# TI*MES

TI99/4A USER'S GROUP (U.K.) CONTACTS

Chairman: Trevor Stevens. Tel. 0623 793077
249 Southwell Rd. East, Rainworth, Notts. NG12 OBN
Vice Chairman & Programming: Mark Wills.
12 "Rosehill", Betton St., Shrewsbury, Shropshire. SY3 7YN
General Secretary: Richard Twyning. Tel. 0623 27670
24 Peel Rd., Mansfield, Notts. NG19 6HB
Publicity Officer & Module Librarian: Philip Trotter. Tel.0642 817356
80 Martonburn Rd., Grovehill, Middlesborough. TS4 2TH
Membership Secretary & Past Issues: Alasdair Bryce. Tel. 0389 65903
51 Dumbuie Ave., Silverton, Dumbarton, Scotland. G82 2JH
Treasurer: Alan Rutherford. Tel. 0625 524642
13 The Circuit, Wilmslow, Cheshire. SK9 6DA
TI*MES Editor & Distribution: Alan Bailey. Tel. 081 508 1053
14 Shelley Grove, Loughton, Essex. IG10 1BY
Hardware & Projects: Mike Goddard. Tel. 0978 843547
"Sarnia", Cemetary Rd., Rhos, Wrexham, Clwd. LL14 2BY
Librarians:Cassette:Nicky Goddard. Tel.etc. as above.
Disk:Stephen Shaw.(Journal Exchange)
10 Alstone Rd., Stockport, Cheshire. SK4 5AH
Publications:Mike Curtis. Tel. 0209 219051
21 Treliske Rd., Roseland Gdns., Redruth, Cornwall. TR15 1QE

## MAGAZINE CONTENTS

ISSUE NO. 39 WINTER 1992/93

## EDITORIAL

This is the second big issue of the year. I hope it will help to entertain you in the longer dark evenings.

As you will see we have had one letter, reproduced in full, which leads to two distinct points. The primary one is the simple one that we have no letters column as such since letter of general interest are merely inserted in the run of other articles, as there seems no reason to separate them. Please write at whatever length you like. The second point raised is the writing of programs for specialist purposes. This is best done by someone with the relevant specialist knowledge, rather than an expert programmer. If the program becomes frequently used it may then be worth tailoring. Mathematical manipulation, e.g.to force convergence in successive approximations is an area where the expertise of an applied mathematician can be invaluable, but this can be substituted by interaction between the computer and the operator.

The product of the moment does seem to be the 80-column card. There are a number of programs available for it. It is necessary to have better resolution than offered by a modulator and RF input, although RGB input will make many TV's adequate.

You will see that Mike Curtis has provided a list of the publications available in the group library. If there are any other publications which you would like to borrow write to this magazine, to see if any other reader can help.

## DISCLAIMER

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

## NEXT COPY DATE

Copy by 1st.March please! Much of the copy for this present issue has been perfect. Some however is very light, and this costs 25p per original for darkening before reduction. Please help us keep costs down by printing nice and black!

## FROM THE CHAIRMANS CHAIR

T.STEVENS 1992

Well here we are again with another issue of TI*MES. Christmas is again with us and the dark nights are drawing in. Just the time for a few hours on the Computer. I set a competion two issues ago which was to write a one line program to do anything you liked. Well it seems that the light nights have not been as productive as the now dark nights as I only had a few replies. I suppose it does'nt take long to judge, but it would have been nice to see a few more entries. Now the winner of the Competion.... It is John DUNNING from Lancashire. So John the tape full of goodies is on its way to you. Here is John's entry.

THE ONE LINE LABEL PRINTER

```
1 CALL CLEAR :: FOR I=1 TO 7
 :: ACCEPT AT((I+8),1)SIZE(2
8):A$(I):: NEXT I :: OPEN #1
:"PIO" :: FOR P=1 TO 7 :: PR
INT #1:A$(P):: NEXT P :: CLO
SE #1
```

This program when run with extended basic and a Printer attached to your computer via a RS232 in PIO mode will print labels with a seven line input. Ideal for disk or cassette labels.

I will be arranging for the new year a TEXAS workshop in the Mansfield area. At this time no date has been fixed but I expect some time in March 1993. I have some demos and progams to show and also will be doing data exchanges. I hope to have all three of my computers on line for the day and also hope to have some more from other people. so we can really do some special things. I also intended to advertise the event so that other outsiders can attend. So if you are in the Nottingham area you may see a few adverts in the papers and radio.

Enough of the chit chat, now down to some real stuff. For the following artical you will require your consol and TI Extended Basic. If you want to save the programs then you will need a tape recorder. That's all you will need.

I have been reading up on Sprites and have come across a few rather fancy things that professional programers use when writing games. Plus some other little routines. When I started the writing of games programs I found that my first hurdel was using the Joystick Command. I found it a very cumbersome animal and ended up with long lists of gosubs and the like, not to mention the complicated X & Y situations that had to be tested. I will show you now how to run three types of joystick control. The first will be a way of moving your SPRITE only when the joystick is moved. This program allows you to move in all eight directions. It also is very precise.

```
1)  100 CALL CLEAR :: CALL SCREE
    N(13):: CALL MAGNIFY(2):: CA
    LL SPRITE(#!,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
```

As you can see the routine is very small and can be adjusted to your own games routine by altering the values. To explain what is going on I will go through the listing. In line 100 the screen is cleared (Call Clear) the screen is then set to dark green (Call Screen(13). The Call Magnify(2) sets a single sprite to twice its size. In other words it occupies 4 character positions. We then with Call Sprite allocate number 1 to be the sprite number with a character of 42 the (*), with a color of white (16) and set the start position of the sprite at dot row 100 dot col 100. We have not put in any movement values, so it is stationary. Line 110 is the work horse of the routine. The routine loops on this single line. It starts with the Call Joyst routine. This looks at No1. Joystick and sets the variables X and Y to be the operators. Call motion is then used to move Sprite (1) and the variables X and Y operated. The rest of the program is display of the X an Y variables onto the screen. The routine when you run it is very smooth indeed. I will not go into the maths deeply but here is the basis of the Y X conditions. As you may or may not know when you move the joystick it returns values in the X Y positions of 4 or -4 or 0 when not operated. See your handbook for a full discussion. Y contains the row and X the column velocities. If no movement of the stick is found then the value is 0, therefore the sprite stays still. If the stick is moved up then X will equal 0 but Y will equal 4. Our sprite needs negative row velocities to move up and positive velocities to go down. To get round this we change the value of the Y. We do this by placing a minus in front of Y. This then makes Y=-4, so up goes the sprite. However when Y equals -4 on the stick return then with -Y it returns a 4. However the X does not have to be changed as it relates directly to the Sprite +- relationship.(A bug in Texy maybe). The sprite is sped up by the *4 operator. If you wish you can change this figure to any positive figure, this will alter the speed to your need. Try 2,8,16 for example. However if you go over 31.75 it will calculate out at 127 which will cause an error message. As you can see this routine is in a loop but you can place this into a games or utility progam loop for use. To follow are two more joystick routines. The first gives you new graphic row/column position every time the stick is moved. The third is a progressive speed routine. This allows you to speed the sprite up then reduce it but moving the stick in the opposite direction. I will not discuss these, I will let you see if you can work them out.

```
100 CALL CLEAR :: CALL COLOR
(2,7,7):: CALL SCREEN(11)::
```

```
CALL HACHAR(24.1,40,64):: CA
L VCHAR(1.31.40.96):: CALL S
PRITE(#1.42.2,17,17):: R.C=3
110 CALL JOYST(1.X.Y):: X=SG
N(X):: Y=-SGN(Y)
120 CALL GCHAR(R+Y.X+C.CH)::
 IF CH=40 THEN CALL SOUND(-6
0,110.9):: GOTO 110 ELSE C=C
+X :: R=R+Y :: CALL LOCATE(#
1.R*8-7.C*8-7):: GOTO 110
```

```
100 CALL CLEAR :: CALL SCREE
N(13):: CALL MAGNIFY(2):: CA
LL SPRITE(#1.42.16.100.100)
110 CALL JOYST(1.C.R):: X=(X
+C)*-(ABS(X)<124):: Y=(Y-R)*
-(ABS(Y)<124):: CALL MOTION(
#1.Y.X):: DISPLAY AT(24.1):"
Y=":Y."X=":X :: GOTO 110
```

MANAGED TO WORK ANY OF IT OUT???

If you have problems then write to me and I will glady discuss the routines with you.

I have seen in our magazine some very good CALL PEEK values. I discussed in my artical PEEK and POKE some time back. some of the more useful. When you have only the consol and the Extended Basic plugged in you can only use PEEK. but it can be used by you. The RND function of basic is good but slow. Have a look at this.

```
100 RANDOMIZE :: CALL PEEK(-
31880.A):: PRINT A::: GOTO 1
00
```

This program has to be generated with the randomize statement at the begining of your program to set the RND flags inside the machine. After that Call PEEK will generate any number between 0 and 99. If you want higher try CALL PEEK(-31880.A+1) now we have 1 to 100. You may find this a little slow due to the print statement but it is much faster than old RND when you run it in program. You can also access DOUBLE RANDOM NUMBER ie 0 to 255. by using another Call Peek -31808.

```
100 RANDOMIZE :: CALL PEEK(-
31808.A.B):: PRINT A.B :: GO
TO 100
```

Speed is the thing to go for. as we noted above. Just to show you the difference in the RND and PEEK routines here is a program that will display that speed.

```
100 CALL CLEAR :: CALL SCREE
N(7):: CALL MAGNIFY(2):: CAL
L SPRITE(#1.42.16.100.100)
110 FOR N=1 TO 50 :: RANDOMI
```

```
ZE :: A=INT(RND5):: B=INT
(RND5):: 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
```

You do not have to put Randomize in twice but this example was done for you to see the two programs together and compare them.

Well that all for this month. In next TI*MES I will discuss Patterns. Key Calls and other nice little things. I hopefully will combine all these modules into a small program. I am affraid memory is full for this quarter. however I wish you all a very Merry Christmas and a Happy New Year.

## TI USERS GROUP UK MEMBERSHIP NEWS

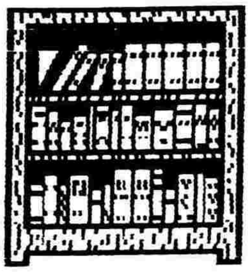
by Alasdair Bryce

The last few months have been rather quiet on the membership front. Since issue 38 we have been joined by only one new member. A warm welcome goes to Christopher Frampton from Southampton who owns an expanded system.

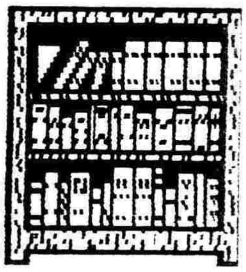
Having printed area lists in the last issue of TI*MES I would be interested to know whether any of you have managed to get local groups organised and if so how successful they have been. For those few group members who live north of the border in sunny Scotland I am extending an open invitation for possible meetings and exchanging programs (subject to copyright rules of course) and ideas to contact me at the address given on the front cover of the magazine. Anyone in the north of England who is interested in making it an international event can feel free to join in.

I still have large numbers of back issues available and would be only too happy to pass these on to you as my loft is beginning to bow under the strain. You can't all have a full set of TI*MES and at £1 each they a real bargain and an ideal late Christmas present.

I hope you all enjoyed the festive period and I'll see you again in the spring.



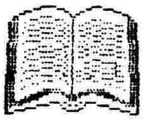
# TI*MES PUBLICATION LIBRARY



There has been no movement with the publications library for some considerable time now. I once got to the stage that I almost forgot that it was here! The only reminder was that my name appeared on the cover of the issue arriving through the letterbox every quarter.

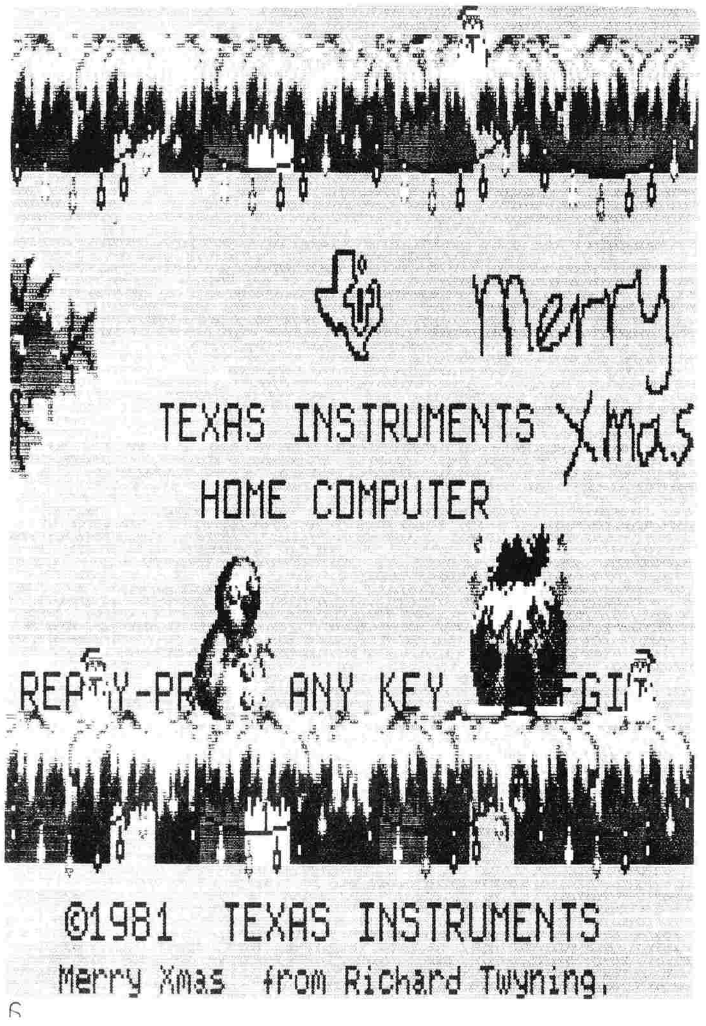
To recap with a list of the books available in the library:-
All books have a TPL prefix.

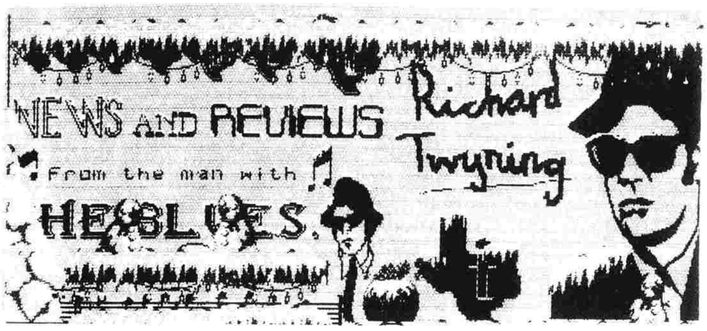
| No. | Title | Author |
|---|---|---|
| 001a | Software development | Vincent/Gill |
| 002a | Editor Assembler manual | TI staff |
| 003b | TI user reference Guide | TI staff |
| 004c | TI user read this first | TI staff |
| 005b | Using & programming the TI | Frederick holtz |
| 006b | TI favorite programs explained | Donald S. Kreutner |
| 007b | Your first TI99/4a program | Rodnay Zaks |
| 008b | Beginners Basic | TI staff |
| 009b | TI writer manual | TI staff |
| 010b | TI writer tips and tricks | Joyce Corker |
| 011b | Intro into assembly language | Ralph Molesworth |
| 012b | TI technical information manual | Unknown |
| 013b | Programming basic | Herbert D. Peckham |
| 014b | Get more from the TI | Gary Marshall |
| 015b | Mastering the TI | Peter Brooks |
| 016b | Getting started with the TI | Stephen Shaw |
| 017b | Learning to use the TI | Kevin Townsend |
| 018b | Games for your TI | Andrew Nelson |
| 019b | The orphan chronicles | Ronald Albright |
| 020b | Smart programming guide for sprites | Craig Miller |
| 021a | Entertainment games in basic & XBasic (Includes a cassette tape) | Khoa & Quyen Ton |
| 022c | The Texas program book | Vince Apps |
| 023a | The best of 99er Vol 1 (The ONLY volume) | Various |
| 024b | Fundamentals of TI assembly language | Morley |
| 025b | Dynamic games for your TI99/4a | Scott Vincent |



DEPOSITS a\8.00 b\ 5.00 c\2.00 MONTHLY HIRE 1.00/month

A happy Christmas and New Year to you all. Mike.


Merry
TEXAS INSTRUMENTS Xmas
HOME COMPUTER
READY-PRESS ANY KEY TO BEGIN
©1981 TEXAS INSTRUMENTS
Merry Xmas from Richard Twyning.



Dear TI'ers,

Did you all manage to read my article in the Autumn issue of TI*MES. I did get slightly carried away I suppose, but due to being away from The Notts Trent University gave me the chance to write a seriously long article which had a more useful content than my previous one.

As you can see, I've yet again spent some time improving my title. I loaded the original TI-Artist images from my normal title with Picture Transfer from MDOS, and put them side by side as one Myart picture. The snow, snowmen, and Christmas lights are pinched from actual game screens of Christmas Lemmings on the Amiga (I've got a bit more to say about the Amiga later!)

My freind came round in September with his Action Replay which is a game hacking device that fits to the side of an Amiga 500. Trevor Stevens came round with his Amiga 500+, and when you run some software on it, you can press an interrupt button on the Action Replay, and your software is interrupted. You can then do what you want with it. You can search your memory for graphics, and then save any screens that you find as IFF files.

We did this for several games, including XENON II and XMAS Lemmings. I converted the Lemmings screens to GIF on the Amiga and sent them up to the CRAY 2 using RS232 at 9600bps.

After getting them on the CRAY 2, I converted them to Myart using Picture Transfer. I loaded them in, and then super-imposed them on to both the TI-Title Screen image, and my article title.

By pressing SHIFT and 1 on YAPP you get an overlay palette which doesn't copy black pixels onto the destination, so you can mask everything off. It's an excellent facility, and works in both 256 colours, and 16 colours. The TI Title picture was done in 256 colours, and my article title was done in 16 colours.

I started back at university on the 5th of October, but the work up to yet is still slightly sad, such as teaching C to those who've never touched it. We are using C for graphics, so before things start hotting up, I'm hoping to get the full version of C for MDOS. It appeared a couple of issues ago on

9640 News, and the compiler was originally written for IBM's.
A 9900/9995 compiler on an IBM? YES! It came from a decent university in Texas (San Antonio I Think), and was used on the IBM's to compile C into 9900 or 9995. The object code could then be uploaded and run directly on the TI mainframe! Therefore, we'll have a mainframe version of C.
I've already got C99 for the 4A, and for MDOS, but Clint Pulley wasn't able to include true floating point routines. He had a complicated selection of routines which stored a single floating point number in several integer variables. Each time you want to add, multiply, or anything else, you have to call a function that would do it instead of just using an expression. Also, the last time Gary and I tried to use C99 for MDOS, either the compiler or linker wouldn't work (This may be 0.97H MDOS that's causing the problem though!).

