abruptly, just after that NEXT J and before the STOP, put in a long duration such as T=12 and a GOSUB 2000 to that REST routine to fade out more slowly.

Now, when you are keying in your own tunes, the notes on your sheet music will usually have two or three of those little eggs on the stem. It is best to use the upper one for A, the next one for B, and the lower one for C; the computer could care less, but you will find it easier to keep track of what you are doing.
If there are less than three, just go directly below to the bass clef and find a note there. If you still don't have enough, you can always use 0 to make that voice silent. Or, you can usually just let the previous note continue. If your sheet music has guitar chords - those little square grids with dots on them - above the staff, they will give you some help - if there is no guitar chord above the note you are working on, the chord has not changed and it is safe to use the previous harmony notes.

There are many other CALL SOUND routines you can use for different effects. This is similar to the one that Bill Knecht used for his hymns - I call it VIBRA.

```
105 D=1 :: V1=1 :: V2=5 :: V
3=11
1000 FOR J=1 TO T*D :: CALL
SOUND(-99,N(A),V1,N(B),V2,N(
C),V3):: CALL SOUND(-99,N(A)
*1.01,V1,N(B),V2,N(C),V3)::
NEXT J :: RETURN
```

This one I call WUBBA, for no good reason -

```
105 D=1 :: V1=1 :: V2=5 :: V
3=11
1000 FOR J=1 TO T*D :: CALL
SOUND(-99,N(A),V1,N(B),V2,N(
C),V3):: CALL SOUND(-99,N(A)
*1.01,V1,N(B),V3,N(C),V2)::
NEXT J :: RETURN
```

And this one I call TREM -

```
105 D=1 :: V1=1 :: V2=5 :: V
3=11
1000 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V2,N(B),V2,N
(C)*1.01,V3):: CALL SOUND(-9
99,N(A)*1.01,V1,N(B),V2,N(C)
,V3):: NEXT J :: RETURN
```

I included line 105 in those, to merge in the duration and volumes along with the sound routine. Change the value of D to suit yourself, even in decimal increments such as D=1.5 .

It is easy to play a song repeatedly but with a different effect each time. Merge in VIBRA and change its line number to 1010. You can do this by typing 1000 and FCTN X, Enter, FCTN 8 to bring it back, type over the line number, and Enter. Merge in WUBBA and change it to line 1020 in the same way, then TREM and change it to line 1030.

Add :: FOR R=1 TO 3 to the end of line 105. Put in a new line 1000 -

```
1000 ON R GOSUB 1010,1020,1030 :: RETURN
```

And change the end of line 210 to

```
NEXT R :: STOP.
```

Next time - more different effects, and autochording.
++

TIPS FROM THE TIGERCUB

NO #10

Tigercub Software
156 Collingwood Ave.
Columbus, OH 43213
(614)235-3545

TIPS #9's challenge was to write a one-line program in Extended Basic which would take only 70 seconds to scramble the numbers from 1 to 255 into a completely random sequence without duplication. Well, you do have to shove it, but:

```
100 FOR J=1 TO 255 :: M$=M$&
CHR$(J):: NEXT J :: DIM N(25
5):: RANDOMIZE :: FOR J=255
TO 1 STEP -1 :: X=INT(J*RND+
1):: N(J)=ASC(SEG$(M$,X,1)):
: M$=SEG$(M$,1,X-1)&SEG$(M$,
X+1,LEN(M$)):: NEXT J
```

The challenge this time - can you unfurl the U.S. flag (49 stars), from the mast out, in 2 lines of Extended Basic?

A tip for you disk drivers flip and back up! Disk sectors don't go bad very often, but it does happen, and the program or file that you lose is usually the one that you can't replace.
So it pays to make a backup, but then you need twice as many disks, and disks do cost Some folks say that a flipped disk is more likely to go bad, others don't think so, but anyway it doesn't happen very often.
So if you back up your program on the flip side of another disk, the chances of both going bad are almost nil, and it hasn't cost you a penny.

And don't spend $20 for one of those "Disk Flipping kits". They consist of a template and a paper punch You can make a template for nothing from the cover of an old disk that's gone bad.And the paper punch you can buy in an office supply store for about a dollar - try to find one that has a plastic protector on the lower jaw to catch the punchings and to protect the disk. A square write-protect notch is not necessary, a half round one works just as well A bottle of typist's "white-out" is handy for making the spots to be punched.

Do you want that "arcade effect" in your musical tones or single-note music? Instead of:
CALL SOUND(D,N,V) write
CALL SOUND(D,N,V,N*1.01,V)

Sombdy actually wrote me a letter and said that they liked the Cryptocoder in the last Tips. So, since Word-search puzzles are so popular as space-fillers in the newsletters...

```
100 CALL CLEAR
110 REM - programmed by Jim
Peterson of Tigercub Softwar
e, 156 Collingwood Ave., Col
umbus, Ohio 43213
120 PRINT " TIGERCUB WORDSEA
RCH MAKER": : :"Make your ow
n wordsearch":"puzzles, Use
the arrow keys"
130 PRINT "and W,R,Z, and C
keys to move":"the asterisk
around. Hold":"down the CTRL
 key when you"
140 PRINT "want to type a le
tter. When":"you have finish
ed putting in":"words, press
 ENTER and the"
150 PRINT "computer will fin
ish the":"puzzle. Then if yo
u want to":"save it on tape
or disk,":"press P.": : : :
160 PRINT : :"Press any key
to start."
170 CALL KEY(0,K,ST)
180 IF ST=0 THEN 170
190 CALL CLEAR
200 DIM L$(24)
```