The advantage with the new compiler will be that it's a totally ANSI C compatible version, with fully working floating point. It's Beery Miller who's doing most of the development work on it, but Clint Pulley is helping out, and converting all of his graphics libraries for block moves, hardware lines, and hopefully multitasking from C99 so they will work with the new system. I have received an early version of the compiler on some 9640 News disks which I received from Richard Sierakowski, but this is still only an IBM file on a TI disk, and has to be transferred to an IBM and run on that. I am not even sure if I used it on a PC to compile something, if the compiled program would even work yet on the GENEVE because of our VDP routines etc.
The latest news from Gary Smith was that Beery Miller was sending out a working version on the next issue of 9640 news. We've not heard anything lately, so I'm going to try and contact Beery Miller through Compuserve, which I've decided to subscribe to. This will enable me to get the very latest information, on hardware and software in the U.S.A.

I've not really noticed articles on C99 appearing in TI*MES very often, (I think the last time one appeared was the issue before the Cuffley workshop) so I thought I would give a useful tip in simplifying some of the things that might go wrong. The first time I tried playing with C99 everything went well until I tried assembling the compiled program. If you've not used C99, it works by compiling a piece of C source code into an assembly source file. The source file that is generated from your C program contains REFerences to other assembly routines that are contained in the C99 libraries. There is a slight problem however, that some routines are used in the source code, but their references are not included at the top of the user program. There is a correct way around this, by including specifiers in the main program, but as I've never sat down long enough to read it properly, I decided that it was just as easy to load the a ssembly source code when it's been generated by the C compiler, and actually insert the references for the desired routines.
This problem will most commonly arise when you use the commands printf and scanf which are C's equivalents to PRINT and INPUT. When you use the C99 compiler to generate an assembly

file from a C file that contains the commands printf or scanf, you can just load the assembly source code into the program editor. Then you will see a list of REF's, into which you should insert REF PRINTF and REF SCANF. Then try assembling the source file. The only option that is required is C, to specify a compressed object file. The R option is not required since C99 leaves off the R when specifying Workspace Registers.

The next step to actually run the C program, is to load the generated object code for your program. Also, the C99 libraries will have to be loaded, and then the program can be executed with the label START.

All of these files are option 3, so you could do what I did first, and spend the time to type all the module names into Editor/Assembler option 3, and then type the label START.

Then I discovered the Script Loader on the Funnelweb loaders menu. This allows you to load and execute a script file which is a DV80 file, and contains the filenames of the individual option 3 modules. The script file will then specify the starting label and the C program will begin to execute. The script loader even displays a list of all the modules which are being loaded, and displays a cursor at the side of each module as it is loaded, so you can see how far it's got! It's a damn useful utility, and as with most TI utilities, I came across it by accident!

<u>**Here's an example C Script File**</u>

```
*   Sample Script-Loader Script
*   Up to 15 DF/80 object files may be included

      AUTO                    Auto RUN with name after LAST
* Stick name of c object file on next line.
      FILE "DSK3.FILES;O"
      FILE "DSK3.CSUP"
      FILE "DSK3.C99PFF"
      FILE "DSK3.C99PFI"
      FILE "DSK3.CFIO"
      FILE "DSK3.PRINTF"
      FILE "DSK3.SCANF"
      LAST START          Terminating directive & RUN entry
```

The script file shown here is for a C program which displays a specified file to the screen. When creating my files I always stick to a particular format.

My original C source code is ended in ;C
The 9900 source produced by the C99 compiler is ended in ;S
The 9900 object code file is ended in ;O
If a list file is generated is it ended in ;L
And the script file for the program is left blank.

Therefore, for the program FILES, you would have the following on disk.

| | | |
|---|---|---|
| FILES | - | The Funnelweb script loader file. |
| FILES;C | - | The original C source code. |
| FILES;S | - | The assembly source code. |
| FILES;O | - | The 9900 object code. |

FILES;L - The list file generated by E/A or FunWeb assembler.

---

In my last article I promised a review of Batch-It.
It's possible that there are still a few people who haven't heard of Batch-It at all. I only heard about it from Trevor Stevens who found a review of Press and Batch-It in an old back issue of TI*MES.
There were rumours about Press for ages, and what it would do. It took advantage of whatever you had got, such as Supercart or Superspace, hard disk, RAM Disk, or GENEVE.
With the GENEVE, however, I'm unsure if it took advantage of resolutions such as 512 * 212. It was claimed that it offered full WYSIWYG which meant you didn't just see the layout of the text, but saw different text sizes displayed on-screen. It also does what my program "WordSpace" will do, which is use a disk file as memory for the editor. This has the advantage that you never worry about power cuts because your text is already on disk.
I've made this even safer by keeping the file closed until you enter the line you are currently editing, or you move to a different line. I might alter this though to improve access speed.

Oops!! Back to the subject.

Well, what will Batch-It do? Just about anything!
Imagine being in the middle of editing a piece of text and forgetting how to do something such as how to merge a file, or the control key sequence for changing font size or printing colour, or even forgetting someones address!
You would have to mess about getting the TI-Writer manual, or printer manual, and waste valuable time.

If only you could open a window in front of TI-Writer which allowed you to select what you wanted from a menu.

As with all TI projects, it's about as easy as getting a standard 99/4A to play digitized sound files!!!

That's where Batch-It comes in. In fact, Batch-It takes advantage of our stackless multitasking machines, and runs behind option 5 programs without them even knowing it!

Unfortunately, what the advertising failed to explain was that Batch-It programs need an area of memory that is 99% guaranteed not to be overwritten by whatever option 5 program it is monitoring. This area, is memory that regular option 5 programs would not expect to find anyway, and this means that you will need a cartridge with additional memory.
This should ideally be an 8K Supercart, or a 32K Superspace, but it will work with the mini-memory, although the size of the batch routine(s) will be limited to 4K.
This makes the 8K Editor/Assembler Supercart project submitted by Trevor Stevens in the last issue an even more worthwhile proposition.

Gary Smith, however, has not yet been able to find the 5565 RAM chips. Given a bit of time, I'm sure we'll be able to possibly work out a small re-wiring job to allow 6264's to be used, and even with a little piggy-backing, up to four, to allow a 32K Superspace to be built.

With the decreasing number of ROMOX boards, however, it may be worthwhile considering a totally fresh project in which we design a totally new board and use a single 32K chip.

Back to Batch-It!!!

Batch-It comes with a compiler that generates your Batch-It process from your DV80 file, and then there is a loader that loads the Batch routine and handles the multitasking between it, and the option 5 program it controls.

The actual option 5 program you are monitoring is loaded by the Batch-It routine itself, which means your routine could load a new option 5 program, or a new batch routine at any time. Your possibilities are only limited by your total disk capacity.

This gets around the problem of the mini-memory not actually having a memory-image PROGRAM loading option. There is still the problem of the compiler though, which is still an option 5 program, but you are provided with a small Batch-It routine that will load the Batch-It compiler.

There is also a routine that will load your batch routine, called MLOAD.

One of the advantages with mini-memory is that once your batch loading routine has been loaded into the mini-memory, it never has to be reloaded due to the mini-memory's battery backup.

The disadvantage with this though is it even further reduces the valuable memory, and only leaves you with 1700 bytes of total batch routine memory. That works out at around a 13 sector batch program source file. However, you could break up your task into as many batch routines you require, and load them as modules from one main batch menu.

Well, what do you do with Batch-It once you've got it loaded?

It's a bit different from normal programming languages, since it's only really intended for your Batch-It program to enhance another program, or programs, although it doesn't necessarily have to.

Batch-It does include variables, which are in the form of strings, and can be declared anywhere in the program with the DEFINE statement.

```
        DEFINE  VAR,40
```

The statement above creates a variable called VAR, which can be up to 40 characters long.

Your Batch-It program can load an option 5 program with the LOAD statement such as LOAD "DSK1.PROGRAM", or LOAD VAR, where VAR is a variable that contains the path and filename for the option 5 program being loaded.

When you've loaded your option 5 program, it won't start executing until you give a RUN statement. This allows programs that aren't compatible with Batch-It to be modified before they are run.

One such program is TELCO, which only requires one patch to memory to make it usable with Batch-It. This is done with the PATCH statement. The PATCH statement allows an hexadecimal address to be poked directly with an hexadecimal value.

If you're writing a Batch-It program that waits in the background for a certain keypress, then this is achieved with the ONKEY statement.

ONKEY 32, BEGIN causes the control of the Batch-It routine, to pass to the label BEGIN when the space bar is pressed. Other ASCII values can be used in place of 32 to allow control characters to be waited for.

The ONKEY statement doesn't cause the program to pause, so you'll have to include a GOTO such as:

```
LABEL
   ONKEY 32,BEGIN
   GOTO LABEL
BEGIN
```

You could even include several ONKEY's to check for more than one keypress.

There are three ways to send keypresses to your option 5 program. You can use KEY which sends more one or more characters to the program as a string, such as: KEY " " where the string of keypresses are enclosed in quotes, or KEY VAR, where the string is contained in the variable VAR.

The ENTER statement works exactly the same as KEY, except after the string is sent to the program, Enter (ASCII 13) is also sent.

The final statement for sending a keypress to the option 5 program is probably the most useful. This is CHAR. The CHAR statement allows control key values to be sent to the option 5 program. This allows Function key values etc. to be entered.

eg. To send an F9 to the option 5 program, you would use CHAR 15

Besides sending data to the option 5 program you will also need to get information from the option 5 program into your Batch-It routine. This is done with the GET statement which reads a string from the screen display of the current program into the specified string.

GET 20,0,VAR would read the characters at row 20, column 0 into the variable VAR. The number of characters transferred are equal to the length that the variable has been defined to.

As well as reading data into your batch program from your option 5 program, you may also want to read data directly into your batch program. This is done with the INPUT statement, which is similar to ACCEPT AT.

INPUT 10,2,VAR would accept a variable at the specified row and column. This variable can then be used for whatever purpose you want in your batch program.

One of the best uses for this is for writing your own option 5 loader:

```
*--------------------------
* Sample Batch-It option 5
* Loader
*--------------------------
       DEFINE NAME,40
       PRINT 23,0,"Enter Opt.5 Filename"
       INPUT 24,0,NAME
       LOAD NAME
       RUN
*--------------------------
```

## Summary Of Batch-It Commands

### Assignment Statements

DEFINE name,size[,"..."] - This command defines a variable with the name "name" of a size "size", and optionally assigns the characters in quotes to it.

GET row,column,name - This reads in characters from the screen starting at the character row, column into the variable with the name "name". The number of characters read will be the size of the variable specified when it was DEFINED.

MOVE name1,name2 - Moves the string stored in "name1" into "name2". If the string in name1 is longer than name2, then it is chopped to the size of name2.

VPAGE value - This command is used when Batch-It programs are run on the GENEVE to assign the VDP page. This is only required for using Batch-It programs with programs like TELCO which use more than one VDP page.

VBASE value - This is the base address for the screen display. This is used for programs that do not use the standard VDP memory map. You should only use this option if a batch program doesn't correctly accept input from the screen. If this occurs, the option 5 program in the foreground has moved the location of the screen table. The screen table can be placed at even increments of 1024 bytes throughout the VDP memory. This command will also allow you to access any part of the VDP memory (color tables, DSR information etc!)

### Control Statements

GOTO label - This command transfers control to the line in the program that begins with the label "label".

GOSUB label - GOSUB to the label "label"

RETURN - Must be placed at the end of a subroutine called with GOSUB. This will return you to the statement after the GOSUB which called the routine.

ONERROR label - This command checks the status of the internal ERROR flag. If an error has occurred (ERROR Flag value is -1), execution will be transferred to the line beginning with "label".

ONKEY key-value,label - Checks to see if the key specified by "key-value" (ASCII value representing that key) has been pressed.
If it has, control is transferred to the line beginning with "label", and the internal MATCH flag is set to zero.

ONMATCH label - This checks the status of the internal MATCH flag. If the MATCH flag equals zero, then the execution of the program is passed to the line of the program beginning with "label". This command is used after SEARCH, ONKEY, COMPARE, WAIT, and LOOK commands.

BATCH name - This loads and executes a previously compiled batch program - similar to chaining programs. Note: this erases the current batch program. This process is completely invisible to any option 5 application that may be running in the foreground when this statement is called.

PAUSE number-seconds - This pauses the batch program for the desired number of seconds.

END - The last statement in your batch program. This terminates the batch program and returns control to the program or cartridge used to run it. Compilation of a batch program also terminates at this point regardless of what it after it.

Input/Output Statements

CLEAR [row,column,size] - Places "size" number of spaces on the screen starting at the location specified by the row and column. If no options are specified the entire screen is cleared.

PRINT row,col,name - Prints the string "name" at the specified row and column.

INPUT row,col,name - Inputs data from the keyboard into "name" at the specified row and column.

ENTER name - Sends the characters contained in "name" to the foreground option 5 program, followed by Carriage Return (ASCII 13).

KEY name - Same as ENTER, but no carriage return is sent at the end of the string.

LOOK row,col,name - Will compare the characters on the screen starting at the row and column specified to the characters in the variable "name". If the characters match, the internal MATCH flag will be set to zero.

WAIT row,col,name - This is similar to LOOK, except execution of the batch program pauses until the characters on the screen at the location specified match those in the string "name". This does not effect the MATCH flag.

LOAD program-name - The option 5 program specified in program-name will be loaded into standard memory, but will not start running, until your Batch-It routine issues a RUN command. The ERROR flag will be set if the option 5 program could not be located on the device and under the name specified.

PATCH hex-address,hex-value - Puts the word (2 byte) value "hex-value" at the even address "hex-address" in a program previously loaded.

CHAR value - This sends the ASCII equivalent of "value" to the running option 5 application.

COMPARE string1,string2 - Compares string1 to string2, provided that both strings are the same size. If they are equal the MATCH flag will be set to zero.

SEARCH substring,mainstring[,pos] - Searches for the string "substring" in the string "mainstring" starting at either the beginning of the string, or at the character specified in the optional argument "pos". If a match is found, the internal MATCH flag will be set to zero. If the search cannot be performed, the ERROR flag will be set to -1.

KEYMODE value - Sets the TI key-scan mode to the value specified in "value". This must be set in the situation where the option 5 program running with your batch program uses some other key-scan mode.

BEEP - This produces a beep.

## Batch-It Flags

In addition to variables that you define in your batch programs, Batch-It has two variables that indicate the current state of the batch program. These variables are called "flags".

ERROR - This flag is set to -1 if the LOAD command is unable to load the application filename given, or if the SEARCH command cannot perform properly. This flag may be tested with the ONERROR command. If an error has occurred (it is set to -1) then control of the program will branch to the line with the label specified. If no error has occurred ERROR will be zero.

MATCH - This flag is set to zero if two strings or variables are equivalent. If not, then it will be set to -1. This flag may be tested with the ONMATCH command.

I've started writing a batch routine that will run behind Funnelweb. The first thing that the program does is to load Funnelweb with the option 5 name "DSK1.FW". It then issues the correct sequence of commands which takes the system straight into the text editor. Then it will wait until it is needed by using a loop which contains the ONKEY statement.

```
* LOAD and RUN Funnelweb
        LOAD "DSK1.FW"
        RUN
* Switch to TI-Writer menu
        KEY "32"
* Enter Text Edit
        KEY "1"
*
* Wait for batch file startup key
LABEL1
        ONKEY 128,FORMAT
        ONKEY 157,MENU
        GOTO LABEL1
```

The small routine above will get you into the text editor and will wait for either of two keypresses. As the requirement to FORMAT a particular document is required quite often, I've decided to give it its own special key. The other option will give you the standard menu that will let you achieve your change of font, printing colour etc.

The values for the "hot-keys" weren't arrived at very scientifically! I loaded Funnelweb and just pressed all the keys I could think of. A few keys appeared to stop the cursor, but they didn't cause Funnelweb to do anything, so I made a note of them and then ran an Extended BASIC program to get the ASCII values of the keys.

The keys that I found are listed below:

| | | |
|---|---|---|
| CTRL + | = | 157 |
| CTRL < | = | 128 |
| CTRL ? | = | 187 |
| FCTN : | = | 189 |

These keys are of course on the GENEVE keyboard, so you will have to try them on the 4A to see if they work, or find an alternate set of keypresses.

When the FORMAT option is selected, you are given an option of printing the text continuously, or with a pause at the end of each page:

```
FORMAT
       DEFINE Q,1
       PRINT 20,1,"                        "
       PRINT 21,1," Page Pause (Y/N)? "
       PRINT 22,1,"                        "
       INPUT 21,20,Q
       COMPARE Q,"Y"
       ONMATCH PAGE_P
*
* Format current file continuously
* Save current file
       CHAR 15
       ENTER "SF"
       CHAR 13
* Exit Text Editor
       CHAR 15
       ENTER "Q"
       ENTER "E"
* Enter Formatter and enter all defaults
       KEY "2"
       CHAR 13
       CHAR 13
       CHAR 13
       CHAR 13
       CHAR 13
       KEY "N"
       CHAR 13
* Return to text edit
       KEY "1"
* Load previous file
       ENTER "LF"
       CHAR 13
* Jump to end of text
       CHAR 15
       ENTER "S"
       ENTER "E"
       GOTO WAIT
*
* Format with pause between pages.
*
PAGE_P
       CHAR 15
       ENTER "SF"
       CHAR 13
       CHAR 15
       ENTER "Q"
       ENTER "E"
       KEY "2"
       CHAR 13
       CHAR 13
```

```
        CHAR 13
        CHAR 13
        CHAR 13
        KEY "Y"
        CHAR 13
* The user must get back to Text editor manually
* Go and wait for hot keys again
        GOTO WAIT
```

Batch-It has also been updated to include the possibility of adding device drivers for mice etc.  This lets you have mouse driven pull down menu's!!!

The additions to Batch-It in version two are as follows:

RESERVE size - Reserves "size" bytes of memory from the data area to use for driver programs.  Note the data area starts at >7FFF and grows backwards to >6000.  The reserve statement should always be the first statement in your batch file, this way you know where to put your driver programs.

EXECADR name - Returns a two byte address that points to the execution address of the previously LOADED driver program.