2022 note: Now we are trying to access the old TI disks it is apparent- the flippies have gone bad- on both sides- far more often....

```
210 R=12
220 C=16
230 CALL HCHAR(R,C,42)
240 FOR J=5 TO 12
250 CALL HCHAR(J,J,ASC(SEG$(
"TIGERCUB",J-4,1)))
260 NEXT J
270 CALL KEY(5,K,ST)
280 IF ST<1 THEN 270
290 ON POS("WERSDZXC"&CHR$(1
3),CHR$(K),1)+1 GOTO 430,300
,340,330,310,370,390,400,360
,460
300 R=R-1-(R=1)
310 C=C-1-(C=3)
320 GOTO 410
330 C=C+1+(C=30)
340 R=R-1-(R=1)
350 GOTO 410
360 R=R+1+(R=24)
370 C=C+1+(C=30)
380 GOTO 410
390 C=C-1-(C=3)
400 R=R+1+(R=24)
410 CALL HCHAR(R,C,42)
420 GOTO 270
430 IF K<129 THEN 270
440 CALL HCHAR(R,C,K-64)
450 GOTO 270
460 FOR R=1 TO 24
470 FOR C=3 TO 30
480 CALL GCHAR(R,C,G)
490 IF (G<>42)*(G<>32)THEN 5
50
500 RANDOMIZE
510 CH=INT(26*RND+65)
520 L$(R)=L$(R)&CHR$(CH)
530 CALL HCHAR(R,C,CH)
540 GOTO 560
550 L$(R)=L$(R)&CHR$(G)
560 NEXT C
570 NEXT R
580 CALL KEY(3,K,ST)
590 IF (ST=1)+(K<>80)THEN 58
0
600 PRINT "SAVE TO:":"(C) CA
SETTE?":"(D) DISK?",
610 INPUT Q$
620 IF Q$<>"D" THEN 660
630 INPUT "NAME OF FILE? ":Q
F$
640 F$="DSK1."&QF$
650 GOTO 680
660 IF Q$<>"C" THEN 610
670 F$="CS1"
680 OPEN #1:F$,INTERNAL,OUTP
UT,FIXED 192
690 FOR J=1 TO 24
700 IF INT(J/6)*6=J THEN 730
710 PRINT #1:L$(J),
720 GOTO 740
730 PRINT #1:L$(J)
740 NEXT J
750 CLOSE #1
```

Of course, if you're one of those lucky folks with a printer......

Here's another musicmaker for you....

```
100 REM TIGERCUB COMPOSER
110 RANDOMIZE
120 DEF A=VAL(SEG$(T$,INT(3*
RND+1) *3-2,3))
130 FOR P=1 TO 4
140 ON P GOSUB 200,220,240,2
00
150 FOR J=1 TO 4
160 CALL SOUND(-999,A,5,A*2,
0,A*4,10)
170 NEXT J
180 NEXT P
190 GOTO 130
200 T$="131165196"
210 RETURN
220 T$="123147196"
230 RETURN
240 T$="110131175"
250 RETURN
```

TIPS FROM THE TIGERCUB
NO #11

The above challenge was to unfurl the U.S.flag (with 49 stars), from the mast out in 2 lines of Extended Basic

```
100 CALL CLEAR :: CALL COLOR
(2,16,5,3,16,16,4,7,7):: A$(
1)="*******080808" :: A$(2)=
RPT$("80",7):: CALL CHAR(33,
RPT$("01",8)):: CALL VCHAR(4
,4,33,20)
110 FOR C=5 TO 22 :: X=1+ABS
(C>11):: FOR T=1 TO 13 :: CA
LL VCHAR(5+T,C,ASC(SEG$(A$(X
),T,1))):: NEXT T :: NEXT C
:: GOTO 110
```

One of the previous challenges was to write the Extended Basic statement IF X=1 THEN Y=7 ELSE IF X=2 THEN Y=33 ELSE IF X=3 THEN Y=19 ELSE IF X=4 THEN Y=21.My solutin was Y=VAL(SEG$("07331921",X*2-1,2).

Jim John- ston in the K-3

User's Group newsletter came up with a method which is better because it does not require that the values of X be in sequence:

```
Y=ABS((7*(X=1))+(33*(X=2))+(
19*(X=3))+(21*(X=4)))
```

Proving once again that there is more than one way to skin the cat, and often a better way - although the cat might not agree.

Advice to disk-drivers - keep and eye on those little tabs of silver tape that you use to cover the write-protect notch on your disks. They tend to become dog-eared from bumping against the slot of the drive. I recently heard a horror story about one of those tabs that came loose and got into the drive!

The following Menu-loader or Auto-booter was originally published by A. Kludge in the 99'er Vol. 1 #4. Marshall Gordon and Thomas Boisseau greatly improved it and published it in the Atlantic 99/4 UG newsletter Vol. 2 #1.
I have no idea how it works, but have managed to modify it so that it will catalog up to 99 programs on a disk, stopping for input after each 19 are listed, or stopping whenever any key is pressed; I also added a delete option, requiring a repeated input to prevent error.

It takes up only 8 sectors. If you have Extended Basic and Disk Drive, load this program under the file name LOAD. It will then automatically run whenever you select Extended Basic, will list all the programs on the disk, and will run whichever program you select.

(This program was successively updated in Tips #15, 18, 20, 22, 23, 24, 27, 28, 29, 30 and 32. - Ed.)

```
100 OPTION BASE 1 :: DIM PG$
(99),T$(5):: CALL CLEAR
110 T$(1)="DIS/FIX" :: T$(2)
="DIS/VAR" :: T$(3)="INT/FIX
" :: T$(4)="INT/VAR" :: T$(5
)="PROGRAM"
120 IMAGE ##
130 DISPLAY AT(1,9)ERASE ALL
:"DISKETTE MENU"
140 ! IF YOU HAVE MORE THAN
ONE DISK DRIVE, DELETE THE !
IN LINE 150
150 ! DISPLAY AT(12,6):"DISK
? (1-3):" :: ACCEPT AT(12,19
)SIZE(-1)VALIDATE("123"):D$
:: D$="DSK"&D$&"."
160 D$="DSK1." :: OPEN #1:D$
,INPUT ,RELATIVE,INTERNAL ::
 INPUT #1:N$,A,J,K :: DISPLA
Y AT(1,1)ERASE ALL:SEG$(D$,1
,4)&" - DISKNAME= "&N$:
170 DISPLAY AT(2,1):"AVAILAB
LE=";K;"USED=";J-K:"PROG  FI
LENAME  SIZE    TYPE":"_____
_____________________" ::
I=0
180 FOR X=1 TO 80 :: IF X/20
<>INT(X/20)THEN 210
190 DISPLAY AT(24,1):"TYPE C
HOICE OR 99 FOR MORE" :: ACC
EPT AT(24,27)VALIDATE(DIGIT)
:K :: IF K=99 THEN 200 :: IF
 K>0 AND K<X+1 THEN 360 ELSE
 190
200 X=X+1 :: CALL VCHAR(1,2,
32,48)
210 I=I+1 :: IF I>127 THEN K
=X :: GOTO 300
220 INPUT #1:P$,A,J,B
230 IF LEN(P$)=0 THEN 270
240 DISPLAY AT(X+4,2):USING
120:X :: DISPLAY AT(X+4,6):P
$ :: PG$(X)=P$ :: DISPLAY AT
(X+4,18):USING 120:J :: DISP
LAY AT(X+4,22):T$(ABS(A))
250 CALL KEY(0,KK,ST):: IF S
T=0 THEN 260 :: FLAG=1 :: GO
TO 280
260 NEXT X
270 DISPLAY AT(X+4,1):" " ::
 DISPLAY AT(X+4,2):USING 120
:X :: DISPLAY AT(X+4,6):"TER
MINATE" :: DISPLAY AT(X+5,2)
```

```
:STR$(X+1)&" DELETE?"
280 DISPLAY AT(X+6,1):"    C
HOICE"
290 ACCEPT AT(X+6,16)SIZE(2)
VALIDATE(DIGIT):K :: IF K<>X
 AND K<>X+1 OR FLAG=1 THEN 3
50
300 IF K=X THEN CALL CLEAR :
: CLOSE #1 :: END
310 DISPLAY AT(X+5,11)SIZE(1
8):" #?" :: ACCEPT AT(X+5,15
)SIZE(2)VALIDATE(DIGIT):KD :
: IF KD<1 OR KD>X-1 THEN 310
320 DISPLAY AT(X+6,1)SIZE(28
)BEEP:"VERIFY - REPEAT DELET
E #" :: ACCEPT AT(X+6,27)SIZ
E(2)VALIDATE(DIGIT):KD2 :: I
F KD2<>KD THEN 340
330 DELETE "DSK1."&PG$(KD)
340 CLOSE #1 :: GOTO 130
350 IF K<1 OR K>99 OR LEN(PG
$(K))=0 THEN 270
360 CLOSE #1
370 CALL INIT :: CALL PEEK(-
31952,A,B):: CALL PEEK(A*256
+B-65534,A,B):: C=A*256+B-65
534 :: A$=D$&PG$(K):: CALL L
OAD(C,LEN(A$))
380 FOR I=1 TO LEN(A$):: CAL
L LOAD(C+1,ASC(SEG$(A$,1,1))
):: NEXT I :: CALL LOAD(C+1,
0)
390 RUN "DSKX.1234567890"
```

Come to think of it, if you have more than one disk drive you will also have to delete the first statement in line 160, and modify line 330.

Here's a memory-saver for you - put your data in strings instead of data statements. My "Hangman Plus" program was only 7764 bytes long but it contained a vocabulary of 315 words in data statements. After reading these into an array, it had too little working memory left, and paused too often for garbage collection

After changing all the DATA statements to strings, it runs without stalling even though the number of words was increased and an array of 50 is still dimensioned for user input of words.

When I loaded the original version in Extended Basic with the Memory Expansion and asked for SIZE after the DATA had been read in, I found that I had 14756 bytes of program and 7669 bytes of stack free.

In the version with DATA in strings, at the same stage in the program I had 14874 bytes of program and 11310 bytes of stack free P a saving of 3730 bytes!

And another advantage is that there is no delay waiting for all those words to be read into the array. However, pulling DATA out of a string is undoubtedly a bit slower, so this method should not be used when speed is of primary importance.

In the "Hangman Plus"program, I used lower case letters as dividers between the upper case words. To pull words at random, I randomly selected a string and a position within the string, using the POS of the lower case letter to find the word. The following is a much abbreviated example:

```
100 M$(1)="aJOHNbJOEcCHARLIE
dMIKEeLARRYf"
110 M$(2)="aGEORGEbPETEcCHRI
SdDONeRALPHf"
120 X=INT(2*RND+1)
130 Y=INT(5*RND+97)
140 X$=SEG$)M$(X),POS(M$(X),
CHR$(Y),1)+1,POS(M$(X),CHR$(
Y+1),1)-POS(M$(X),CHR$(Y),1)
-1)
```

It is of course essential that all the strings contain the same number of elements of DATA. If lower case letters are needed, the separators can be ASCII codes 129 thru 154, obtained by holding down the CTRL key while typing the alpha-

bet -
 It's a bit hard to keep track of those, because they're invisible! Numeric DATA can be also be stored, using the VAL function to convert it to numeric after it is pulled from the string.

You probably already know this, but you don't have to type in the blank spaces before and after the :: in multiple statements in Extended Basic. Just run every thing together 100 CALL CLEAR::RANDOMIZE::FOR D=1 TO 100::NEXT D and the computer will separate it for you, shoving statements into additional lines if necessary.

OUT OF MEMORY

HAPPY HACKIN'
Jim Peterson

TIPS FROM THE TIGERCUB #31

Copyright 1986

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

 A few people have asked for a program that they could use to encode personal messages-here is a coder/decoder to create code that should be quite difficult to crack. First we need another of those programs that write a program -

```
100 !CODEPRINT by Jim Peters
on - creates a random code i
n a MERGE format program COD
ESTRING to be MERGEd into CO
DEMAKER
110 FOR J=1 TO 254 :: N$=N$&
CHR$(J):: NEXT J
120 FOR J=1 TO 254 :: RANDOM
IZE :: X=INT(RND*LEN(N$)+1):
: C$=C$&SEG$(N$,X,1):: N$=SE
G$(N$,1,X-1)&SEG$(N$,X+1,LEN
(N$)):: NEXT J
130 OPEN #1:"DSK1.CODESTRING
",VARIABLE 163,OUTPUT :: PRI
NT #1:CHR$(0)&CHR$(1)&"C$"&C
HR$(190)&CHR$(199)&CHR$(127)
&SEG$(C$,1,127)&CHR$(0)
140 PRINT #1:CHR$(0)&CHR$(2)
&"C2$"&CHR$(190)&CHR$(199)&C
HR$(127)&SEG$(C$,128,127)&CH
R$(0)
150 PRINT #1:CHR$(0)&CHR$(3)
&"C$"&CHR$(190)&"C$"&CHR$(18
4)&"C2$"&CHR$(0):: PRINT #1:
CHR$(255)&CHR$(255):: CLOSE
#1 :: END
```