EXEC name1,name2,name3,name4 - Executes the driver program at the address in name (use EXECADR to get each address), passing four required parameters name1 to name4.  Name1's address will be in R0, name2's address will be in R1, name3's address will be in R2, and name4's address will be in R3.

Name1 to Name4 are stored using the following method:

Byte1 max size of name
Byte2 to Byte2 plus max size - data
Byte2+maxsize+1 - null byte terminator

Note that all strings in DEFINE are null terminated to the length of the actual data.  For example, DEFINE NAME,4,"X" will look like:

```
  >04X>00>00>00>00
```

Registers R11,R14 and R15 should not be altered by the user or Batch-It will become corrupt.  R11 (RT) will contain the return address to return from your driver back to Batch-It.

Well I hope you found that interesting.  It's definitely one of the most useful pieces of software we've got available.
The last time I knew of the price of Batch-It it was $14.95.  This was in their excellently produced 1992 catalog. If you would like a catalog, their address is:

Asgard Software
P.O. Box 10306
Rockville, U.S.A.
MD 20849

Here are some other products:

High Gravity - An educational and entertaining simulation of gravity in space. Shoot objects through conflicting gravity fields and watch the effects. Newtonian physics has never been better explained. Highly rated by MICROpendium. $4.95.

Infocom Adventures - Infocom has now gone out of business! Asgard have the complete rights to all these Infocom adventures at $14.95 each:

| | |
|---|---|
| Plundered Hearts | Suspect* |
| Hollywood Hijinx | StationFall |
| The Lurking Horror* | Leather Godesses of Phobos* |

* Requires 8K Supercart

The EXCELLENT Asgard mouse - Works with both the 99/4A and the GENEVE and compatible with all configurations of both computers. The Asgard mouse has a lifetime warranty and is compatible with TI-Artist, YAPP, Batch-It, Page Pro, Poster Maker, TI Pei, Classic Checkers etc.
The mouse costs $39.95 and the package includes driver software for various programs such as TI-Artist, YAPP etc. It requires an RS232 Card.

Picasso 2.0 - Picasso is an excellent drawing program that lets you create huge screens up to 480 * 342 in size - over twice the size of TI-Artist! The latest version includes definable icons, a proper fill, fill patterns, and an excellent zoom.
Picasso costs $14.95.

YAPP - Yet Another Painting Program by Alexander Hulpke!
The most advanced drawing system for the TI-99/4A and the Myarc GENEVE 9640. Combining all the popular features of other drawing programs, along with the spectacular graphics capabilities provided by the new generation of graphics hardware.

YAPP is the first program to take advantage of the 9938/9958 video processors on both machines.

Supports four different drawing modes: 256 * 212 in 256 colours,
256 * 424 in 256 colours,
512 * 212 in 16 colours,
512 * 424 in 16 colours.

Extensive drawing commands including an airbrush, different sized drawing brushes, lines, boxes, frames, (FAST!!)filling, circles/ellipses, cross-hairs, an "undo" function,
TEN! different palette types, providing masking operations, and logical colour mixing.
Ability to use colour clip-art, and compatible with TI-Artist Fonts and Instances!

Also, a FAST zoom mode that requires 192K VDP RAM for zooming on interlaced pictures. Also loads GIF pictures directly! HARDCOPY has also been incorporated directly into the program!

For an example of the output produced by YAPP, just look at my Merry Christmas message, and my Christmas title!

YAPP is priced at $29.95, and an essential first purchase for anyone with a newly completed 80-column card!

Pix Pro - The "Super-Convertor" will convert MacPaint, GRAPHX, Picasso, TI-Artist, Instance, RLE, Page Pro, to other formats. Any conversion you could want! GRAPHX to Instance, and back!
Pix Pro will even let you save a full-page MacPaint picture as a full page Page Pro!! Full colour support is included.

Spell It - A spelling checker in several versions for DS/DD disk drives, SS/SD disk drives, and hard disks. The SS/SD and DS/DD versions have over 25000 words, and the hard disk version has over 250000 words.

| | |
|---|---|
| SS/SD version: | $24.95 |
| DS/DD version: | $19.95 |
| HFDCC version: | $34.95 |

This is only a fraction of the range offered by Asgard!!!

---

In the last article I said that I needed to modify the address decoding of my Corcomp RS232, Horizon RAM disk etc.
As soon as I received the MEMEX 512 card, the higher banks of RAM were already turned off with the DIP switches. This prevents the more serious address clashes being incurred.
When I put the card in and began using it I noticed that when I set up the GENEVE RAM disk as DSK5, it became corrupt in certain places, and I narrowed this down to clashing addresses.

I did the address decoding on my Horizon RAM disk first, which gave me the problem of saving the data on it before powering it down. The actual programs aren't the problem, since they can be saved to a floppy, but on the GENEVE, the real problem comes when saving the actual DSR chip.

The loader for the RAM Operating System that I've been using won't run on the GENEVE at all, because it's incompatible with MDOS. This left me wondering how to reload the DSR after modifying the card, without sticking the 4A back in.

Then I remembered about a couple of programs by John Johnson. These are DSRSAVE and DSRLOAD.
Both of these will only run on the GENEVE from MDOS and need a parameter, which is the CRU address to save or load.

If you're at the A: prompt, which is DSK1, and type DSRSAVE 1100 then you would get a file on DSK1 called 1100 which holds the exact contents of your DSR which is located at CRU

address 1100, which would usually be your TI Disk Controller.
I was unsure if the Horizon was located at 1400 or 1600, so I saved all the CRU addresses I could think of, including 1000, which is the hard disk controller.
After saving everything to disk, I felt safe to power down the system and carry out the upgrade.

The modifications for the addresses decoding of each card came with the MEMEX card.

## HORIZON RAMDISK AMA, AMB & AMC DECODE MODIFICATION

This modification allows full use of the MEMEX card when properly applied. All CorComp Cards need a similar modification.

REQUIRES
Two 75LS244's and one 74LS138 and 18 inches of fine wire wrapping wire.

DIRECTIONS:
On the 74LS138 bend out pins 2, 4, 5, 6, 7, 16 for attaching wires. Cut off pins 9, 10, 11, 12, 13, 14, 15.

On one 74LS138 bend out pins 7, 13, 15, 17 for attaching wires.
Cut off pins 2, 4, 6, 8, 9, 11, 12, 14, 16, 18.
Trim lower end off of pins 3 and 5.

NEXT: Place the (cut) 244 on top of the uncut 74LS244.
Solder pins 1, 10, 19, 20, of the top chip to the bottom chip.

Add the 74LS138 on top of the 244's so pin 8 of the 138 can be soldered directly to pin 10 of the 244. Pin 1 and 3 of the 138 should be soldered to pin 3 and 5 of the 244 respectively.

Connect pin 5 of the 138 to pin 10 of the two 244's.
Connect pin 5 of the top 244 to pin 2 of the 138.
Connect pin 6 of the 138 to pin 16 of the 138 and pin 20 of the 244's.
Connect pin 4 of the 138 to U20 socket pin 4 (on the back of the card)
Connect pin 7 of the 138 to U20A pin 4 (bent out to clear socket)
Connect pin 13 of top 244 to #48 on edge connector.
Connect pin 15 of top 244 to #45 on edge connector (on back)
Connect pin 17 of top 244 to #46 on edge connector.

## CORCOMP CARD... AMA, AMB & AMC DECODE MODIFICATION

REQUIRES
One 75LS244 and one 74LS138 and 18 inches of wire wrapping wire.

DIRECTIONS
On the 74LS244 bend out pins 2, 4, 5, 6, 7, 16 for attaching wires.
Cut off pins 9, 10, 11, 12, 13, 14, 15.

On the 75LS244 bend out pins, 1, 7, 10, 13, 15, 17, 19, 20 for attaching wires.

Cut off pins 2, 4, 6, 8, 9, 11, 12, 14, 16, 18.
Trim lower end off of pins 3 and 5.

NEXT: Place the 74LS138 on top of the 244 so pin 8 of the 138 can be soldered directly to pin 10 of the 244. Pin 1 and 3 of the 138 should be soldered to pin 3 and 5 of the 244 respectively.

Connect pin 5 of the 138 to pin 10 of the 244.
Connect pin 5 of the 244 to pin 2 of the 138.
Connect pin 6 of the 138 to pin 16 of the 138 and pin 20 of the 244.
Secure the 244 to the card with double sided tape or glue. Pick a location to allow the following connections.
Connect pins 1, 10, and 19 of the 244 to any available ground on the CorComp card.
Connect pin 20 of the 244 to the 5volt (top right pin of an available chip).
Connect pin 7 of the 138 to the trace leading from #56 of the edge connector about 1/2 inch above the connector. Then cut the trace below the new connection.
Connect pin 4 of 138 to #56 on the edge connector.
Connect pin 13 of 244 to #48 on edge connector.
Connect pin 15 of 244 to #45 on edge connector.
Connect pin 17 of 244 to #46 on edge connector.

After doing all this, you can replace the cards. After modifying the Horizon I used DSRLOAD to reload the RAM Operating System, and then copied the files back. It worked first time, and I was extremely relieved. The RS232 card also worked, and I tested the card in the box. The glitching I was getting before had gone, and I decided to also activate the higher banks of the MEMEX card.
I activated the extra RAM and tried again. This time the glitching had returned. I've narrowed this down to the RAVE 99 speech adapter, for which there are no instructions for modification in the MEMEX manual.
Now I've subscribed to Compuserve I can contact Ron Walters personally, and ask him about the RAVE adapter, which also causes my MDOS MEMTEST program to lock up.

The TI-Forum on Compuserve is amazing, and they have a conference on Monday nights! I will possibly include some information from the TI-Forum in my next article.

---

In my last article, besides promising a Batch-It review, I also promised a MYART image loader. I had already produced a MYART loader for XB and XHI which would handle 256 colour pictures. This may seem the more difficult version to start with, but it isn't.
In G7 you need a byte for each pixel, so you need a byte when you save a picture. Each block of data is saved as a word, so in G7 pictures, the first byte contains the colour, which means the second byte contains the number of pixels that the colour runs for. In G7 a value of 0 for pixels means 256 pixels.

The problems start for G6 pictures because you've only got 16 colours. You can specify the colour in only 4-bits, so the rest of the word will specify the run-length for that colour. This means you need to seperate the colour value from the other half of the first byte, and this needs a bit of an equation which does slow down the loading process a bit more. However, I suppose this is slightly compensated for because you've got more bits to specify the run-length, so you can draw large areas of pixels without having to load an extra word of data.

To mask off the bits you will need to use an AND operator. If you AND the byte with 11110000 then you will be left with the four bits that describe the colour. This still needs to be divided by its least significant bit, whose value in this case is 16. What you're left with should be the correct colour value.

The other four bits that are joined to the four bits of colour data should also be masked, and added to the rest of the run-length data. The value of these bits start at 1, so they must be multiplied by the lowest bit value they will occupy when joined to the other byte. The highest value in the other byte is 128, so the four bits must be multiplied by 256.

The final routine you are left with for extracting the data from the file and calculating the true values is as follows:

```
30000 IF A$="" THEN 30020 ! If current string is exhausted, then
get another
30001 COL=ASC(A$)::RL=ASC(SEG$(A$,2,1))::A$=SEG$(A$,3,LEN(A$))
30002 RL=RL+((COL AND 15)6)::COL=((COL AND 240)/16)::RETURN
30020 IF EOF(1) THEN CLOSE #1::GOTO (done) ELSE LINPUT #1:A$ ::
GOTO 30001
```

Whenever you GOSUB to 30000, when it returns your new colour value will be stored in COL, and your new Run-length value will be stored in RL.

I have jumped ahead of myself slightly, because I've not mentioned what's at the very start of each file. The first word is an identifier to state what sort of picture you are loading (16/256 colours). This is one of the only crazy areas in TI computing that I know about!(Also some Macpaint picture formats!)

There doesn't seem to be much agreement as to what is contained in these two bytes! A bug in MYART caused the header to be written out wrongly which started the problems, and then some people use the first byte as a background colour value.

There are also differences between the header generated by XHI, and that Generated by YAPP. I think though that for the purpose of this program and future programs that may be for FORTRAN 99/9640, I'll make it recognize the headers for XHI and YAPP.

If you want, you can disable the checking, and force it to load as either a G6 or G7.

For G6 you'll also need to load the palette data for all your colours. I also had a few problems with this, because Alexander said he used the palette data from the XHI patch area when saving the palette values to the file. In the patch area, the data is organized as follows:

```
                                 |
         |<----1byte---->|<----1byte---->|
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         | | | | | | | | | | | | | | | | |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |  5-bits  |  5-bits  |0|  5-bits  |
             Red        Blue         Green
```

Whereas, on disk it's saved as follows (I discovered by a bit of experimenting, after getting it wrong the first time!):

```
                                 |
         |<----1byte---->|<----1byte---->|
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         | | | | | | | | | | | | | | | | |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         | 4bits | 4bits |0|0|0|0| 4bits
             Red     Blue            Green
```

Why should Missing Link users have all the fun! If you've got a GENEVE, or an 80-column card, and would like to know how MYARTs are saved, then first, here are the XB & XHI loaders for MYART images:

This is the 256 colour version with no checking

```
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
```

This is the 16 colour version with no checking

```
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)6):: COL=((COL AND 240)/16)+1 ::
      RETURN
30020 IF EOF(1)THEN CLOSE #1 :: GOTO 117 ELSE LINPUT #1:A$ :: GOTO
      30001
```

Both programs can be MERGEd with the Systex loading version of XHI, or you can stick in CALL INIT::CALL LOAD("DSK1.XHI") at the beginning.

At this time I must report that I've had problems with the Missing Link version of the Myart loader. This is the effect of the colour mixing of standard 9918/9929 bit map mode. The program draws an initial pixel, but when it draws the next pixel the pixels it's drawn previously get changed to the new value! Unfortunately, at the present time I've got seven assignments, so I'm unable to write a serious version. Could someone have an experiment with the Missing Link version of the program.
Here's the 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)6):: 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.

---

In the last issue was Stephen Shaws abbreviated TI-Writer information. I hope everyone read this. If you haven't, get it out again and read it now! After reading it I learned that if you type either @@ or && you will get @ or & when you run the file through the text formatter! I didn't know this.

TI-Writer manual is too big to sit down and read right through, but you can reach the end of a short article in a few minutes. The formatter I'm currently using is sector edited so I can use control characters for double print and underline.

After reading the article, I can convert the program back to normal so I can use 1000 Words with it again!

I also made an order from the disk library of the Disney disks. I can confirm that Peter Pan is an Option 5 file, but needs at least the 8K supercart. I suppose the game is not up to much, but it's more than worth it for the speech.

The speech is also excellent on Plant Genetics. Unfortunately, the rest of the disks are all GRAM files, so to run these requires a GENEVE or a GRAM KRACKER. Every bit of Plant Genetics uses speech, and it even says the full chemical name of DNA out loud!! The quality is a million times better than the Swedish-sounding Amiga.

To update the Amiga situation, you may remember that in the last article I said I had been through a few Amiga's, and after having one with a disk problem, I had got one that appeared to be working!

At this point I decided to connect my EPSON RX80 printer to the Amiga. Nothing happened! I was trying it for ages. I rewired the cable a few times (using a wiring diagram for an Amiga 500 printer port!). They don't give you any pin outs for any of the Amiga ports!

Still nothing happened, so I phoned Commodore. They wouldn't let me talk to an engineer. They said I'd got to contact my dealer, and my dealer could talk to the engineer! SAD! I phoned Dixons, and they said they would phone Crapadore, and then phone me back. After phoning me back, they gave me some crap about doing a copy command in Workbench shell to copy a text file to the printer. It didn't work. I said I needed a pin out of the printer port. The Ape on the phone at Dixons was slightly out of his depth when you mentioned ports! He said he would phone Crapadore and get them to phone me back.

He phoned back and said Crapadore refused to phone me! Very helpful for the sad company that dominates the profit crazed home computer industry! The dood at Dixons said I should go out and buy a printer cable. I did! Still nothing happened.

I phoned Dixons again, and he said phone the Commodore help line so they can send Wang computers to your house to fix it!

A week later, the bloke turned up and repaired it with a board that had a completely working printer port. It's a pity the keyboard didn't work! He said he would get a new board as soon as possible!

Over a week later, Trevor Stevens, and a friend with an

Amiga from University came round for a play. My friend from University had an Amiga magazine with him that said Commodore had reduced the price of the Amiga by 100 quid! I was instantly sadenned, and phoned Dixons to tell them I was fed up, and wanted to take it back. I phoned Crapadore and cancelled the service engineer. The next day I took the Amiga back, and was refunded 439.99. The Amiga cost 369.99, and the three year warranty cost 70.00.

That afternoon I went to Pheonix in Leeds and bought a 20 Meg hard disk version of the A600 with 2 Megs of RAM for 497 quid including joystick.

I got it home, and IT DIDN'T WORK!!!!!!!!!!
It had an intermittent fault on the video chip, and would totally lock out, and would even cause games to skip levels!!!!!!!
SAD!!!!

I phoned Pheonix, and they more or less told me to get lost. They said, "well, that's what the help line's for!" SAD! After phoning the help line, over a week later a bloke from Wang (should that be spelt with a K?) came and put a new board in it!

I can now report that Amiga number 5 is still working. We really have progressed a long way; Even with 2 Megs of RAM it won't load GIF images over about 60K!!! Very SAD!

I must also report that Oliver Timm in Germany has contacted me about the modifications to the Hard Disk Controller. Unfortunately he had to phone me from Germany, because he had prepared a letter, but his system was shorted out during an electrical storm. I have however, got the information that unlocks the universe:

On chip U9, cut pin 5 and pin 12 from the board and bend them upwards.
On chip U17 cut pin 5 from the board and bend it up!
Then connect some wire wrapping wire (left over from address decode modification!) between pin 5 on U17 to pin 12 on U9.

After the modification, the hard disk controller will then run drives with up to 15 heads.

When talking on the phone recently Gary Smith and I calculated the maximum storage capacity of the controller chip in the Myarc Hard Disk controller.

If you had a drive with 15 heads, 1024 cylinders, and 32 sectors per track, you would have 15 * 1024 * 32 sectors. This gives 491520 sectors. Using our standard of 256 bytes per sector would give you 125829120 bytes per drive.

However, the chip will handle 64K (65536 bytes) per sector, which means you could have 32212254720 bytes per drive.

If you divide this by 1024, then you have 31457280K per drive. If you divide this by 1024 again, you have 30720Megabytes per drive!!!!! That's THIRTY GIGA BYTES per drive.

The chip is limited to only allowing the control of four drives. I'm sorry, but that's only ONE HUNDRED AND TWENTY GIGA BYTES.