And now the coder/decoder -

```
100 !TIGERCUB CODEMAKER writ
ten by Jim Peterson
110 !The MERGE format progra
m CODESTRING created by the
program CODEPRINT must be ME
RGEd into lines 1-3 of this
program
120 DIM A$(254):: DISPLAY AT
(3,6)ERASE ALL:"TIGERCUB COD
EMAKER" :: DISPLAY AT(12,1):
"Do you want to": :"(1)Encod
e":"(2)Decode"
130 CALL KEY(0,K,ST):: IF K=
49 THEN 140 ELSE IF K=50 THE
N 290 ELSE 130
140 OPEN #1:"DSK1.CODE",VARI
ABLE 254,OUTPUT
150 DISPLAY AT(5,6)ERASE ALL
:"Type message in segments o
f":"not more than 254 charac
ters":"and Enter. When done,
 type"
160 DISPLAY AT(9,1):"END and
 Enter. Type slowly":"to avo
id skipped characters.":"Bac
kspace with FCTN S to":"corr
ect.": :"Press any key"
170 CALL KEY(0,K,ST):: IF ST
=0 THEN 170
180 CALL CLEAR :: CALL LONGA
CCEPT(0,M$):: IF M$="END" TH
EN 280
190 DISPLAY AT(20,1):"WAIT,
PLEASE - ENCODING"
200 FOR J=1 TO LEN(M$)
210 A$(ASC(SEG$(C$,J,1)))=SE
G$(M$,J,1)
220 NEXT J
230 FOR J=1 TO 254 :: RANDOM
IZE
240 IF A$(J)="" THEN A$(J)=C
HR$(INT(26*RND+65))
250 CODE$=CODE$&A$(J)
260 NEXT J :: PRINT CODE$
270 PRINT #1:CODE$ :: CODE$=
"" :: FOR J=1 TO 254 :: A$(J
)="" :: NEXT J :: GOTO 180
280 CLOSE #1 :: END
290 OPEN #1:"DSK1.CODE",VARI
ABLE 254,INPUT :: CALL CLEAR
 :: DISPLAY AT(12,10):"DECOD
ING"
300 LINPUT #1:CODE$ :: FOR J
=1 TO 254 :: M$=M$&SEG$(CODE
$,ASC(SEG$(C$,J,1)),1):: NEX
T J :: PRINT M$;:: M$=""
310 IF EOF(1)<>1 THEN 300 ::
 CLOSE #1 :: END
320 SUB LONGACCEPT(L,M$):: X
=0 :: IF L<>0 THEN R=L ELSE
R=R+1
330 M$="" :: C=3 :: CH=140 :
: CALL CHAR(140,RPT$("0",14)
&"FF")
340 CALL HCHAR(R,C,CH):: CH=
CH+5+(CH=160)*25 :: CALL KEY
(0,K,ST):: IF ST<1 THEN 340
350 IF K<>8 THEN 370 :: X=X-
1 :: C=C-1 :: IF C=2 THEN C=
30 :: R=R-1
360 M$=SEG$(M$,1,LEN(M$)-1):
: GOTO 340
370 IF K=13 THEN 410
380 X=X+1 :: M$=M$&CHR$(K)::
 CALL HCHAR(R,C,K):: IF X=25
4 THEN 410
390 C=C+1 :: IF C=31 THEN C=
3 :: R=R+1 :: IF R=25 THEN C
ALL CLEAR :: R=1
400 GOTO 340
410 R=0 :: SUBEND
```

============

Here is a simple little game I call Cover-up. Use the #1 joystick, try to cover the white square with the black square. Press the fire button to speed up, release it to slow down.

```
100 CALL CLEAR :: CALL CHAR(
96,RPT$("F",64)):: CALL SPRI
TE(#1,96,5,92,124):: CALL MA
GNIFY(4):: CALL SPRITE(#2,96
,16,100,100)
110 X=INT(20*RND)-INT(20*RND
):: Y=INT(20*RND)-INT(20*RND
):: CALL MOTION(#2,X,Y):: T=
T+1 :: IF T=250 THEN 300
120 CALL JOYSPEED(1,1):: CAL
L COINC(#1,#2,8,A):: IF A=-1
 THEN 130 ELSE 110
130 Z=Z+1 :: DISPLAY AT(1,1)
:Z :: CALL SOUND(-50,500,5):
: GOTO 120
300 CALL DELSPRITE(ALL):: DI
SPLAY AT(12,5):"YOUR SCORE I
S "&STR$(Z):: DISPLAY AT(20,
1):"PRESS ENTER TO PLAY AGAI
N"
310 CALL KEY(0,K,S):: IF S=0
 OR K<>13 THEN 310 :: T,Z=0
:: GOTO 100
21110 SUB JOYSPEED(N,A):: CA
LL JOYST(N,X,Y):: CALL KEY(N
,K,ST):: S=S+K/9-1 :: S=S*AB
S(S>0):: IF S>30 THEN S=30
21111 CALL MOTION(#A,-(Y*S),
X*S):: SUBEND
```

======

For a one-handed BREAK, if you can't reach FCTN and 4, try FCTN with J and the space bar together.

======

Probably useless info - holding down FCTN and CTRL together and typing 1, 2, 3 and 5 will give ASCII codes 145, 151, 133 and 148, which are the codes obtained from CTRL Q, W, E and T, the keys diagonally below the 1, 2, 3 and 5.

=========

Occasionally someone sends me a program they have keyed in from my newsletter, and asks why it won't run, so I wrote this routine to help find the errors. It is also useful to check whether two copies of a program are identical, but only if they have not been resequenced.

(the two programs must first be LISTED to one disk by LIST DSK1.(filename, using a different filename for each - Ed.)

```
100 !CHECKER by Jim Peterson
 - to compare two programs a
nd list all differing lines
to the printer
110 DISPLAY AT(12,1)ERASE AL
L:"1st program DSK/filename?
":"DSK" :: ACCEPT AT(13,4):F
1$
120 DISPLAY AT(12,1)ERASE AL
L:"2nd program DSK/filename?
":"DSK" :: ACCEPT AT(13,4):F
2$
130 OPEN #1:"DSK"&F1$,INPUT
:: DIM M$(500),CH(500):: OPE
N #2:"PIO",VARIABLE 255 :: P
```

```
RINT #2:CHR$(15)
140 X=X+1 :: LINPUT #1:M$(X)
:: M$(X)=M$(X)&" " :: IF EOF
(1)<>1 THEN 140 :: CLOSE #1
:: OPEN #1:"DSK"&F2$,INPUT
150 IF EOF(1)=1 THEN 230 ::
LINPUT #1:X$ :: X$=X$&" "
160 FOR Y=1 TO X
170 IF X$=M$(Y)THEN CH(Y)=1
:: GOTO 150
180 NEXT Y
190 P2=POS(X$," ",1):: P2$=S
EG$(X$,1,P2-1)
200 FOR Y=2 TO X :: P1=POS(M
$(Y)," ",1):: P1$=SEG$(M$(Y)
,1,P1-1)
210 IF P2$=P1$ THEN PRINT #2
:"1st program = ";M$(Y):"2nd
 program = ";X$ :: CH(Y)=1 :
: GOTO 150
220 NEXT Y :: PRINT #2:"2nd
program = ";X$ :: GOTO 150
230 FOR J=1 TO X :: IF CH(J)
=0 THEN PRINT #2:"1st progra
m = ";M$(J)
240 NEXT J
250 CLOSE #1 :: CLOSE #2
```

=============

Here's a poor idea that was printed and reprinted in several newsletters -
At the beginning of a program that will run only in Basic, add the lines -

```
1 IF PI=0 then (first line
of program)
2 PRINT "YOU ARE IN EXTENDED
BASIC":"THIS PROGRAM RUNS
ONLY IN BASIC"
3 STOP
```

The idea is that PI is a function in XBasic with the value of pi, but is just a variable name in Basic with an undefined value of 0.

The trouble is, it doesn't work! If PI is keyed in from Basic and saved, it is saved in token format as a variable name, and when loaded back into XBasic is still just a variable name. And if PI is saved from XBasic, it is tokenized as a function, loads back into Basic as an unrecognized function and crashes! Can anyone come up with a way around that?

==========

Here is a handy PEEK that hasn't been published as widely as most of them -

```
100 CALL INIT
110 CALL PEEK(8192,X)!Thanks
to Dale Loftis in the Orange
County UG newsletter!
120 PRINT X !If X=32 you are
in Extended Basic; if X=165
you are in Basic with the
Editor Assembler or
MiniMemory module inserted.
```

=========

And another 3-D sprite demo, just to make all the Apple polishers jealous. See if you can figure out how it works.

```
100 CALL CLEAR :: CALL SCREE
N(5):: CALL CHAR(100,RPT$("F
",64)):: CALL MAGNIFY(4):: F
OR S=5 TO 9 :: CALL COLOR(S,
16,1):: NEXT S
110 DISPLAY AT(3,3):"TIGERCU
B SPRITE SHUFFLE" !by Jim Pe
terson
120 DATA 70,116,2,75,121,7,6
9,124,11,78,115,16
130 FOR J=5 TO 8 :: READ P(J
,1),P(J,2),L(J):: CALL SPRIT
E(#J,100,L(J),P(J,1),P(J,2))
:: NEXT J :: W=45
140 DATA 5,6,7,8,8,5,6,7,7,8
,5,6,6,7,8,5
150 RESTORE 140 :: FOR Y=5 T
O 8 :: READ A,B,C,D
160 FOR J=1 TO W :: CALL LOC
ATE(#A,P(A,1)-J,P(A,2),#B,P(
B,1),P(B,2)-J,#C,P(C,1)+J,P(
C,2),#D,P(D,1),P(D,2)+J):: W
=90 :: NEXT J :: GOSUB 180
170 NEXT Y :: GOTO 150
180 FOR J=5 TO 7 :: CALL POS
ITION(#J,P(J+1,1),P(J+1,2)):
: NEXT J :: CALL POSITION(#8
,P(5,1),P(5,2))
190 T=L(8):: L(8)=L(7):: L(7
)=L(6):: L(6)=L(5):: L(5)=T
200 FOR J=5 TO 8 :: CALL SPR
ITE(#J-4,100,L(J),P(J,1),P(J
,2)):: NEXT J
210 FOR J=5 TO 8 :: CALL SPR
ITE(#J,100,L(J),P(J,1),P(J,2
)):: NEXT J :: CALL DELSPRIT
E(#1,#2,#3,#4):: RETURN
```

============

Do you need some really REAL BIG letters on the screen? Just type your letter at the beep.

```
100 DIM X$(96):: CALL CLEAR
```

```
:: FOR CH=33 TO 89 STEP 8 ::
 FOR A=0 TO 7 !REAL BIG LETT
ERS by Jim Peterson
110 CALL CHARPAT(CH+A,X$(CH+
A-32)):: CALL CHAR(CH+A,"0")
:: L$=L$&RPT$(CHR$(CH+A),3):
: NEXT A
120 FOR T=1 TO 3 :: R=R+1 ::
 DISPLAY AT(R,4):L$ :: NEXT
T :: L$="" :: NEXT CH
130 CH$(1)=RPT$("0",16):: CH
$(2)=RPT$("F",16)
140 CALL SOUND(100,500,0)
150 CALL KEY(0,CH,S):: IF S=
0 OR CH>96 THEN 150
160 CALL HEX_BIN(X$(CH-32),B
$):: FOR J=9 TO 64 :: CALL C
HAR(J+32,CH$(VAL(SEG$(B$,J,1
))+1))
170 NEXT J :: GOTO 140
180 SUB HEX_BIN(H$,B$):: HX$
="0123456789ABCDEF" :: BN$="
0000X0001X0010X0011X0100X010
1X0110X0111X1000X1001X1010X1
011X1100X1101X1110X1111"
190 FOR J=LEN(H$)TO 1 STEP -
1 :: X$=SEG$(H$,J,1)
200 X=POS(HX$,X$,1)-1 :: T$=
SEG$(BN$,X*5+1,4)&T$ :: NEXT
 J :: B$=T$ :: T$="" :: SUBE
ND
```

============

Thought for the day. The excuses for piracy are exactly the same as the excuses for shoplifting, but you probably won't have to tell them to the judge - in this world, at least.

TIPS FROM THE TIGERCUB
#32
Copyright 1986

I've found a bug in the Tigercub Menuloader V.#5 which won't let you print a disk catalog if the disk contains the maximum 127 files. This should fix it.

```
340 I=I+1 :: IF I>127 THEN K
=X :: GOTO 430
520 DISPLAY AT(X+5,12)SIZE(1
2):" #?" :: ACCEPT AT(X+5,15
)SIZE(3)VALIDATE(DIGIT):KD :
: IF KD<1 OR KD>NN THEN 520
```