If you find four drives that have 2048 cylinders, then your 1978 TI Home Computer is limited to only SIXTY GIGA BYTES PER DRIVE. With four drives you can only have up to: TWO HUNDRED AND FORTY GIGA BYTES!!!!!!!

Do that on an Amiga, or a couple of VAXes!!!!!!!!!!!!

I must thank Colin Hinson at Texas Instruments for most graciously sending me of one of his only two TMS99000 data manuals.

What a chip! How does an instruction such as MPILCK sound. Multi-Processor Interlock! If you connect a totally different microprocessor to the TMS99000, you can send the instructions for this co-processor straight to the TMS99000. The 99000 says to itself, "This isn't one of my opcodes! I wonder if it's an instruction that the user has written in on-chip Macrocode!" If it is, then it is executed in the Macrocode deceder as if it was one of the 99000's own instructions. If it isn't found, then the 99000 says to itself, "I can't find this instruction in my Macrocode RAM. It's probably an instruction that's recognized by the current co-processor."

Then the instruction will be sent to the connected microprocessor. This is the most efficient system for supporting co-processors, as both chips share one area of memory that is managed by the TMS99000. I am unsure if the 99000 is still available though! There is also a floating point version with floating point instructions, which was out a few years before SAD intel chips had floating point!

I think that the Texas Instruments TMS34020 24-bit colour card project may be back on also. We were considering a colour palette chip that allowed 256 colours per pixel line selected from a palette of 16.7 million. Gary says there's a palette chip now that allows 512 colours per pixel line.

I've got to write to TI for more information on this.

Trevor Stevens has also given me the address for Alexander Hulpke, and I am going to write to him for any information he can give me about the 4A and GENEVE PC emulator card. If I find anything out I will report on it in the next article.

I have recently had a letter printed in an Amiga magazine called CU Amiga. It is edited by a sad dancer who recently let an history of computing slip through which didn't mention TI. We only got a mention in the 1983 price list which said you could buy the 4A for 189.99.

I wrote a very long letter pointing out the truth, which was printed, but he ommited a few things. I said that TI invented efficient speech synthesis using Linear Predictive coding that didn't sound Swedish, but the bit about sounding Swedish was omitted! He obviously didn't like it because he knew it was true. All the bits about our stackless RISC architecture that describes Workspace switches was included which I was pleased about, and he even left the bit where I said "Chew on that Motorola" because TI invented the first transistor radio in 1954, and Motorola are supposedly known in the USA for supplying radio equipment to Police, Ambulance and Fire services!

I also said that TI invented a system of fast moving graphics which he may have been familiar with called sprites!

He tried to put the letter down with his sad comments, but that's what you expect from a sad little dancer with no culture. He couldn't think of anything useful to say when dealing with a Bluesman!

I even made a quote from Amiga Format magazine which had an article about the new Amiga 4000's. A normal Amiga 24-bit animation rendering program would take five hours. It takes the Amiga 4000 less than a second. Do you think it could be because it contains a Texas Instruments TMS34020 producing 10MIPS.

I don't buy the aforementioned magazine, and only bought the copy that had my letter in it. I only knew it had been printed because I received a phone call from someone who had read the letter. He must have used directory enquiries to get through to me because they didn't print my number.

I mentioned our group, so he phoned to tell me that someone had given him some boards from a Texas Instruments PC which he wanted to go to a good home rather than throwing them away.

He gave me his address so I could publish it in case anyone wanted to contact him about the boards.

Don Cox
5 Cornfield Road
Middlesborough

Phone: 0642 815914

The boards are for a PC compatible, but they are made by TI and should have some good chips on them.

I've done a bit of work to the After Hours bulletin board, and at Christmas I may be able to get it up and running. When we get it sorted out it will be run from 3pm on Sunday afternoon, which as you may have seen is now charged as a local call, so no matter where you are in the country it will only cost a quid an hour.

Here's another 28 column printing program I've recently done. It needs a temporary file to work with.

```
1 OPEN #1:"PIO.LF" :: PRINT #1:CHR$(27);"G";CHR$(15):: CLOSE #1
2 PRINT "":"":"":"" :: INPUT "Enter program list name:":PRG_FILE$
   :: PRINT "":"":""
3 INPUT "Enter temporary work file: ":COL137$
100 OPEN #1:PRG_FILE$,DISPLAY ,VARIABLE 80 :: OPEN #2:COL137$,
    DISPLAY ,VARIABLE 28
110 LINPUT #1:A$ :: PRINT A$ :: PRINT #2:A$ :: IF EOF(1)THEN
    CLOSE #1 :: CLOSE 2 ELSE 110
120 OPEN #1:COL137$,DISPLAY ,VARIABLE 28 :: OPEN #2:"PIO",DISPLAY
    ,VARIABLE 137 :: DIM SC$(67):: R=0
130 LINPUT #1:A$ :: SC$(R)=A$&RPT$(" ",45-LEN(A$)):: R=R+1 :: IF
    R=67 THEN 139 ELSE IF EOF(1)THEN GOTO 170 ELSE 130
139 R=0
```

```
140 LINPUT #1:A$ :: SC$(R)=SC$(R)&A$&RPT$(" ",45-LEN(A$)):: R=R+1
    :: IF R=67 THEN 149 ELSE IF EOF(1)THEN GOTO 170 ELSE 140
149 R=0
150 LINPUT #1:A$ :: SC$(R)=SC$(R)&A$ :: R=R+1 :: IF R=67 THEN 160
    ELSE IF EOF(1) THEN GOTO 170 ELSE 150
160 FOR R=0 TO 66 :: PRINT SC$(R):: PRINT #2:SC$(R):: NEXT R ::
    PRINT #2:CHR$(12 ):: R=0 :: GOTO 130
170 FOR R=0 TO 66 :: PRINT SC$(R):: PRINT #2:SC$(R):: NEXT R ::
    CLOSE #1 :: CLOSE #2 :: CALL SAY("FINISHED")
```

And finally, here's a bit of 9900 that calculates the first 20000 prime numbers in 8 seconds.

```
0100:  LWPI >D0
0102:
0104:  LI   R0,3
0106:
0108:  LI   R1,>300
010A:
010C:  MOV  R0,*R1
010E:  LI   R2,>300
0110:
0112:  MOV  *R2,R3
0114:  MPY  R3,R3
0116:  C    R4,R0
0118:  JH   >126
011A:  MOV  R0,R4
011C:  CLR  R3
011E:  DIV  *R2+,R3
0120:  MOV  R4,R4
0122:  JEQ  >12E
0124:  JMP  >112
0126:  MOV  R0,*R1+
0128:  MOV  R0,R5
012A:  BL   @>200
012C:
012E:  INCT R0
0130:  CI   R0,20000 * Maximum value to be checked
0132:
0134:  JL   >10E
0136:  XOP  0,0

0200:  A    R4,R2
0202:  JNC  >206
0204:  INC  R1
0206:  A    R3,R1
0208:  RT
```

And definitely finally, here's a bit of code that will pass the Status Register to an XB program.

```
       DEF  STTS
STRASG EQU  >?????  I can never remember the addresses for equates
GPLWS  EQU  >?????
MYWS   BSS  32         Create my own set of registers!
ST_R   BYTE 2          Length of status register value (2 bytes)
ST_REG DATA 0          Actual contents of status register
```

```
SV_REG DATA 0            Used to save contents of GPL WS Reg. 8
STTS   MOV  R8,@SV_REG   ! save a register !
       STST R8           store Status Register in R8
       MOV  R8,@ST_REG
       MOV  @SV_REG,R8   restore R8 to previous value
       LWPI MYWS         Load my own Workspace
       LI   R0,0         Standard variable
       LI   R1,1         First parameter
       LI   R2,ST_R      Location of Status Register value
       BLWP @STRASG      Assign Status Reg value to XB variable
       LWPI GPLWS        Reload GPL's Workspace
       B    *R11         Return to XB
```

Well that's if for now!!!!! Have fun!!!
MERRY CHRISTMAS from Richard T.

## ATTRACTIVE STAIRCASES.

Walter Allum

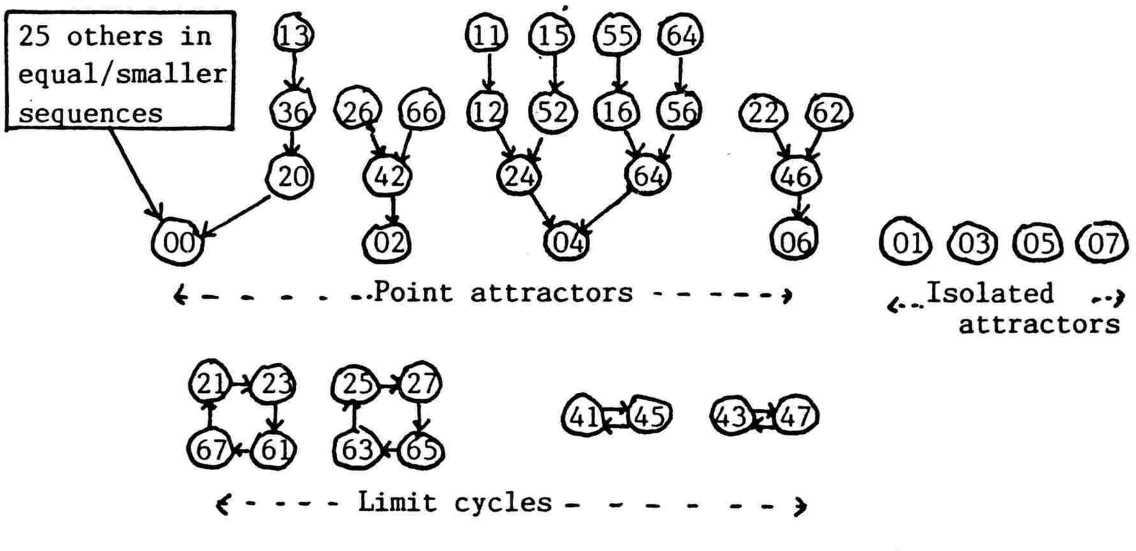
No, not something out of the Architectural Journal!

You recall that Stephen Shaw brought us a puzzle about numbers, cast in the form of a staircase (TI*MES 38,p.31). As, no doubt, he is inundated with your answers, I shall not bother giving mine. My purpose here is just to observe that the problem gains dimension, and the coding greater scope, if you work in other arithmetics and/or make simple extensions to the defining relationships.

This is, of course, a study in "attractors"; how many there are, of what kind and how distantly their influence is felt. With the original specification, we soon discover that, regardless of the radix (base) of the number system:

a) There are always point attractors--once reached, never left. Some are isolated, i.e. they cannot be reached from any other starter, and so define a one-tread staircase. This appears to happen only for numbers 0,n where n has no factor in common with the radix R.
b) There are sometimes "limit cycles" which, when entered, eventually repeat the entry value and thus terminate the staircase.
c) Some numbers can feature only as starting treads.

So far as my skimpy evidence goes, you get limit cycles only when R is a power of a prime. The diagram below illustrates the case for R=8 (octal).



Whether there are limit cycles with linear sequences feeding into them I do not know. Nor do I know whether the longest staircases for a particular radix are ever associated with such cycles or only with linear sequences.

It appears that, with a prime radix, p, all staircases end on 0,0 except for the trivial one-treads arising from the isolated attractors 0,1 to 0,p-1. The maximal sequence is also longer than for any other radix of similar size, e.g.

| Radix: | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|
| Max.seqn. | 5 | 12 | 4 | 6 | 13 | 32 | 6 | 44 | 12 | 13 |

You can also consider modifying the calculation as, for instance

A(I+1) = units part of (K1 * A(I) * B(I))

B(I+1) = units part of (K2 * B(I) + K3 * A(I+1))

Working in arithmetic radix R, there is clearly no point in making any K equal or exceed R. Negative values might cause problems and I'll leave them to you! Stephen's original specification corresponds, of course, to all K=1 and R=10.

Hunting for laws in this business seems very much a matter of "now you see them, now you don't".

```
 _     _     _       _     _  ___     _
|_|   |  _  |   |_| |_| | |_   |  | |_| |\ |
| \ .|_| .|_| | | | \ | _|   |  | | | | \|
```

MEng,CEng,MIEE,MSAE

Chartered Engineer        Chartered Electrical Engineer
Chartered Electronic and Radio Engineer

27 Howey Rise, Frodsham, Cheshire, WA6 6DN

Tel: Frodsham 33723

Amateur Radio G3GKS    RSGB    G-QRP    RAOTA    BVWS

12th October 1992

Alan Bailey,
TI*MES Editor,
14 Shelley Grove,
Loughton,
Essex,IG10 1BY.

Dear Alan,

Most magazines have a page or more devoted to readers letters which allows the readers to raise matters of interest or concern but TI*MES seems to have no such facility. This lack of a forum is my first complaint. It could be argued that the AGM is the place where members can air their views which brings me to my second complaint which is the choice of Derby as a venue. As I had no wish to travel some seventy miles of single carriageway road both ways I spent several hours with the full BR timetable looking at several routes from Frodsham to Derby. The latter may claim to be a railway town but none of the options would endorse that view as each route would have occupied a full day. That is why I was absent and the same goes for the Wrexham workshop to some extent. Would it not be possible to choose more suitable locations such as Crewe (THE railway town),Manchester,Stockport,Liverpool, Chester or any station on the Manchester to Holyhead North Wales line ?

My third complaint is the apparent lack of publicity for the group although I am sure Philip Trotter is doing his best. Richard Twyning has not seen any mention of TIUGUK in Personal Computer World recently as they have not run group lists since July 1991 after which time PCW was completly revamped. May I suggest that since most radio and electronic enthusiasts are usually computer owners that the appropriate magazines are targeted ? For example, Maplin Electronics magazine runs a free Club Corner every month where we could have a permanent notice. Other possible magazines include Television, Practical Wireless,Radio Communication,Short Wave Radio,Ham Radio Today,and ETI.

In order to emphasise the question of publicity I should like to quote my own case. I bought my TI99/4A in 1982 and, because of the high cost of expansion proceeded to expand very slowly as cash became available. By the time I got around to a disk system TI had opted out and in spite of numerous telephone calls to various dealers I was left to struggle on with only tape storage. Also there were the two TI user groups which being commercial enterprises both failed and a module exchange service which was of no interest as such. One day a faulty Acorn Electron console only came into possession which was repaired and expanded to operate my printer and with disk storage. The TI99 was dumped in my loft and would be there today had I not seen mention of the group in PCW early in 1989 so I joined the group and retreived my TI99 from the loft. A mention of TEX-COMP in TI*MES enabled me to obtain a COR-COMP

disk controller from the USA as well as TI-WRITER and EDITOR/ASSEMBLER so enabling me to make use of the group disk library. Thus I am grateful to both PCW and TIUGUK. All I need now is an 80 column card which is why this letter is being processed on a PC !

My final grouse concerns the programs which appear in TI*MES which represent computing as an art form for its own sake rather like the pure mathematics practised by pure mathematicians which has no practical use. My main interest is in writing number crunching programs which produce a usable end product. These programs are mainly for the analysis,design and optimisation of radio, electronic,and electrical devices and systems mostly in connection with my hobby of amateur radio etc. although before my retirement I have used them to set examination questions for polytechnic degree students. (Richard Twyning can thank his lucky stars he didn't cross my path !) If any members are at all interested in my programs I would be happy to print them in TI*MES or submit them to the disk library.

Yours sincerely,

R.G.Christian (Chris)

R.G.(Chris)Christian.

## Treasure Trail
by Peter Hutchison

Right! Treasure trail is back again. What a time I've had with Scott's adventure's particularly Ghost Town. You'll all be glad that I managed to complete it after a WHOLE year being stuck, trying to find that last treasure - there is NO CLUE what-so-ever in the game to get to it, it is part of the tips below. Many, many thanks to Barrie Clark who gave me some really vital information that lead to the solution. Thanks Barrie!!

### PYRAMID OF DOOM

How do I get into the pyramid? dnuora gniggid emos od ot deen uoY
How do I open the locked door? ti tih dna tsif ruoy revoC
How do get rid of snake? cisum yalP
What do I do with the skeleton? ydob eht etelpmoC
How do I get up from ledge? epor worhT
What is the Heart of Iron? der gnihtemos yortseD
How do I survive the poison? dnah ruoy revoC
Where is the last treasure? erutinruf ehT

You'll need 13 treasure, most are revealed after some examining especially the objects there and doing things to them. Carry the pistol at all times!

### GHOST TOWN

What do I do if it gets dark? deb a deen uoY
How do I get rid of the ghost? sdnah esU
What does the map point to? aera suoniatnuom eht yrT
How do I open the jail door? citengam gnihtemos esU
How do I open the other door? efas dna eciffo hpargelet rht yrT
Why doesn't 'ole Paint move? mih eohs, mih rupS
How do I get back to the town from teepee? !yltcaxE
How do I destroy safe? slacimehc eerht deen uoY
What do I use to cause explosion? yek a esU
The mirror is giving me 7 years bad luck? ti roveC
Where do I find the Cup? noolas ni krad eht ni ecnaD

That last one is a nasty one, how the hell was I supposed to know that!
Barrie's adventure editor helped me, he's using a disk version, does anyone have a Mini-memory copy at all? I would like a copy.
I am now working on Savage Island part 1, yes, I know a really difficult one. Has anyone completed that?
Hope you'll find the above tips useful. See you later.

Peter Hutchison
6 Moorlands View
Free School Lane
Halifax
W Yorks
HX1 2XQ

ARTICLE by Stephen Shaw

In the last issue's Disk Library report you may have noticed that the PLATO interpreter is now available on disk- it requires an Editor Assembler grom and ram at >6000 in the module (Super Space, Super Carte or home built as described in Issue #38).
There are 500 disks of Plato data out there!!!! Library funds are alas now non-existant, as the last pennies were used to obtain a sampling of 50 Plato disks.
I must admit to a certain appreciation for any serious computer program which puts up on screen, as a menu selection:
  PRACTICE: HUMAN REPRODUCTION.
PLATO was not commercially sold here, and barely advertised in the States, so you may not have heard of it. It is a VERY old educational tool produced by Control Data Corporation for use in schools (on those large pre-home computer computers!). PLATO is in the form of an interpreter which uses data on subject disks to go through tutorials and drills. There is little in the way of graphics, and no sound - the TI implementation pushes TI's modules pretty far in terms of memory size and there was no room for any extra stuff.
Except that very late on they managed to squeeze some speech onto THREE disks!
PLATO material covers ages 5 upwards - anyone fancy a course on Nuclear Physics?
.............
My experiences of using a SMALL computer came in handy at work- required: a sorted output from 400 data items from a program that could only handle 220 items. The data was well and truly random! Solution: Split the raw data into two chunks about the middle mark. Sort each of the two chunks.
From each chunk extract data to form half of the total when added to a similar amount from the other chunk- in this case, we ended up with two files one for A-J and one for K-Z. Each of these was of a size that could be sorted and then the two files (which now did not need sorting) could be appended to form a fully sorted file for A-Z. No-one else could think of how to do it!
.............
I notice several of our members are shy about using their postcodes. I recently made a phone call to a national body to ask for some literature. Insetad of giving my full address, subject to mis-spelling etc, I was asked for only the house number and the post code, and IMMEDIATELY my full address was read to me for checking. Think of the computer and the program to handle that! Post codes have their uses.
......................
FUNNY DISKS
If you ever find a disk which youf disk manager programmer tells you has a single file upon it called $$1, occupying only 7 or so sectors, and at the same time tells you that 360 sectors have been used (or in some cases 358 when the first two are ignored by some managers), you probably have a PLATO disk, which has the all important data USUALLY found in Sector 1 actually all the way out at Sector 167. This means you cannot catalog the files very well the usual way.
Of course you could copy Sector 167 to Sector 1 to catalog the disk, but the information still would not be very useful, being made up of very odd names.
What you need is a special PLATO cataloguer, and we do indeed have one, which will read the disk and display data as shown over the page...

sample "catalog" of Plato disk:
5246.3
CURRICULUM: Basic Skills Reading
SUBJECT: Judging What You Read
PROGRAM PACKAGE: Author's Purpose and Your Conclusion
Working with Facts and Opinions Tutorial
Working with Facts and Opinions Drill
The Author and Your Conclusions: Review
Review Drill
Available from the disk library!

---

## EMBEDDING ASSEMBLY CODE INTO AN EXTENDED BASIC PROGRAM CARRIER

by Col Christensen
Brisbane User Group
Australia

We all know how slow the Extended Basic loader for object code is, so the benefits in speed of loading with the code embedded in an Extended Basic program carrier are tremendous. There's many a good assembly program gathering dust because it is sooo slloooww in loading.

A discussion on the procedure could probably be covered in something like a dozen easy lessons. I would like to take, in this issue, the case of those who have produced some Assembly Code of their own, in which instance, the discussion can be covered in just one easy lesson. It's all really easy, but what I mean to say is, that trying to explain the procedure to make it easy, is not as easy as anticipated. Now that that's as clear as a summer day, we can go on.

First I shall set out the ten (holding up all the fingers of the left hand and three on the right) easy steps to embedding your own assembly code into a basic program then explain each of the steps individually.

1. DEFine the start address of your code. Who doesn't?
2. Make sure the assembled code WILL run in the Extended Basic environment. Another normality.
3. Next assemble into relocatable code in compressed form. Not normal for Extended Basic code. Needed for 4 and 5.
4. Assess the size, in bytes, of the object code.
5. Find the address of the entry point.
6. Make the source code absolute so that the assembled code ends at >FFE0, and again assemble this time as TAGGED OBJECT code, not compressed.
7. Go to Extended Basic.
8. In the command mode:- CALL INIT :: CALL LOAD("DSK1.filename")
9. In the command mode:- CALL LOAD(-31952,a,b,a,b). More on the values for "a" and "b" later.
10. Type in a small 5-line program.
11. Edit line 110 in the program.
12. Type SAVE DSK1.LOAD

All very simple, wasn't it?

Well, almost! Last things first then. Here's the program that you need to type for STEP 10.(next page)

```
100 CALL INIT
110 CALL LOAD(16376,67,67,32
,32,32,32,x,y)
120 CALL LOAD(8196,63,248)
130 CALL LINK("CC")
140 ! DO NOT RESEQUENCE
```

Now back to the beginning for the explanations.
STEPS 1, 2 and 3. No comments necessary.

STEP 4. Use a sector editor, such as my favourite DISK-AID, and read the first sector of the assembled code.
e.g.

```
Drv1 ASSEMBLY   Sze>2D0 Sec>22
Addr 0 1  2 3  4 5  6 7
     8 9  A B  C D  E F
 00  011E C053 4255 4720 ...SBUG
 08  4949 2041 0000 4204 II A..B.
```

Ignore the very first byte. Take the next two bytes, e.g. 1E and C0. these indicate the size of the file, >1EC0. Keep this value for later use.

STEP 5. Still with the sector editor, find the last sector or so of the file. Look for the label you DEFined, e.g. XBENTR.
e.g.

```
Drv1 ASSEMBLY   Sze>2D0 Sec>6F
Addr 0 1  2 3  4 5  6 7
     8 9  A B  C D  E F
 40  2020 2020 2020 2020
 48  2020 2020 2020 2020
 50  3500 6E58 4245 4E54 5..XBENT
 58  5220 2020 2020 2020 R
 60  2020 2020 2020 2020
```

Three bytes in front of the text, "XBENTR", you should see a "5". Between them both are two bytes indicating the actual entry point address relative to the start of the code. In the above, the address is >006E. Keep this value for use later.

STEP 6. Place an absolute origin directive at the beginning of your source code. In the example, the size of the assembled code as found in step 4 is >1EC0 and it needs to end at >FFE0. By calculation in hex, >FFE0 - >1EC0 + >E120. Insert at the beginning of your source code:

```
AORG  >E120
```

Assemble the source code with the output as TAGGED OBJECT code.

STEP 7. No need to comment

STEP 8. "filename" is the assembled tagged object code with an absolute origin.

STEP 9. The values for "a" and "b" when placed in the CALL LOAD statement, ensure that the program lines typed (or merged) in, will be placed in memory, just below the assembly code loaded in Step 8. We must subtract 2 from the absolute origin (>E120) previously found.

So subtracting 2 leaves >E11E. Convert each byte to decimal. >E1 = 14*16+1 = 225 the value for "a". The second byte is the value for "b". >1E = 1*16+14 = 30. We have the values for "a" and "b" so type:-
CALL^LOAD(-31952,225,30,225,30)
for the example above. This sets the Extended Basic memory space pointers to just below the Assembly code.

10. Typing in, or merging, the sample program places it just below the Assembly Code. The bigger the Extended Basic program, the further down in memory it will extend.

11. Line 110 sets up the REF/DEF table at >3FF8 to show a dummy label for the entry point and also its address in memory so that CALL LINK can find it. The entry offset address, >006E, found in Step 5 must be added to the absolute load address in Step 4 above. >E120 + >006E = >E18E, the actual entry point. Use its two bytes for the values of "x" and "y". >E1 = 14*16+1 = 225. >8E = 8*16 +14 = 142.

12. Saving the Extended Basic program in the usual way results in all the memory space from the bottom of the XB program to almost the top of memory being saved. You might notice that the size of the XB program you just saved to disk uses many more sectors than the little Extended Basic porgram carrier would have used. Run the program now and see the wonders of your efforts.
======================

## BITS AND PIECES

by Col Christensen Brisbane User Group Australia

Printing errors sometimes occur in Basic programs published in magazines. A few have come to light lately, being the result of the original program being listed to disk in text format then processed through the TI-Writer formatter. There are some symbols that are used in Basic but to the formatter are flags for specific action. The @,&, * and ^ characters come to mind. The formatter will ignore each of these as a print character and set out to follow a set course of action relative to the particular character flag. But when the * is encountered, it and the next two characters are removed from the text. So a program line that should read

```
    420 N=L*256+M     would show
as
    420 N=L6+M
```

Print errors such as these are a pain to the novice or even the not so novice programmer and make it difficult to get a program typed in up and running. So if you come across something that looks unusual when running a typed program, try to see if the @, &, * or ^ might have been deleted accidently.

[Art Greens version of TI Writer allows EASY amendment to Formatter control characters, to avoid this problem- sjs]

I had been playing around with an XB routine to scramble some numbers or letters from a string. After trying a few ways that randomly manipulated the string, I found the routine too slow. Then the following evolved which kept track of which char was already used by means of a parallel array. It had difficulty though in finding the last few unused chars by randomly stabbing here and there in lines 160 and 170 until one was found.

```
100 REM SCRAMBLER/UNSORTER1
110 RANDOMIZE
120 N=20 :: DIM X(20)
130 REM HAVE AN ORDERED STRI
NG TO START WITH
140 FOR I=1 TO N :: Z$=Z$&CH
R$(I):: NEXT I
150 FOR I=1 TO N
160 R=INT(RND*N)+1
170 IF X(R)=1 THEN 160
180 REM
190 X(R)=1
200 Y$=Y$&SEG$(Z$,R,1)
210 NEXT I
220 REM SHOW THE RESULT
230 FOR I=1 TO N :: PRINT AS
C(SEG$(Y$,I,1));:: NEXT I
```

By using the lines below to replace 160-180 that problem is overcome. If it randomly selects a char that has been already used, it just steps through until it comes to the next unused one and selects it. See lines 160-180.

```
160 R=INT(RND*N)+1
170 IF X(R)=0 THEN 190
180 R=R+1 :: IF R>N THEN R=1
 :: GOTO 170 ELSE 170
```

Had an inquiry on how to chain BOOT/MENU programs so that one would be able to run the next and so on. With the advent of 1 Mbyte ramdisks and harddrives, the 24 menu options can soon get filled up. The idea is to use two copies of the BOOT program but with different filenames for the second copy.

The snag with using the two copies was that when you tried to configure the second copy then save the configuration back to ramdisk, the alterations would be done to the original BOOT program. To overcome this it was necessary to go into the second copy with a sector editor, find the filename BOOT and change it to its new name. All would hopefully work perfectly then.

BOOT has a utility to save all the menu options and other configurations such as screen colours and filename defaults to itself on disk. Type in the defaults for the key presses 1, 2, 3, P, T, and D but escape out after typing each by pressing FCTN/9. Press the F and B keys to get your favourite foreground and background colours. Then save everything and the only way to save your configuration is to press FCTN/5 as if to edit menu names then escape (FCTN/9) to reach the save option. All your current filename options and screen colours will reappear next time you load BOOT.

Another great addition in Version 10/16/89 (press V at the menu screem to see your version) is the ability to save the menu options list to a disk file. Pressing FCTN/T (]) produces a prompt saying "Put DSK..." for you to state the pathname to save the option list to. If you press FCTN/R ([) you get "Get DSK..." and you can load a previously saved list to the screen. If this list is what you want permanently for your menu, then save to BOOT after pressing FCTN/5 and FCTN/9. These CFG lists are saved in D/V80 format and can be edited with TI-Writer but you must save as PrintFile (to DSKetc) so that the Tab lists don't get saved too.

A SIMPLE way to get more than 24 menu options is to use only one BOOT program and have multiple configure files. You could save, through the "Put DSK..." facility, a number of sets of menu option files onto the ramdisk under different xxxx-CFG filenames. (WARNING - Do not save through the FCTN/5, FCTN/9 Save method as BOOT itself will be re-configured on the disk. It needs to remain configured to your first menu.) Later, by loading the required file through "Get DSK..." you have whichever set of menu options you like. I like that idea and shall try it shortly.

[BOOT is available through the Group disk library- sjs]

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

DISK LIBRARY ADDITIONS...
NEW DISKS since SEPTEMBER 1992

PLATO... now that we have the PLATO module on disk, the disk library has added some 50 data disks, covering various subject. PLease refer to file PLATO on main library disks for details. Full library list available on four disks- just send four disks and return postage! There are MANY more Plato data disks available, your disk library merely lacks the funds to obtain them all.

>VIDEO- TI DEMO VIDEO- This video opens with the only known video designed for use with the rare peripheral PHP 2300- Video Controller. The controller used the index pulses on the videotape to allow keyboard interaction with the video recorder and this tape was designed for use in stores to allow shoppers to see demo's of their own selection. Also on the tape are tv adverts, other point of sale videos, and appearances by the TI on tv shows, including Tenille singing along with Music Maker. The tape concludes with a TI tutorial video for TI salesmen - how many statements would get them into trouble with our Trade Descriptions Act?
Five pounds per week rental, fifteen pounds deposit. PAL STANDARD VHS.

>BLUES 1A. A number of TI artist pictures (_P) of legendary blues singers with some textual detail in dv80 files.

>BLUES 1B. A very very large Sound FX file of Midnight Blues, 353 sectors. You will need additional memory to hear all of it - adding Super Space's 8k only allows about two thirds of it to play. Refer to docs with the required commercial program Sound f/x.

>MYART & YAPP A. Four pictures: Alien-My, Bigclck-My, Calc-My, Office-Yap. I have not been able to produce what I consider satisfactory pictures on my TI. Requires SmartCopy or a Geneve plus yapp.

>MYART & YAPP B. As above, three pictures, PAL-YAPP, WINDOW-MY and WINDOW2-MY.

>MYARC DM5 V 1.3 -UPDATE ONLY for Myarc Disk Manager 5. DOUBLE SIDED DISK. (Counts as two).

>GIFS A. Two pictures-Guitar and JLH-GIF to be used with commercial program gif-mania or a Geneve. Also a Geneve program GIF2-EXE.

>GIFS B. One gif picture file, TI-99.

>GIFS C. One large gif picture file TI-AD.

(Gifs A,B and C also available on one double sided disk, counts as two disks).

>STOR MOR a commercial program from Harrison Software, duly licensed. PLEASE DO NOT PASS AROUND. Price including disk and postage FOUR POUNDS. This program allows you to use very large string arrays in your Extended Basic programs by a few simple CALL LINK commands, which place a single dimensioned array in either high or low memory (your choice).

I have identified some of the cartoon characters on the RLE disks (MinMay, Joylef, Jpnmat, Robotek etc) and located a source for the genuine videos they came from - not commercially available in the UK. If you are interested in renting Japanese animation videos, let me know. Two pounds per week, fifteen pounds deposit for those I get (3 or 4 hour tapes), or I can put you in contact with a UK source. These are NOT the sort of cartoons the BBC would show. WHEN they become commercially available (if?) expect to see high prices- typically US$35 per hour in America. SOME of them are profoundly explicit. Watch out for some on Channel 4 over Christmas (4Mations series). ROBOTECH is often available for 1.99 or 2.99 from budget stores.

The disk library is operated by Stephen Shaw (address on front cover), and the copying cost is just one pound per disk side plus a handling fee of one pound per order. If you do not wish to send blank disks, please also add one pound per disk. Donations of funds and programs are always welcome.
The complete disk library catalog is available only on disk in DV80 text files, please send four disks and return postage.
=====================================================

Feather or Mira Fractal....

This is a very little program with a great deal to offer, and many of the finest graphics workers have been associated with it. My first contact was in the American magazine ALGORITHM (#3.3) (edited by famed mathematician A K Dewdney) in an artical by equally famed computerist Clifford A Pickover. It is this form of the program printed below.

While I was still exploring this very rich world of images, along came the UK magazine FRACTAL REPORT (#23) with an artical by Paul Gailiunas, quoting from a book by Hans Lauwerier, of a mapping by Gumowski and Mira. The listing given there differed minutely from this one.

The shape is defined by variables AA and B, which are best located between the values of -1 and +1. AA seems to control the shape- the number of "legs" - while B seems to control the level of intricacy.

Some combinations of variables will result in the plot quickly becoming inescapably entwined with one or more attracting fixed points. No need exploring those too much. Many combinations result in a feathery form appearing.

I found better results keeping AA negative.

There seems to be a special case when B is +1, when a different family of patterns can be found. Reducing the scalar multiplier in line 140 is often necessary when you do this.

Variables X and Y are the start locations. With feather plots they seem to have little effect on the result, but when B=1 they can make a difference.

Here is the original program....

```
100 AA=-0.48 :: B=0.93 :: C=2-2*AA :: X=3 :: Y=0
110 W=AA*X+C*X*X/(1+X*X)
120 !
130 FOR N=1 TO 1E12
140 CALL LINK("PIXEL",X*8+95,Y*8+120)
150 Z=X :: X=B*Y+W :: U=X*X :: W=AA*X+C*U/(1+U):: Y=W-Z
160 NEXT N
```

The program is written for Extended Basic+ The Missing Link (which requires 32k + disk system). ANY program that allows you to plot on screen is suitable, and there are several such.

The variables X and Y are multiplied by a scalar which determines the size of the image on the screen, and an offset is then added (or could be subtracted) which determines the position of the image on the screen (up a bit, down a bit etc).

This program is plotting a chaotic attractor.

The loop is VERY large, and you must determine yourself when a pattern has ceased meaningful development. For some plots I saw continued meaningful plotting even at N=28,000 - while for others as few as 5000 points sufficed. The Missing Link allows you to dump to printer at any time, and you can break the program without losing the image- say to save as a TI Artist file. Other graphics languages may require you to add a CALL KEY routine to sense when you have had enough, although this would slow things down a little.

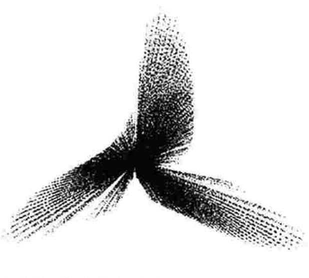
==============================

This is the same program slightly modified....

This time I have presented the program in one "random image" format, and you will find some images lacking development- just break and run again.

This listing is adapted from Fractal Report (#22) and was presented there by Ian Entwistle.
(for the Missing Link):

```
100 ! ENTWISTLE GENERATOR 91
110 ! FROM GUMOWSKI & MIRA
120 ! -1<A<1      .9<B<1.001
130 ! IF B<1 THEN CY<40
140 RANDOMIZE :: CALL LINK("BOX",1,1,192,240)
150 A=RND*.8 :: IF RND<.3 THEN A=A-RND*.5 :: IF RND<.1 THEN A=-A
160 B=1 :: IF RND<.15 THEN B=B-RND*.12
170 IF B=1 THEN CY=99 ELSE CY=30
180 SC=2 :: IF RND<.2 THEN SC=3 :: IF RND<.1 THEN SC=4
190 FOR T=1 TO CY
200 IT=994 ! better with MORE points
210 C=2-2*A :: X=4 :: Y=0 ! try random x & y too!
220 W=A*X+C*X*X/(1+X*X)
230 FOR N=0 TO IT
240 CALL LINK("PIXEL",X*SC+85,Y*SC+123)
250 !
260 Z=X :: X=B*Y+W :: U=X*X
270 W=A*X+C*U/(1+U):: Y=W-Z
280 NEXT N :: SC=SC+1 :: CALL LINK("PRINT",180,200,STR$(SC)):: NEXT T
290 GOTO 290
```



In the above variation as an added bit it has been made into TWO loops, the outer loop controlling the scalar multiplier. We then plot a SMALL number of points, then do it all again with a different multiplier. The result is quite interesting!