I think that all program listings should be printed in 28-column format, exactly as they appear on the screen - it makes it so much easier to key them in without errors. I combined parts of two of my programs to make the following. It is written for the Gemini 10X but the lines of printer control codes are annotated to help others make adjustments.

```
100 DIM K$(240):: LN=100 ::
DISPLAY AT(3,4)ERASE ALL:"TI
GERCUB PROGLISTER": :" Will
convert a program":"listing
to 28-column format,"
110 DISPLAY AT(7,1):"exactly
 as it appears on the":"scre
en, and print it in 4":"colu
mns."
120 DISPLAY AT(11,1):" Progr
am must be RESequenced":"and
 LISTed to disk by":"RES (en
ter)":"LIST DSK1.(filename)
(Enter)"
130 DISPLAY AT(18,1):"Filena
me? DSK" :: ACCEPT AT(18,14)
BEEP:F$
140 OPEN #1:"DSK"&F$,DISPLAY
 ,VARIABLE 80,INPUT
150 IF EOF(1)=1 THEN 260 ::
LINPUT #1:A$
160 IF LEN(A$)<80 THEN LN=LN
+10 :: GOTO 210
170 LINPUT #1:B$ :: IF POS(B
$,STR$(LN),1)=1 THEN FLAG=1
:: LN=LN+10 :: GOTO 210
180 A$=A$&B$ :: IF LEN(A$)<1
60 THEN LN=LN+10 :: GOTO 210
190 LINPUT #1:B$ :: IF POS(B
$,STR$(LN),1)=1 THEN FLAG=1
:: LN=LN+10 :: GOTO 210
200 A$=A$&B$ :: LN=LN+10
210 S=1
220 L$=SEG$(A$,S,28)
230 IF L$<>"" THEN 240 :: IF
 FLAG=1 THEN FLAG=0 :: A$=B$
 :: GOTO 160 :: ELSE GOTO 15
0
240 X=X+1 :: K$(X)=L$ :: S=S
+28 :: IF X=240 THEN 250 ::
GOTO 220
250 X=0 :: CALL PRINTER(K$()
):: GOTO 220
260 CLOSE #1 :: FOR J=X+1 TO
 240 :: K$(J)="" :: NEXT J :
: CALL PRINTER(K$()):: PRINT
 #2:CHR$(12):: END
270 SUB PRINTER(B$()):: IF F
=1 THEN 340 :: F=1
280 OPEN #2:"PIO.LF",VARIABL
```

```
E 132 :: PRINT #2:CHR$(15);C
HR$(27);"N";CHR$(6);!condens
ed print and perforation ski
p
290 PRINT #2:CHR$(27);"G";!
- double-struck printing, op
tional
300 PRINT #2:CHR$(27);CHR$(4
2);CHR$(0);!download normal
characters - required if lin
es 310-330 are used
310 PRINT #2:CHR$(27);CHR$(4
2);CHR$(1);CHR$(48);CHR$(0);
CHR$(64);CHR$(30);CHR$(96);C
HR$(17);CHR$(72);CHR$(5);CHR
$(66);CHR$(61);CHR$(0);!slas
h the zero - optional
320 PRINT #2:CHR$(27);CHR$(4
2);CHR$(1);CHR$(42);CHR$(0);
CHR$(8);CHR$(34);CHR$(8);CHR
$(0);CHR$(62);CHR$(0);CHR$(8
);CHR$(34);CHR$(8);!broaden
the asterisk - optional
330 PRINT #2:CHR$(27);CHR$(3
6);CHR$(1);!activate redefin
ed characters - required if
lines 310-320 are used
340 FOR C=1 TO 60 :: IF B$(C
)="" THEN 360 :: PRINT #2:TA
B(10);B$(C);TAB(41);B$(C+60)
;TAB(72);B$(C+120);TAB(103);
B$(C+180);CHR$(10)
350 NEXT C
360 SUBEND
```

I had trouble in debugging that program because printing the control codes gave me unwanted line feeds, and using semicolons to prevent line feeds will interfere with tabs in the first line of text. An article by Art Byers in the Central Westchester UG newsletter gave me the solution - suppress all the line feeds by opening the printer with PIO.LF, and put them back in where you need them with CHR$(10)!

================

We haven't had a random music player in a long time. This one is called ECHO but I don't know where it came from.

```
100 RANDOMIZE :: DEF X=INT(R
ND*7):: FOR B=0 TO 6 :: A(B)
=VAL(SEG$("24726229433034939
2440",(B+1)*3-2,3)):: NEXT B
:: B,C,D=X
110 CALL SOUND(-900,A(B),0,A
(C),9,A(D),19):: D=C :: C=B
:: B=X :: GOTO 110
```

========

Sound effects - thanks to Greg Healy in the Edmonton User Group newsletter -

```
100 CALL INIT
110 FOR J=2000 TO 2300 STEP
10 :: CALL LOAD(-31568,J)::
 NEXT J
```

=========

To go directly from XBasic to console Basic - thanks to Greg Healy in the Edmonton User Group newsletter -

```
CALL INIT :: CALL LOAD(-3196
2,8787)
```

Enter. Ignore the error message. Type NEW and Enter.

```
> TI BASIC READY
```

==============

This routine will read a file of 28-character records and scroll them up the lower half of the screen without disturbing the upper half. [the file is DV80 format]

```
100 DISPLAY AT(12,1)ERASE AL
L:"FILENAME? DSK" :: ACCEPT
AT(12,14)BEEP:F$ :: CALL CLE
AR
111 OPEN #1:"DSK"&F$,INPUT
112 DIM M$(480)
113 X=X+1 :: LINPUT #1:M$(X)
120 DISPLAY AT(24,1):M$(X)
125 R=24
130 FOR T=X-1 TO 1 STEP -1 :
: IF R>13 THEN R=R-1 :: DISP
LAY AT(R,1):M$(T)
140 NEXT T :: IF EOF(1)<>1 T
HEN 113 ELSE CLOSE #1
```

=========

A number always prints out with a blank space before and after it (except that a negative number is preceded by -). This is not always desirable when formatting a screen or printout. The solution is to change the number to a string by using STR$ -

```
100 CALL CLEAR
110 PRINT "  MULTIPLICATION
TABLES": :
120 FOR J=1 TO 9
130 FOR K=1 TO 9
```

```
140 PRINT TAB(K*3-2);STR$(J*
K);
150 NEXT K
160 PRINT : :
170 NEXT J
```
==========

Regarding the CHECKER program in Tips #31, I should have mentioned that the two programs to be compared must first be LISTed to one disk by -
LIST "DSK1.(filename)
- using a different filename for each.