FRACTAL REPORT is ten pounds per year (approx bimonthly) from Reeves Telecommunications Labs Ltd; West Towan House, Porthtowan, TRURO, Cornwall, TR4 8AX (subs for UK).

ALGORITHM is a quarterly from Canada, a years sub to the UK is US$24 from:
ALGORITHM, P O Box 29237, Westmount Postal Outlet,
785 Wonderland Road, London, Ontario, CANADA, N6K 1M6.

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

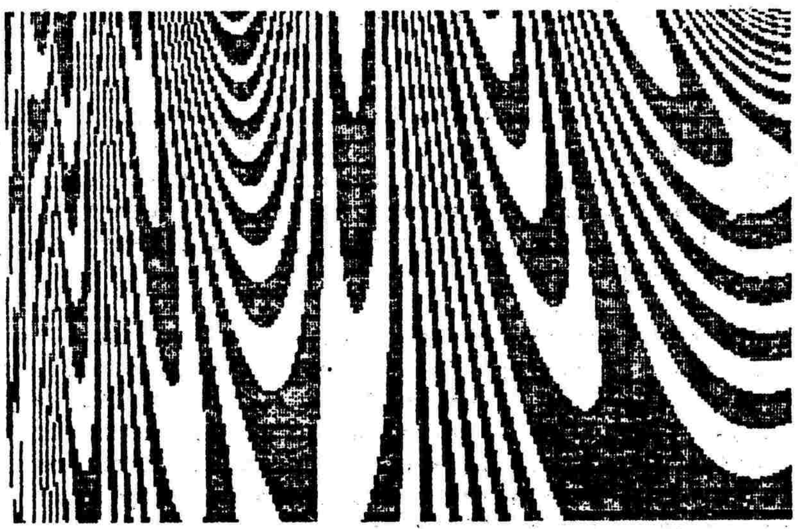
The next little listing takes absolutely ages to fill the screen, ~~for~~ so I will give some details to justify it. As usual the commercial disk program THE MISSING LINK is used, but any language allowing you to plot single pixels will suffice.

The program plot is based upon: X(T+1)=R*X(T)*(1-X(T))

Normally this would be used to plot X(T) on the X-R plane, which yields a bifurcation plot, which we covered a while back, but this program instead plots X(T) on the X(0)-R plane which gives a different result.

Source: L D Magguilli writing in ALGORITHM (see above) Oct-Dec 1992, Volume 3 Number 4. He cites M Szyszkowicz.

```
100 SW=240
110 SH=190
120 RS=3.5 :: RE=4
130 DR=(RE-RS)/SH
140 XS=0 :: XE=0.5
150 DX=(XE-XS)/SW
160 IT=3
170 SC=20
180 FOR J=1 TO SH
190 R=RE-J*DR
200 FOR I=1 TO SW
210 X=XS+I*DX
220 Y=X
230 FOR K=0 TO IT
240 X=R*X*(1-X)
250 ! X=3*X*(1-X)
260 NEXT K
270 K=INT(SC*ABS(X-Y))
280 IF K/2=INT(K/2)THEN CALL LINK("PIXEL",J,I)
290 NEXT I
300 REM
310 NEXT J
320 GOTO 320
```



For a different but slower plot remove the ! in line 250 above, or change 270 to: K=INT(SCALE*X).
This program may take several hours to fill the screen!
If you have 80 column capability, try plotting different colours for different values of K in line 280.
==================================

# INTERMEDIATE LEVEL ASSEMBLY LANGUAGE TUTORIAL
by Mack McCormick

## Direct Access to the TMS9902
## Universal Asynchronous Receiver Transceiver (UART)

### Introduction

The purpose of this tutorial is to describe the operation and programming of the UART directly at the chip level by-passing the RS232 ROM. These procedures will allow you to customize your RS232 input/output routines as well as better understand the operation of the system. As a practical example I recently found the need to transmit a code to my "Smart" modem but due to critical timing problems the standard RS232 ROM code would not handle the operation. A few lines of assembly language direct to the chip and I was in business.

This tutorial assumes you have the following minimum hardware; console, monitor, 32K, RS232, one disk drive, and the Editor/Assembler package.

The following references will provide additional insight to the operation of the RS232 and are all non-proprietary; Editor/Assembler manual, TMS9902A Asychronous Communications Controller data manual (TI), Microprocessors/Microcomputers/System Design (TI), and How to Build Your Own Working 16-bit Microcomputer (TAB books). The only proprietary information used was the RS232 ROM source code which is not necessary to understanding the operation of the 9902.

### Chip Architecture and Overview

The purpose of the chip is to provide interface between a microprocessor and an asynchronous communications channel. Let's review for a moment here what we mean by this term. The data is transmitted to the chip one bit at a time using the Communications Register Unit (CRU) over the hex bus (fire hose).

The chip takes the bits and transmits them out the RS232 port one bit at a time like Indians in a row. The data stream is asynchronous (not constantly synchronized with the other system) so a start bit is required at the beginning of the stream to tell the other system to prepare to receive data and one or more stop bits are required at the end to tell it we're finished.

The chip has several key features which enhance it's versatility. It can send/receive characters from 5 to 8 bits long, be programmed for 1, 1 1/2, or 2 stop bits, even, odd, or no parity, fully programmable baud (bits per second) data rates, and a highly accurate interval timer with resolution from 64 to 16,320 microseconds (good for clock applications). These features enable the computer to be programmed to interface with any system using standard RS232 protocols.

The chip is interfaced via the Communications Register Unit (CRU). It is assumed the reader is somewhat familiar with the operation of the CRU. If not, it is recommended you stop here and read the accompanying novice tutorial on the CRU before proceeding . The 9902 occupies 32 bits of CRU space which are used to control/interface with the microprocessor.

The 9902 interfaces over the RS232 port (outside world) using five lines; request to send (RTS), data set ready (DSR), clear to send (CTS), data out (XOUT), and data in (RIN).

The RTS line goes low whenever you activate the transmitter but data will not be sent until the CTS line is active. The DSR line does not affect the receiver transmitter but your application software may check this line before sending/receiving data.

Output CRU Bit Assignments

Of the available 32 CRU bits only 23 are used and we will be using even less. Bit 31 and 0-21 are used.

| ADDRESS | NAME | DESCRIPTION |
|---|---|---|
| 31 | RESET | Resets the chip |
| 21 | DSCENB | Data Set status change interrupt enable |
| 20 | TIMENB | Timer Interrupt Enable |
| 19 | XBIENB | Transmitter Interrupt enable |
| 18 | RIENB | Receiver Interrupt enable |
| 17 | BRKON | Break On |
| 16 | RTSON | Request to Send on |
| 15 | TSTMD | Test mode |
| 14 | LDCTRL | Load control register |
| 13 | LDIR | Load Interval register |
| 12 | LRDR | Load receiver data rate register |
| 11 | LXDR | Load Transmit data rate register |
| 10-0 | | Control, Interval, Receive data rate, transmit data rate, and transmit buffer registers. |

Bit 31 (RESET)- Writing a one to this bit resets the device.

Bit 15 (TSTMD)- Writing a one to this bit causes RTS to connect to CTS, XOUT to connect to RIN, and DSR to be held low. This bit allows us to test our programs without another computer connected.

Bits 14-11 - These four bits control which of the five registers are loaded with the data in bits 10-0.
1XXX - Control register
01XX - Interval register
001X - Receive data rate register
00X1 - Transmit data rate register
0000 - Transmit buffer register
0011 - Loads receive and XMIT registers
X - indicates don't care.

Bits 10-0 - The data.

The control register is used to select the character length, clock operation, parity, and the number of stop bits for the transmitter.

Bits 7 and 6 - Stop bit selection. 00=1 1/2, 01=2, 10=1, and 11=1
Bits 5 and 4 - Parity selection. 00=None, 01=None, 10=even, and 11=odd
Bit 3=1 always
Bits 1 and 0 - Character length. 00=5 bits, 01=6 bits, 10=7 bits, and 11=8 bits.

The receive data rate register is used to select the receive baud rate using the following formula. The frequency is 1 Mhz. Take half this value and divide by the desired baud rate. For example to obtain the value for 1200 baud the formula is 500,000/1200=416.66 . Using normal rounding the value becomes 417 or >1A1.

The transmit data rate register is programmed in the same manner but is usually programmed with the same value simultaneously.

The transmit buffer register is used to hold the next character to be transmitted. You must always send 8 bits.

Input CRU Bit Assignments

The same 32 bits of CRU space are also used for CRU input. They are defined as follows:

| ADDRESS | NAME | DESCRIPTION |
|---|---|---|
| 31 | INT | INTERRUPT |
| 30 | FLAG | REGISTER LOAD CONTROL FLAG SET |
| 29 | DSCH | DATA SET STATUS CHANGE |
| 28 | CTS | CLEAR TO SEND |
| 27 | DSR | DATA SET READY |
| 26 | RTS | REQUEST TO SEND |
| 25 | TIMELP | TIMER ELAPSED |
| 24 | TIMERR | TIMER ERROR |
| 23 | XSRE | TRANSMIT SHIFT REGISTER EMPTY |
| 22 | XBRE | TRANSMIT BUFFER REGISTER EMPTY |
| 21 | RBRL | RECEIVE BUFFER REGISTER LOADED |
| 20 | DSCINT | DATA SET STATUS CHANGE INTERRUPT |
| 19 | TIMINT | TIMER INTERRUPT |
| 18 | N/A | ALWAYS 0 |
| 17 | XBINT | TRANSMITTER INTERRUPT |
| 16 | RBINT | RECEIVER INTERRUPT |
| 15 | RIN | RECEIVE INPUT |
| 14 | RSBD | RECEIVE START BIT DETECT |
| 13 | RFBD | RECEIVE FULL BIT DETECT |
| 12 | RFER | RECEIVE FRAMING ERROR |
| 11 | ROVER | RECEIVE OVERRUN ERROR |
| 10 | RPER | RECEIVE PARITY ERROR |
| 9 | RCVERR | RECEIVE ERROR |
| 8 | N/A | ALWAYS 0 |
| 7-0 | RBR7-0 | RECEIVED DATA |

Bit 22 (XBRE) - Transmit Buffer register empty. The transmit buffer does not have the next character to be transmitted.

Bit 21 (RBRL) - Receive Buffer register loaded. Bit is true when a complete character has been received. This bit is reset by an output to bit 18 (RIENB, receiver interrupt enable).

Bits 7-0 - Receive Buffer Register. Contains the most recently received character. If fewer than 8 bits then right justified.

Transmitter Operation

The transmitter is initialized by the RESET bit being set. RTS and XOUT are set placing the transmitter in it's idle state. When the CPU sets RTSON then RTS is active (LOW) and data is sent when (CTS) input goes LOW. Let's examine what an asynchronous character looks like.

```
      Start                        Parity  Stop
       Bit        Character          Bit    Bits
1--         --------------------------------------
  !         !LSB!LSB+1!       !MSB!      !       !
0 !_____!___!_____!______!___!______!_______!
#    1      5,6,7, or 8        0 or 1 1,1 1/2, 2
Bits
```

The start bit is always a logic 0. The character is sent/received LSB first. The parity bit is set if enabled to make the number of 1's in the character even, odd, or don't care (none). Finally 1, 1 1/2, or 2 stop bits are sent as required by the receiving system. Thus we see that each 8 bit character may require up to 12 bits to transfer.

Summary

There are obviously many aspects of this chip we have not covered. The reader is encouraged to consult the non-proprietary information available to learn more. The principle reference is the 9902 data manual. If you would like to see more tutorials of this nature please give the editor your comments and suggestions. We have included an accompanying A/L tutorial which demonstrates the key features of the chip. Happy programming. Mack

```
****************************************
*                                      *
*  DIRECT ACCESS TO THE TMS9902 UART   *
*       BY MACK MCCORMICK              *
*          74206,1522                  *
*                                      *
*  R0-SCREEN LOCATION, R1-CHARACTER    *
*  R12-CRU SOFTWARE ADDR, R15-VDPWA    *
*     FCTN QUIT TO EXIT                *
****************************************

       IDT  'RS232'       IN CASE YOU'VE NEVER USED THIS PUT RS232 IN OBJECT CODE
       TITL 'DIRECT ACCESS TO THE TMS9902 UART' PRINTS THIS AT TOP OF EACH PAGE
       DEF  RS232         ENTRY POINT TO PROGRAM
       REF  KSCAN,VSBW
* USE E/A LOAD AND RUN. CALL THIS FILE ANYTHING YOU LIKE
* I DID NOT AUTO-START THE PROGRAM SO YOU COULD USE DEBUG ON IT
* THIS PROGRAM ACCEPTS A CHARACTER FROM THE SCREEN AND
* SEND IT OUT THE RS232/1 PORT AND BACK IN (TEST MODE).

*-- SYSTEM EQUATES --*
LED     EQU  7->40        CRU BIT FOR PRETTY LIGHT ON CARD
RESET   EQU  31           RESET BIT
LDIR    EQU  13           LOAD INTERVAL REGISTER
RBRL    EQU  21           (INPUT) RECEIVE DATA REGISTER LOADED
RIENB   EQU  18           RECEIVER INTERRUPT ENABLE
RTSON   EQU  16           REQUEST TO SEND ON
XBRE    EQU  22           TRANSMIT BUFFER REGISTER EMPTY
GPLWS   EQU  >83E0        THE GPL WORK SPACE LOCATION
MYREG   EQU  >8300        USE SOME HIGH SPEED RAM
KEYVAL  EQU  >8375        BYTE LOCATION FOR KEYVALUE RETURN
STATUS  EQU  >837C        GPL STATUS BYTE
R0LB    EQU  MYREG+1      LEAST SIGNIFICANT BYTE OF R0
BAUD    DATA >1A1         VALUE FOR THE RECEIVE AND XMIT REGISTERS
CNTRL   BYTE >82          VALUE FOR THE CONTROL REGISTER
        EVEN

*-- PROGRAM STARTS HERE --*
RS232   LWPI MYREG
        LI   R12,>1340    CRU BASE RS232/1, >1380 FOR RS232/2
*                         >1540 FOR RS232/3, >1580 FOR RS232/4
```
* ---->continued

```
*********************************************************
*                                                       *
*   INITIALIZE RS232/1 TO 7 BITS, NO PARITY, 1 STOP BIT *
*                                                       *
*********************************************************

       SBO  LED            TURNS ON THE LIGHT ON THE CARD. NOTE WE USED A MINUS
*                            OFFSET TO GET BACK TO >1307 FROM R12'S >1340.
*                            NOTE THE MYARC RS232 LED IS NOT MAPPED HERE
       SBO  RESET          RESET 9902

* NOTE HERE THAT RESET SETS LDCTRL, LDIR, LRDR, AND LXDR SO IF WE LOAD THE
*  REGISTERS IN ORDER CONTROL, INTERVAL, RECEIVE, XMIT AS EACH IS LOADED
*  IT RESETS THE CORRESPONDING BIT AND WE DON'T HAVE TO FOOL WITH THEM.
*
*      LOAD CONTROL REGISTER
*
*      7 6 5 4 3 2 1 0
*
*      BITS 7,6 => 10 : 1 STOP BITS
*      BITS 5,4 => 00 : NO PARITY
*      BIT  3   => 0  : 9902 CLK=1MHZ
*      BIT  2   => 0  : NOT USED
*      BITS 1,0 => 10 : 7 DATA BITS
*        THIS IS >82
*
       LDCR @CNTRL,8       LOAD IT. SEND THE MOST SIGNIFICANT 8 BITS OR >82

       SBZ  LDIR           DISABLE INTERVAL REGISTER
*
*      LOAD THE XMIT AND RCV DATA REGS
*
*     1200 BAUD = >1A1 = 1199.04 BPS
*
       LDCR @BAUD,12       LOAD THEM BOTH  12 BITS

       SBO  15             PLACE THE 9902 IN THE TEST MODE. TAKE OUT TO USE WITH
*                           ANOTHER RS232 DEVICE

****************************************
*                                      *
* TRANSMIT A CHARACTER FROM KEYBOARD   *
*                                      *
****************************************

XMIT   LI   R0,160         SCREEN POSITION OR'D WITH >4000 FOR WRITE

XMIT1  BLWP @KSCAN         SCAN THE KEYBOARD
       MOVB @STATUS,@STATUS HAS A KEY BEEN PRESSED?
       JEQ  XMIT1
       MOVB @KEYVAL,R1     GET THE VALUE OF THE KEY PRESSED
       CI   R1,>0500       QUIT PRESSED?
       JEQ  EOJQ

       BL   @SND232        TRANSMIT THE CHARACTER

       BL   @RCV232        RECEIVE THE CHARACTER

       BLWP @VSBW          WRITE THE CHARACTER TO THE SCREEN
       INC  R0             COUNTER FOR SCREEN LOCATION
       CI   R0,224         ALLOW TWO LINES OF TEXT
       JEQ  XMIT           START BACK AT THE FIRST SCREEN POSITION
```

---> continued

```
       JMP   XMIT1          STAY IN THE LOOP

EOJQ   SBZ   LED            TURN OFF PRETTY LIGHT
       LWPI  GPLWS          QUIT THE PROGRAM
       SB    @>837C,@>837C       CLEAR THE STATUS
       B     @>0070         BRANCH TO THE GPL INTERPRETER

****************************************
*                                      *
*    RECEIVE FROM RS232 (RCV232)       *
*                                      *
****************************************
RCV232 TB    RBRL           CHAR READY?
       JNE   RCV232         FORCES THE PROGRAM TO WAIT FOR THE CHARACTER
       CLR   R1             JUST SO YOU KNOW IT REALLY WORKS CLR THE OLD R1 VALUE
       STCR  R1,8           GET THE CHARACTER
       SBZ   RIENB          RESET RBRL
       RT                   GO HOME RCV232

*******************************************
*
* SEND OUT RS232 (SND232)
*
********************************************
SND232 SBO   RTSON          MAY WE SEND?
SNDO   TB    XBRE           IS THE TRANSMIT BUFFER EMPTY?
       JNE   SNDO           NO, TRY AGAIN
       LDCR  R1,8           TRANSMIT MSB OF R1
       SBZ   RTSON          INTERRUPT OFF
       RT                   GO HOME SND232

       END
```

======================eof====================

# MUSIC

PROGRAMMING MUSIC THE EASY WAY
PART 2
by Jim Peterson

In Part 1 I showed you how to set up a musical scale to create notes, and how to merge in various little routines to create a variety of musical effects, but I didn't tell you how to figure out what numbers to put in between those GOSUBs. So, here is the little program that makes it all easy.

```
100 CALL CHAR(127,"000F080F0
868F870000F08080868F87000080
8080868F870000808080868987O"
):: CALL CHAR(131,"000000000
0609070")
110 CALL CHAR(132,"0000120C4
8302040000000221C0810200000201
020103020000003CFF"):: CALL
 CHAR(136,"000000FF3C")
120 CALL CLEAR :: S$="GFEDCB
A" :: CALL CHAR(45,"00000000
FF"):: A$=RPT$(S$,3):: FOR R
=2 TO 22 STEP 2 :: IF R=12 T
HEN 130 :: DISPLAY AT(R,1):R
PT$("-",28)
130 NEXT R :: CALL CHAR(98,"
0020202834242830")
140 FOR R=1 TO 21 :: DISPLAY
 AT(R,1):SEG$(A$,R,1);:: NEX
T R
150 DATA 127,127,128,128,129
,129,130,130,131,131
160 DATA 1/16,1/8,1/4,1/2,1/
1
170 FOR R=1 TO 20 STEP 2 ::
READ N :: DISPLAY AT(R,15):C
HR$(N);:: NEXT R :: FOR R=3
TO 19 STEP 4 :: DISPLAY AT(R
,16):".";:: NEXT R
180 C=132 :: FOR R=1 TO 17 S
TEP 4 :: DISPLAY AT(R,17):CH
R$(C);:: C=C+1 :: NEXT R
190 FOR R=1 TO 17 STEP 4 ::
READ M$ :: DISPLAY AT(R,20):
M$;:: NEXT R
200 DATA 35,33,32,30,28,27,2
5,23,21,20,18,16,15,13,11,9,
8,6,4,3,1
```

```
210 FOR R=1 TO 21 :: READ N
:: N$=N$&CHR$(N):: DISPLAY A
T(R,6):STR$(N);:: NEXT R
220 G$="b" :: Z=-1 :: GOSUB
320 :: IF F=0 THEN 230 ELSE
GOSUB 330 :: GOTO 240
230 G$="#" :: Z=1 :: GOSUB 3
20 :: IF F<>0 THEN GOSUB 330
240 DISPLAY AT(24,1):"Shorte
st note? 1/" :: ACCEPT AT(24
,18)VALIDATE("12468")SIZE(2)
BEEP:L :: T$="1/"&STR$(L)::
RESTORE 160 :: FOR J=1 TO 5
:: READ L$ :: IF L$=T$ THEN
260
250 NEXT J :: GOTO 240
260 DISPLAY AT(24,1):"Is it
dotted? Y/N" :: ACCEPT AT(24
,19)VALIDATE("YN")SIZE(1):D$
 :: D=1-(D$="Y")
270 T=-3+J*4
280 FOR R=T TO 19 STEP 4 ::
DISPLAY AT(R,11):STR$(D);::
DISPLAY AT(R+2,11):STR$(D*1.
5);:: D=D*2 :: NEXT R
290 GOTO 360
300 FOR R=1 TO 20 STEP 2 ::
READ N :: DISPLAY AT(R,15):C
HR$(N);:: NEXT N
310 GOTO 310
320 DISPLAY AT(24,1):"How ma
ny "&G$&" on upper scale?" :
: ACCEPT AT(24,28)VALIDATE("
01234567")SIZE(1)BEEP:F :: R
ETURN
330 Y$="" :: FOR J=1 TO F ::
 DISPLAY AT(24,1):"On which
letter?"
340 ACCEPT AT(24,18)VALIDATE
(S$)SIZE(1)BEEP:L$ :: IF POS
(Y$,L$,1)<>0 THEN 340 ELSE Y
$=Y$&L$
350 S=1 :: FOR K=1 TO 3 :: P
=POS(A$,L$,S):: DISPLAY AT(P
,2):G$;:: DISPLAY AT(P,6):ST
R$(ASC(SEG$(N$,P,1))+Z);:: S
=P+1 :: NEXT K :: NEXT J ::
RETURN
360 OPEN #1:"PIO" :: FOR R=1
 TO 22 :: FOR C=3 TO 30 :: C
ALL GCHAR(R,C,G):: CALL HCHA
R(R,C,30):: R$=R$&CHR$(G)::
NEXT C :: PRINT #1:R$ :: R$=
"" :: NEXT R :: STOP
```

Get yourself a piece of sheet music and compare it to the screen display from that program. You will see that music is written on two sets of 5 lines. The upper set is marked at the left end with something like a fancy script capital S; it is used to write the higher notes, including the melody, which a pianist plays with the right hand. The lower set, marked with a sort of a backward C, contains the low notes played with the left hand. Your sheet music probably has a wide space between the sets, to make room for the lyrics, but there are really only three notes between them.

The screen display shows letters at the left, which are not on the sheet music. Those are the names of the notes, which we will have to refer to a couple of times to get started; observe that the notes are named A through G and then repeated.

The numbers along the left side are the numbers you will key in to play those notes. However, the screen display is set up in the key of C, which is played entirely on the piano white keys. The sheet music you want to program from may be in a different key, so -

The computer is asking you how many there are of something that looks like a squashed lower case b - I guess that's why they call it a flat? It means that the note will be played a bit lower, on the black key just left of the white key - and we will program it one number lower. So, look next to that capital S and see how many flats there are. If none, type 0. Otherwise, the computer will ask which letters they are next to. Type them in, one at a time, and presto - the computer will put them on the staff and adjust the numbers accordingly.

If there were no flats, the computer will want to know if there are any sharps - those are what you get by typing a shift 3 on the keyboard, and they mean that the note is played on the black key above the white key, and is programmed one number higher.

Now, the computer needs some information in order to help you set up the length of your notes - how long they are sounded. The various notes are depicted at the right. A 1/16 note is a little black egg with a stem (it may go up or down, makes no difference) and two flags on the stem. A 1/8 has only one flag and a 1/4 note has none. A 1/2 note is a hollow egg with a stem and a whole note has no stem.

Those little doodads to the right of the notes are rests, used to indicate a silent pause of the same length as that note - more on that later.

Look through your sheet music and find the shortest note. Tell the computer. It will want to know if any of those shortest notes are dotted - have a little dot to their right, as the screen display shows. A dotted note is played half again as long as normal. Presto again, the computer will show you the duration number to key in for each note. Then, if you have a printer attached, it will print out an XBasic screen dump of that screen - you will have to squash your own b's and sketch in the notes and rests.

If your software library contains an assembly screen dump, delete that last program line and put in a CALL INIT, CALL LOAD and CALL LINK to get a better printout - or ask me for it. If you don't have a printer, why not copy those numbers right onto the corresponding lines and spaces on your sheet music, and number some of the notes.

Now we're ready to make music! Let's keep it simple at first, just a single note melody - and I hope you picked a simple piece of music. Clear the TI's brain with NEW, then merge in that line 100 scale from part 1 by MERGE DSK1.SCALE . In the same way, merge in one of those line 1000 CALL SOUND routines. Put in a temporary stopper line 999 STOP, and a line 110 D=200 to set the duration.

The melody is almost always on the upper set of 5 lines. If a note has 2 or 3 eggs on its stem, as they usually do, the upper one is the melody note - we will get into harmony later.

Start with line 110. Check your chart to see what number denotes the length of the first note - maybe 2, if so key in T=2 :: Then check to see what number applies to the position of the upper egg of that note. Maybe 22, so key in A=22 :: GOSUB 1000 Enter RUN, and if you've done everything correctly, you will hear the note. You might decide already that you want to change that 200 in line 110.

Now for the second note. If it is of the same length as the first, you don't have to type anything - that's what makes this shorthand method so quick and easy. If the note position is also the same, you don't key that in either - just another GOSUB 1000.

If you have EZ-KEYS or another "hot keys" program, you can program a control key to put in the GOSUB 1000 with just one keypress - wish I had thought of that when I was programming music by the diskfull!

So keep plugging along, keying in durations and notes. After every half dozen notes or so, type RUN to see if everything sounds OK so far - it's easier to catch errors before they are too far back in the music.

You can get up to 5 screen lines on one line number, but you might better stick to 3 lines. You will note that the sets of notes are divided by vertical bars. You might program the notes between bars on a separate line, then add a ! followed by the words of the song that go with those notes - I find that a very good way to track down sour notes.

Regarding those bars - it might help you sometime to know this. At the beginning of the music, right after the big script S and the flats and sharps, you will see something like a 3 over a 4, or a 4 over a 4, or whatever - but often a symbol such as a barred C is used instead.

A 3 over a 4, for instance, means that the notes between two of those bars will add up to 3/4 - might be three quarter notes, or two eighth notes and two quarter notes, or whatever, but they will add up to 3/4. Sometimes the very first notes will add up short, but in that case the very last ones will make up the difference.

The notes between those two bars make up a bar of music, and the emphasis is on the first note - for instance, that 3/4 is the 1-2-3, 1-2-3 beat of waltz time.

While you are keying in that music, you might come to one of those rests. You can just key in its T= value and then A=0 for a silent note. However, computer notes stop so abruptly that somehow a rest just doesn't sound right, so I often just use the previous note instead.
You may come across one of those flat or sharp symbols next to a note in the music. Give the note a number 1 lower if a flat, one higher if a sharp, and the same for any subsequent occurrences of that note, until you find next to it a symbol that looks like the sharp sign with half its legs knocked off; that means to go back to normal. You might also come across that symbol to tell you to play a normally flat or sharp note as if it was not.

I think that covers all that you absolutely have to know for now, and I have horrified all serious students of music just about enough. There are all kinds of other squiggles on the sheet music but usually they are not essential in programming music.

There is one other time-saving shortcut that I should tell you about right now. Most music consists at least partly of musical phrases, of a series of notes, which are repeated two or more times within a melody. So, the first thing you should do before you start programming a song is to search through the music for such phrases.

If you find one, of more than a few notes, that is repeated elsewhere - and make sure it is repeated exactly the same - mark it off each place it occurs and label it 500. If you find a second repeating phrase, label it 600, and so on.

Then, when you start programming, start with line 500, key in that series of notes first, and end it with RETURN. If you have another phrase, put it in lines starting with 600, again ending with RETURN.

Now, start programming from the beginning of the song in line 120, but when you come to one of those phrases, just put in GOSUB 500 - the program will jump to that line number, play those notes, and come right back to where it was.
In Part 3, we will get into programming in 3-part harmony, bass notes, auto-chording, and all kinds of things.

## TIPS FROM THE TIGERCUB

NO #7

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

Here's a challenge for you amateur hackers. In Extended Basic you can write 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. In Basic that would take several lines of programming - or would it? Can you do it in one line?

Some of the deepest secrets of the TI are being discovered and published by Craig Miller in "The Smart Programmer". They are already appearing on bulletin boards and elsewhere without credit to Craig. Perhaps the most valuable of them is:

```
CALL INIT ::
CALL LOAD(-31806,16)
```

Type that in before you begin programming, and you will never again lose a program to that infernal FCTN = key! Also, put it in as one of the first program lines and your program will be kidproofed against the open-palm press-all key board technique.

Another extremely useful one is:
CALL PEEK(-28672,A) If the Speech Synthesizer is attached, A will equal 96, otherwise it will equal 0. If you are putting optional speech in an Extended Basic program, you can avoid those silent pauses by putting that CALL PEEK at the beginning of the program, and then a line with IF A=0 THEN before each CALL SAY to skip over it if the synthesizer is not on.

I have also found good use for Craig's routine CALL INIT :: CALL LOAD(-31888,63,255)
This shuts down all of the Disk Drive files and makes it possible to load programs over 12K long from tape, and then copy them back to tape, without having to physically disconnect the Disk Drives.

I'm now receiving about 40 of the User's Group newsletters, and some of them contain invaluable tips that deserve wider circulation. Here's one from Doug German in the Central Iowa 99/4A UG "4A Forum" -

```
100 PRINT "Press Fire Button
to continue"
110 CALL KEY(1,K1,S)
120 CALL KEY(2,K2,S)
130 IF K1+K2<2<>17 THEN 110
140 JS=INT(K1/18+K2/9+1)
```

Now, when you program CALL JOYST(JS,K,S) the program will respond to whichever of the two joysticks you were holding when you pressed the fire button!

In previous Tips, I reported that if you absentmindedly typed OLD CS1 instead of SAVE CS1, and pressed Enter, you can save your program by typing Shift E, Enter, get an I/O error and start over. It turns out that this will work in Basic but will not work in either version of Extended Basic unless you have the Memory Expansion.

I am now the proud owner of a P-Box, disk drive, and Memory Expansion, so let me pass on a few things I've already learned the hard way

If you have keyed in a program with the disk drive turned off, DO NOT try to turn it on and save your program - you will lock up the system, will have to shut down and lose the program. You must save the program to tape, then turn on the drive, reload the program from tape and save it to the disk.

If you are trying to load a program from tape and you get the ERROR FOUND IN DATA message at the very beginning, it means that your program is too long - the disk drive has stolen more

2000 bytes of your memory.

If you just want to get the program into the computer, you can disconnect the disk drive (It doesn't help to just turn it off), or use Craig Miller's routine described on the preceding page. If you want to get the program onto a disk, you might be able to do so by first typing CALL FILES(1) - but you probably won't be able to get it back from disk to tape.

If the program is still too long, you can still get it onto a disk if you have Extended Basic and the memory expansion. The method recommended by Texas Instruments involves re-typing them. My method is slow, but it eliminates the retyping.

Proceed as follows:
Shut down or disconnect all disk drives.
Load the program from tape (in Extended Basic if possible because you can delete lines so much faster).
Delete the first 50 lines.
Save the shortened program to another tape.
Turn on the disk drive, and memory expansion.
Load the shortened program in Extended Basic.

Save it to a disk with the merge option.
Shut down the drives again.
Load the original program from tape.
Delete the last 50 lines.
Save it to a tape.

Turn on the disk drive and memory expansion.
Load it in in Extended Basic.
Use the MERGE command to merge the previous shortened program into this one.
Save the merged program to a disk.
It will show up on the catalog as a file but it will load and run as a program. Don't give it to anyone who does not have Extended Basic and Memory Expansion!

According to Duane Fischer in the 99/4 Users of America newsletter, there are two models of the new white-plastic TI-00/4A.

They can only be identified by the copyright notice on the bottom ridge edge of the title screen.

It will read either (c) Texas Instruments 1981 or (c) Texas Instruments V2.2 - and if it is the latter, it is one of those which were gimmicked so they would not run the command modules produced by Funware, Romox, Atari or Parker Bros.

If you are having trouble determining locations on the screen during programming, put on a temporary grid with

```
1 CALL CHAR(32,"FF80808080808080")
```

If it won't interfere with your graphics, you can even number the grid:

```
2 FOR J=1 TO 32
3 C=C+1-ABS(C=9)*10
4 CALL HCHAR(24,J,48+C)
5 IF J>24 THEN 7
6 CALL VCHAR(J,14,48+C)
7 NEXT J
```

## TIPS FROM THE TIGERCUB
## NO #8

Last challenge was to 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, in just one line of Basic.
That didn't seem to stir up much interest but

anyway, here's my solution- Y/VAL(SEG$("07331921",X*2-1, 2)) You can even put that in a DEF and then use Y repeatedly without further defining it. Don't be putting TI Basic down - It's powerful stuff.

So here's the challenge for this month. Can you write a program in two lines of Basic, or one line of Extended Basic, to compose and play random music in 2-part harmony in the key of C?

I'm not talking about just any old random frequencies, I want random music composed of pairs of notes that belong together in chords of the key of C! If you figure it out, write me or give me a call.

Above we passed on Duane Fischer's tip that TI-99/4A consoles which display V2.2 instead of 1981 on the title screen will not run 3rd party command modules. It is now reported that the white modules with 1983 on the title screen will give the same problem.

If you have set up your equipment, tried to load a program, and the screen prints out "21D cS1" or something else goofy, don't panic - just unplug the cassette interface cable from the joystick port and plug it in where it belongs!

Did you ever absentmindedly type SAVE CS1 instead of OLD CS1, push RECORD, and not realize it until you had erased a program from your cassette? Did you know that the cassette has two tabs on the back edge that can be removed to keep that from happening?

Just slip the tip of a knife blade under them, and pry up to snap them off. Each side of the cassette is protected by the tab on its back left edge; when the tab is removed, the recorder's RECORD button can't be pushed down. Later on, if you do want to record over that side, just put a bit of scotch tape over the hole.

According to Michael A. Covinton in Compute! of Dec "82, if you are using a black and white TV for a monitor you can get a sharper screen by starting your program with a line 1 CALL SCREEN(15).

```
100 REM DOLLARS AND CENTS P
RINTER DEVELOPED FROM A MICR
OSOFT ONE-LINER BY CHUCK EME
RSON-HENRY IN COMPUTE! 2/84
110 REM TO GENERATE PRICES T
O DEMONSTRATE
120 A+INT(10*RND)+.1*INT(5*R
ND)
130 REM ONE-LINER TO PRINT
140 PRINT "$"&SEG$(STR$(A+.0
01),1,LEN(STR$(A+.001))-1)
150 GOTO 120
```

```
100 !Where did this come fro
m?
110 FOR CH=24 TO 30 :: CALL
HCHAR(1,1,CH,768):: CALL COL
OR(0,5,11):: NEXT CH :: GOTO
110
```

ASCII codes 1 through 31 can be placed on the screen by CALL HCHAR or CALL VCHAR. ASCII 30 is the cursor which prints out as that black square. The others do not represent a character, cannot be redefined, and are mostly blank, but ASCII codes 24 through 27 often contain odd bits of graphics ASCII codes 1 through 23 are always transparent.

Codes 24 through 31 are in character set 0 which is normally transparent, but if you are in Extended Basic it can be colored.

Some of the books of TI programs are full of stuff that is not worth keying in, but there is at least one good one, called Terrific Games for the TI-99/4A by Hal Renko and Sam Edwards-

programmed in the Netherlands, published in Great Britain and printed in Finland!
The publisher is Addison-Wesley, 1983.

If you are making a back up disk and you accidently initialize the Master instead of the copy disk, you will lose the data on the Master. If you key in a program, try to save it to a disk, and find that not enough sectors are available, you had better have another initialized disk or a cassette recorder available.
The solution to both programs - when you buy a package of disks, initialize them all right away! And watch to make sure that they don't have any bad sectors.

Here is a handy debugging routine. Right after the first CALL CLEAR, put in these temporary lines:

```
100 FOR @=1 TO 4
102 CALL COLOR(@,16,1)
103 NEXT @
104 GOTO 104
```

Then type LIST, as soon as the first lines have scrolled to the top of the screen, stop the list with FCTN 4.

Type RUN.

All of the numbers and punctuation will turn white.

Check for l's instead of 1's and o's instead of 0's and vice versa, equal numbers of opening and closing parentheses, misplaced commas, etc.

Then break with FCTN 4, LIST (The last line on the screen) and hyphen, ENTER, stop it again etc.