================

Re TI*MES issue 37-
In Tips #26 I listed three algorithms to alternate between the two joysticks. Rick Humburg sent me another which is the simplest and fastest of all -
```
100 Z=2
110 Z=3-Z :: CALL JOYST(Z,X,
Y).......and back to 110!
```
==================

Here are some more dark secrets Texas Instruments didn't tell us. The User's Reference Guide claims that the computer can produce frequencies up to 44733 Hz, "well above human hearing limits", but then admits "the actual frequency produced may vary from 0 to 10 percent depending on the frequency." According to Jim Hindley, the highest frequency actually produced is 37287 (which is certainly not above the hearing range of some humans, but neither is 44733!), and the maximum error rate far exceeds 10 % because any frequency you call for from 31953 to 43733 ends up as exactly 37287! Not to worry, the frequencies in the normal range of music are accurate enough and your TV speaker probably can't reproduce frequencies above 20000 anyway.

And did you know that TI really gave us only 15 volumes, not 30? Listen and count them -
```
100 FOR V=0 TO 29 STEP 2
110 CALL SOUND(1000,500,V)
120  CALL SOUND(1000,500,V+1
1)
130 FOR D=1 TO 500
140 NEXT D
150 NEXT V
```

And the duration values are just as inaccurate. Experimenting with a series of 8 CALL SOUNDs in a loop repeated 100 times, I found that execution time was 40 seconds for any duration between 1 and 49, or a negative duration; 54 seconds for any duration between 50 and 66; 67 seconds between 67 and 83; 80 seconds between 84 and 99; 94 between 100-116; 106 between 117-133....!

==========

I guess I've been neglecting those who don't have the Extended Basic module, so -
```
100 CALL SCREEN(16)
110 CALL CLEAR
120 PRINT TAB(8);"GREENSLEEV
ES": : : : : : : : : : : : :
 :"programmed by Jim Peterso
n"
130 DIM S(15)
140 FOR N=1 TO 12
150 READ S(N)
160 NEXT N
170 M$="421800995ABDC324E7DB
A5186699182400425A00DBC35A66
A5243C7E81994200A57E66BD3CA5
423C187E423CBD5A810099FFC3"
180 RANDOMIZE
190 FOR R=1 TO 12
200 CALL COLOR(R+1,1,1)
210 CALL CHAR(32+R*8,CH$&CH$
)
220 FOR T=R TO 25-R
230 CALL HCHAR(T,R,32+R*8,34
-2*R)
240 NEXT T
250 NEXT R
260 CALL SCREEN(2)
270 FOR R=1 TO 12
280 CALL COLOR(R+1,R+2,1)
290 CH$=SEG$(M$,INT(47*RND+1
)*2-1,8)
300 CALL CHAR(32+R*8,CH$&CH$
)
310 NEXT R
320 DATA 247,277,294,311,330
,370,392,440,494,523,554,587
```

```
330 DATA 2,5,5,4,7,5,2,8,5,3
,9,5,1,10,1,2,9,3,4,8,3,2,6,
3,3,3,1,1,5,3
340 DATA 2,6,1,4,7,5,3,5,2,1
,4,2,2,5,2,4,6,1,2,4,4,4,1,1
350 DATA 2,5,1,4,7,5,2,8,5,3
,9,5,1,10,5,2,9,5
360 DATA 4,8,3,2,6,3,3,3,3,1
,5,3,2,6,3,3,7,5,1,6,2,2,5,1
370 DATA 3,4,1,1,2,2,2,4,1,4
,5,1,2,1,5,6,5,1
380 DATA 2,12,9,2,12,7,2,12,
3,3,12,12,1,11,9,2,9,7
390 DATA 4,8,6,2,6,3,3,3,3,1
,5,5,2,6,3,4,7,5,2,5,3
400 DATA 3,5,5,1,4,4,2,5,5,4
,6,1,2,4,1,6,1,1
410 DATA 6,12,9,3,9,12,1,11,
8,2,9,7,4,8,6,2,6,3,3,3,3
420 DATA 1,5,3,2,6,2,3,7,5,1
,6,6,2,5,5,3,4,1,1,2,2,2,4,4
,6,5,1,1,1,5,7,5,1
430 FOR J=1 TO 223 STEP 3
440 READ T,A,B
450 GOSUB 530
460 FOR TT=1 TO T
470 CALL SOUND(-999,S(A),0,S
(B),7)
480 NEXT TT
490 NEXT J
491 FOR V=0 TO 20
492 CALL SOUND(-999,S(A),V,S
(B),V+7)
493 NEXT V
500 CALL SCREEN(INT(14*RND+2
))
510 RESTORE 330
520 GOTO 270
530 CALL COLOR(A+1,INT(14*RN
D+2),1)
540 CALL COLOR(B+1,INT(14*RN
D+2),1)
550 RETURN
```

```
1 !from 9 T 9 UG newsl. Aug
85
100 PRINT """Hello"" said TI
"
110 PRINT "Press ""ENTER"" t
o continue"
```

===============

If you bite the hand that feeds you, you'll go hungry tomorrow. Don't be a pirate!

MEMORY FULL TO BUSTIN'

Jim Peterson

DISK LIBRARY REPORT.
NEW DISKS   JANUARY 1993

THE DRAWING MASTER is now updated to version 1.3.

I have a new version of Danny Michaels DUMP program modified by Bruce Harrison.

The library collection of PLATO data disks is now up to around 80 disks.
I wonder if anyone is writing programs these days? If you know of anything not in the disk library, why not let me know, and if you have it- donate a copy!

To reflect the lower cost of new disks, if you wish the library to supply a disk the cost is now just 50p per disk- so if you wish to order 8 disks, all supplied by the library, the cost is:

```
  8 SSSD sides copied at 1.00 each....8.00
  8 disks supplied by library @ 50p....4.00
Handling charge (post & packing).......1.00
                              total   13.00
```

The full disk library details are available on disk (DV80 text files)- just send 4 disks with return postage.
Disk Librarian:
Stephen Shaw, 10 Alstone Road, STOCKPORT, Cheshire, SK4 5AH

Copy for this issue was printed using a very old ribbon on single strike non-emphasised, the ribbon having been "refreshed" with WD-40.

Jim Peterson's TIPS was printed using PRINTALL16 on disk UTIL-31.

Dear Editor,

I have some comments prompted by the letter from "Chris" Christian in issue 39. My concern is mainly with his final grouse.

Chris dislikes programs that he thinks have no usable end product. A little thought about what we consider "Usable" for the Group in general does not seem out of place. After all, I suppose the TI*MES is almost the whole material benifit that most members receive. If Chris's views were widely echoed, this could account for some of the decline in our membership. We all have the opportunity, on the sub renewal form, of saying what we think, but this of course would reflect the opinions only of those not yet so fed up as to resign. Still, Its a start and maybe Alasdire Bryce will tell us what the indications are.

My view on this topic is that members still at work are unlikely to be looking hard at their TI's for support or to TI*MES as a prime source of technical information. No, they and the rest of us are doing what we do for intellectual stimulation, free from pressures of deadlines. Almost inevitably, I think, this drives us away from specialist programs unless they are presented in a way that highlights the universal features in them. To illustrate, I recently had occasion to inspect a program from Chris's area, although not up to his level. It was to design a filter for a phase-locked loop. Interest value for me, Zero. I soon went onto something else Only later did I reflect that it probably performed an optimization, under stated restraints, of several quantities, one at least of which could have been a complex number. If Chris had been around to outline the logic, I might have been using it now. So, I suggest that authors of specialist programs should send them to the library but let us see the trickiest bits and some commentary in TI*MES.

Another point about presentation. Members without XB are a minority but they pay their subs. While writing in XB makes better use of page space, authors might try to avoid contructions that are difficult to replace by plain basic.

Finally, about support for TI*MES. I have been through the two years of issues since I joined. Leaving aside routine announcements, I found contributions from 14 people of whom 6 were current or recent past officers, ie each year, about 6% of private members sent in something. That's on a head count; on volume contributed, we all know the answer. Through you, Editor, I would just say: Come on now Tom, Dick, and Harriet out there. Tell us about things you're rather pleased to have done, write of your difficulties. Don't stand in awe of the multicomputered polyglots who seem to breath a differnt air. OK., you may not have disks or printers or even a typewriter, but, I am sure that one or our leading brains could publish a simple program for you to write to tape in a way that our Editor could transmit to a printer via his expansion. Elaborate editing facilities unnecessary.

Yours

Walter Allen

This page has no content

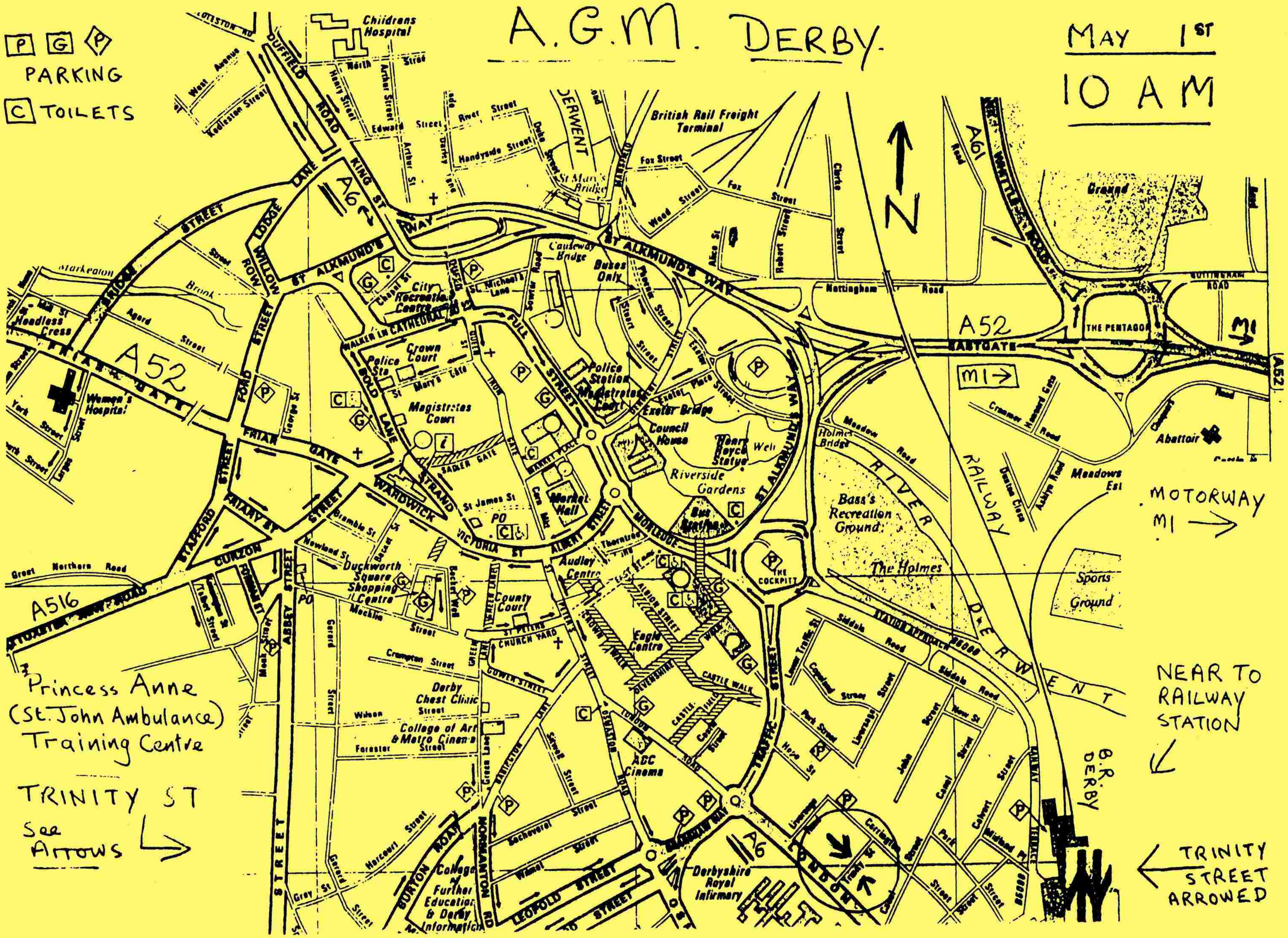
This page has no content

This page has no content

A.G.M. DERBY.

MAY 1st

10 AM



P G P PARKING
C TOILETS
N
Princess Anne (St. John Ambulance) Training Centre
TRINITY ST See Arrows
MOTORWAY M1
NEAR TO RAILWAY STATION
TRINITY STREET ARROWED
B.R. DERBY
A52
A6
A61
A516
M1
Childrens Hospital
Women's Hospital
Police Station
Crown Court
Magistrates Court
Council House
Riverside Gardens
Bus Station
The Cockpitt
Eagle Centre
Duckworth Square Shopping Centre
County Court
Derby Chest Clinic
College of Art & Metro Cinema
ABC Cinema
Derbyshire Royal Infirmary
Bass's Recreation Ground
The Holmes
Sports Ground
The Pentagon
British Rail Freight Terminal
Abattoir
RIVER DERWENT
RAILWAY