Yes, TI Basic will let you use @ in a variable name. I never use it in a program, but I do try to remember to use it in temporary debugging lines, in utility routines which I will save to merge or build programs around, in modifying other people's programs, etc. That way, I don't breed new bugs by duplicating a variable name that is already in the program.

STOP THE PRESS!! I have just received a review copy of a new book called:
THE TI-99/4A IN BITS & BYTES
by Remo A. Loreto
This one is worth the money! It is a thick book of 142 big pages. The first 20 pages are filled with explanations of programming statements and methods, with examples.

The rest of the book contains 50 programs for you to key in, and at an average of well over 2 pages per program these are definitely not merely short routines. There is a wide variety of games, education and utility programs, all original, written by Mr. Loreto's associates.

I have not had time yet to key any of them in, but have previously seen a few of them run, and they were good! The programs are reproduced from listings, for accuracy, and listed in large print in 28- column format for easy copying - Don't you wish everyone did that?

## TIPS FROM THE TIGERCUB NO #9

The above challenge was to write a program in two lines of Basic, or one line of Extended Basic, to compose and play random music in 2-part harmony in the key of C. I did't hear from anyone else, so here's my solution-

```
100 REM - TIGERCUB 1-LINE MU
SIC COMPOSER
```

```
110 CALL SOUND(-999,VAL(SEG$
("262262294330349392440494523587659698784",INT(12*RND+1)
*3-2,3)),0,VAL(SEG$("1311751
96",INT(3*RND+1)*3-2,3)),5)
120 GOTO 100
```

Just another challenge - can you write a one-line program in Extended Basic which will take only 70 seconds to scramble the numbers from 1 to 255 into a completely random sequence without duplication?

Who needs line numbers, anyway? Try keying this in- (in Extended Basic)

```
DIM S(36):: F=262 :: FOR N=1
 TO 36 :: S(N)=F*1.059463094
^N :: NEXT N (Enter)
FOR J=1 TO 1000 :: CALL SOUN
D(-99,S(INT(35*RND+1)),0)::
NEXT J (Enter)
```

Here's another one - key this in to your friend's computer while he is getting you a beer-

```
M$="0018243C425A667E"(Enter)
FOR C=128 TO 143 :: FOR L=1
TO 6 :: C$=C$&SEG$(M$,INT(8*
RND+1)*2-1,2):: NEXT L :: CA
LL CHAR(C,"00"&C$):: C$="" :
: NEXT C :: CALL CLEAR
```

Now, when he gets back with the beer, rest your left pinkie on the CTRL key while you type in any of the letters A through O, and show him that his computer has a built-in Mongolian alphabet!

One More-

```
FOR CH=65 TO 79 :: CALL CHAR
PAT(CH,CH$):: FOR J=1 TO 16
STEP 2 :: X$=SEG$(CH$,J,2)&X
$ :: NEXT J :: CALL CHAR(CH+
64,X$:: X$="" :: NEXT CH
Enter)
CALL CLEAR (Enter)
```

Again, hold down the CTRL key while you type letters between A and O. You can also change that to read FOR CH=33 TO 90 and CALL CHAR(CH,X$), omit the CALL CLEAR, and watch the fun on the screen.

If you are programming for speed, don't use DEF! Put a stopwatch on this routine and see why -

```
100 DIM N(100)
110 FOR J=1 TO 100
120 N(J)=RND*10
130 NEXT J
140 INPUT DUMMY$
150 DEF RD=RND*10
160 FOR J=1 TO 100
170 N(J)=RD
180 NEXT J
```

Here's a one-line GOSUB for your business programs, to give you the number of days (D) in any month (M) of the year (Y) including the extra day in February of Leap Year -

```
100 D=VAL(SEG$("312831303130
313130313031",M*2-1,2))+(ABS
(M=2)*ABS(Y/4=INT(Y/4)))
```

If you accidently hit a letter key in response to an INPUT N request for a numeric variable value, you get a nasty burp and a severe reprimand -
WARNING INPUT ERROR IN......
This is annoying to a user disconcerting to a non-user, and even frightening to a child.

If you program INPUT N$, the computer will accept any thing as the value of a string variable, and the VAL function will change a string of numeric value - but if the string contains anything non-numeric, the program will crash- which is even more annoying. disconcerting, and/or frightening! The solution? Tigercub presents-

```
100 REM - THE TIGERCUB POLIT
E COMPUTER
110 INPUT "TYPE A NUMBER, PL
EASE":N$
120 FOR J=1 TO LEN(N$)
130 IF POS("1234567890",SEG$
(N$,J,1),1)<>0 THEN 160
140 PRINT "THAT IS NOT A NUM
BER"
150 GOTO 110
160 NEXT J
```

```
170 N=VAL(N$)
```

If you want to accept decimals and negative numbers, change the string in line 130 to "1234567890.-"

Do you like to work those letter substitution puzzles in the newspapers and puzzle books? Why not let your computer make them for you? Just get anyone to type a message to be encoded.

```
100 REM - TIGERCUB CRYPTOCOD
ER by Jim Peterson
110 CALL CLEAR
120 T$="THIS PROGRAM WILL CR
EATE A  CRYPTOGRAM BY SUBSTI
TUTING  ONE LETTER FOR ANOTH
ER."
130 GOSUB 450
140 DIM A$(26,2)
150 M$="ABCDEFGHIJKLMNOPQRST
UVWXYZ"
160 FOR T=26 TO 1 STEP -1
170 A$(T,1)=CHR$(T+64)
180 RANDOMIZE
190 X=INT(T*RND+1)
200 A$(T,2)=SEG$(M$,X,1)
210 IF A$(1,2)="A" THEN 150
220 IF A$(T,2)=A$(T,1)THEN 1
90
230 M$=SEG$(M$,1,X-1)&SEG$(M
$,X+1,LEN(M$))
240 NEXT T
250 FOR J=1 TO LEN(T$)
260 D=ASC(SEG$(T$,J,1))-64
270 IF (D<1)+(D>26)THEN 300
280 C$=C$&A$(D,2)
290 GOTO 310
300 C$=C$&SEG$(T$,J,1)
310 NEXT J
320 T$=C$
330 C$=NUL$
340 GOSUB 450
350 IF FL=0 THEN 370
360 GOTO 360
370 FL=1
380 FOR D=1 TO 500
390 NEXT D
400 CALL CLEAR
410 PRINT "TYPE YOUR MESSAGE
 OF NOT":"MORE THAN 4 LINES.
 USE EXTRA":"SPACES TO AVOID
 BREAKING A":"WORD AT THE EN
D OF A LINE.":" THEN ENTER"
420 INPUT "
           ":T$
430 CALL CLEAR
440 GOTO 150
450 R=5
460 C=3
470 FOR J=1 TO LEN(T$)
480 CALL HCHAR(R,C,ASC(SEG$(
T$,J,1)))
490 C=C+1
500 IF C<31 THEN 530
510 C=3
520 R=R+1
530 NEXT J
540 RETURN
```

ALMOST OUT OF MEMORY, so

HAPPY HACKIN'
Jim Peterson

## TIPS FROM THE TIGERCUB
## #29
Copyright 1985

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

Re TI*MES #38:
I goofed again. In the I/O ERROR routine in Tips #28, the ON ERROR STOP will do no good in the place where I put it. It should be placed after the file is opened in line 100 so that it will become the current error trap if the file is opened correctly.

And the CALL KEY example in Tips #28 will look better if R=14. A couple of very knowledgeable programmers have written to tell me that I was wrong, and the manual is right, about CALL KEY status -1. They say that -1 simply means that the same key is being pressed as was pressed during the last keyscan, and that it could have been released and repressed in the interim.

This may be, but try this routine and see if you can release and repress a key without getting a status code 0 (no key pressed) and status code 1 (different key pressed) before another status code -1.

```
100 CALL KEY(0,K,S):: PRINT
K,S :: GOTO 100
```

George Steffen has responded

to the challenge in the last Tips, by publishing in the LA 99ers TopIcs a remarkably compact routine to translate the internal format string representation of numeric data back into numbers. The following lines will update the Menu Loader accordingly.

```
100 !by A. Kludge/M. Gordon/
T. Boisseau/J. Peterson/G. S
teffen/etc.Version #8, 11/85
140 @,@@,A,A$,B,C,D$,E,F,FLA
G,I,J,K,KD,KK,M,M$,N$,NN,P,P
$,PG$(),PP,PP$,Q$,S,ST,T$(),
TT,VT,V(,),W$,X,X$,Y,K2,S2
810 F=1 :: E=ASC(SEG$(M$,1,1
)):: M=ASC(SEG$(M$,2,1)):: I
F E=0 AND M=0 THEN GOTO 817
ELSE IF E>128 AND M>128 THEN
 F=-1 :: E=255-E :: M=256-M
815 FOR I=1 TO 6 :: M=M+(ASC
(SEG$(M$,I+2,1)))/100^I :: N
EXT I :: M=M*F*100^(E-64)
817 PRINT #PP:M
870 FOR P=1 TO NN-1 :: PRINT
 #2:PG$(P);TAB(15);V(P,3);TA
B(20);T$(ABS(V(P,1)));TAB(25
);V(P,2);TAB(31);CHR$(89*ABS
(V(P,1)<0)):: NEXT P :: CLOS
E #2
```

The change in the last line is my own, because it was pointed out to me that the catalog output to the printer did not indicate protected files.

That last line is a good example of the power of relational expressions to accomplish compact programming.

The variable V(P,1) picks up its value from the variable A which is read from the disk directory in line 350. This is a number from 1 to 5, indicating the type of file, and if the file is write-protected the number is negative.

A true expression has a relational value of -1. If the file is protected, V(P,1)<0 is true, and its value is -1, converted by ABS to +1 and multiplied by 89 to give ASCII 89, converted by CHR$ to "Y".

If not protected, V(P,1) is a positive number, V(P,1)<0 is false and has a relational value of 0; 89 times 0 is still 0, and CHR$(0) prints nothing.

George also mentioned in a letter:
that my remarks on the UPDATE mode applied only to VARIABLE files;

that RESTORE without a number,to return the record pointer to the beginning of a file, works only with VARIABLE files;

that RESTORE with a number works only with RELATIVE files;

and that therefore the only way to RESTORE a SEQUENTIAL FIXED file is to close it & reopen it.

On trying this out, I find that you can write to a FIXED SEQUENTIAL file and still be able to read the following records - but you can't simply "read a record, change it in some way, and then write the altered record back out on the file", as the Reference Guide indicates, because you will change the record FOLLOWING the one you read!

It is possible to UPDATE a FIXED SEQUENTIAL file without reading it all into an array and writing it back out, but you must read sequentially to the record you want, close the file, reopen the file, read back to the record just before the one you want to update, then write in the updated record.

I have received several other suggestions regarding the Menu Loader, too many to describe here. You can all modify it to your own tastes and needs. Remember to turn off the pre-scan and ON ERROR while you're working on it, then add any new variable names or CALLs to the pre-scan. And remember, that last line MUST be the LAST line of the program! You can resequence it higher, and change the GOTO accordingly, but don't put anything after it!

I did change my version to slash the zero, since this will carry over into a program that is loaded. If you do this, be sure to add a CALL CHAR to the list in line 150!

```
190 CALL CLEAR :: FOR S=1 TO
 14 :: CALL COLOR(S,7,16)::
NEXT S :: CALL COLOR(0,2,16)
:: CALL CHAR(48,"003A444C546
444B8")
```

When you just want to load a program, waiting for it to be read from the disk directory can be a drag. And, you may have trouble recognizing the filename. So, here is the Tigercub Quickloader which I have placed on all my Collection Disks.
First you will need Catwriter, another program that writes a program. This one will read the disk directory, ignore everything other than programs, ask you for a complete program name for each filename, and write all that into a MERGE format program called CATMERGE.
(a much improved version of this program was published in Tips #47. - Ed.)

```
100 !CATWRITER by Jim Peters
on
110 OPEN #1:"DSK1.",INPUT ,R
ELATIVE,INTERNAL :: INPUT #1
:N$,A,J,K :: OPEN #2:"DSK1.C
ATMERGE",VARIABLE 163 :: LN=
1000 :: FN=1100
120 X=X+1 :: INPUT #1:P$,A,J
,B :: IF LEN(P$)=0 THEN 160
:: IF ABS(A)=5 OR ABS(A)=4 A
ND B=254 THEN 130 ELSE X=X-1
 :: GOTO 120
130 DISPLAY AT(12,1)ERASE AL
L:P$;"    PROGRAM NAME?" ::
ACCEPT AT(14,1)SIZE(25):F$
140 PRINT #2:CHR$(INT(FN/256
))&CHR$(FN-256*INT(FN/256))&
CHR$(147)&CHR$(200)&CHR$(LEN
(F$))&F$&CHR$(0):: FN=FN+1
150 M$=M$&CHR$(200)&CHR$(LEN
(P$))&P$&CHR$(179):: IF X<11
 THEN 120
160 IF M$="" THEN 180
170 PRINT #2:CHR$(INT(LN/256
))&CHR$(LN-256*INT(LN/256))&
CHR$(147)&SEG$(M$,1,LEN(M$)-
1)&CHR$(0):: LN=LN+1 :: M$="
" :: X=0 :: IF LEN(P$)<>0 TH
EN 120
180 PRINT #2:CHR$(INT(LN/256
))&CHR$(LN-256*INT(LN/256))&
CHR$(147)&CHR$(200)&CHR$(3)&
"END"&CHR$(0)
190 PRINT #2:CHR$(255)&CHR$(
255):: CLOSE #1 :: CLOSE #2
```

Next, key in the Quickloader. Do not change the line numbers, do not RESequence, because CATMERGE will be merged into the middle of it and that last line must be the last. Then, enter MERGE DSK1.CATMERGE and then SAVE DSK1.LOAD .

```
100 CALL CLEAR :: DIM M$(48)
:: CALL CHAR(94,"3C4299A1A19
9423C"):: CALL SCREEN(2):: F
OR SET=1 TO 14 :: CALL COLOR
(SET,15,1):: NEXT SET :: DIS
PLAY AT(1,4):"TIGERCUB QUICK
LOADER"
110 X=X+1 :: READ M$(X):: IF
 M$(X)<>"END" THEN 110
115 CALL PEEK(8198,A):: IF A
<>170 THEN CALL INIT
120 R=3 :: FOR J=1 TO X-1 ::
 READ X$ :: DISPLAY AT(R,1):
STR$(J);TAB(4);X$ :: R=R+1 :
: IF R<23 THEN 150
130 DISPLAY AT(24,1):"CHOICE
? OR 0 TO CONTINUE 0" :: ACC
EPT AT(24,26)VALIDATE(DIGIT)
SIZE(-2):N
140 IF N<>0 THEN 160 :: R=3
150 NEXT J :: DISPLAY AT(24,
1):"CHOICE?" :: ACCEPT AT(24
,9)VALIDATE(DIGIT):N
160 IF SEG$(M$(N),LEN(M$(N))
,1)="*" THEN DISPLAY AT(12,1
)ERASE ALL:"Return to BASIC"
: :"Type OLD DSK1."&M$(N)::
```

```
STOP
170 CALL CHARSET :: CALL CLE
AR :: CALL SCREEN(8):: CALL
PEEK(-31952,A,B):: CALL PEEK
(A*256+B-65534,A,B):: C=A*25
6+B-65534 :: A$="DSK1."&M$(N
):: CALL LOAD(C,LEN(A$))
180 FOR J=1 TO LEN(A$):: CAL
L LOAD(C+J,ASC(SEG$(A$,J,1))
):: NEXT J :: CALL LOAD(C+J,
0):: GOTO 30000
30000 RUN "DSK1.1234567890"
```

If you don't want to give your Basic-only programs a filename ending in an asterisk, you can leave out that warning routine, or you can modify it to warn of E/A or MiniMemory programs. If Catwriter has picked up any unloadable program-format files, etc., just delete them from the DATA lines.

Gene Burchfield asked if I had a program to print banners vertically. I had never heard of such a thing, so I wrote one.

```
100 DISPLAY AT(12,1)ERASE AL
L:"TIGERCUB STREAMER PRINTER
" !by Jim Peterson
110 DATA 0000,0001,0010,0011
,0100,0101,0110,0111,1000,10
01,1010,1011,1100,1101,1110,
1111
120 RESTORE 110 :: DIM B$(16
):: FOR J=1 TO 16 :: READ B$
(J):: NEXT J :: P$(0)=" " ::
 P$(1)=CHR$(230)
130 INPUT "TEXT TO BE PRINTE
D? ":T$ :: PRINT :: INPUT "P
RINTER DESIGNATION? ":PD$ ::
 OPEN #1:PD$
140 PRINT :: INPUT "SIZE? (1
-10) ":Z :: IF Z<1 OR Z>10 T
HEN 140
150 FOR J=1 TO LEN(T$):: A=A
SC(SEG$(T$,J,1)):: IF A=32 T
HEN GOTO 200
160 CALL CHARPAT(A,H$):: FOR
 W=1 TO 15 STEP 2 :: K$=SEG$
(H$,W,2):: FOR L=1 TO 2 :: L
$=SEG$(K$,L,1):: B=POS("0123
456789ABCDEF",L$,1)
170 M$=B$(B):: FOR M=1 TO 4
:: N=VAL(SEG$(M$,M,1)):: N$=
N$&RPT$(P$(N),Z):: NEXT M
180 NEXT L :: FOR Q=1 TO Z/2
+.5 :: PRINT #1:TAB((81-Z*8)
/2+.5);N$ :: NEXT Q :: N$=""
 :: NEXT W :: FOR R=1 TO Z/2
+.5 :: PRINT #1:"" :: NEXT R
190 NEXT J :: STOP
200 FOR T=1 TO Z*4 :: PRINT
#1:"" :: NEXT T :: GOTO 190
210 CALL KEY(0,K,S):: IF S=0
 THEN 210 ELSE RETURN
```

If your printer doesn't have the special characters of the Gemini, substitute 88 instead of 230 in line 120, to print X's, or whatever else you want. If you do have the special characters, try some others, such as 239, for this and other graphics printing programs.

This routine will print a handy reference chart of them.

```
100 IMAGE ### #    ### #    ##
# #    ### #    ### #    ### #
110 P$=RPT$(CHR$(251)&CHR$(2
53),21):: X=0
120 OPEN #1:"PIO" :: PRINT #
1:CHR$(27);"E"
130 PRINT #1:P$:"  ASCII COD
ES FOR GEMINI SPECIAL CHARAC
TERS":P$
140 FOR J=160 TO 175 :: K=J-
X
150 PRINT #1,USING 100:K,CHR
$(J),K+16,CHR$(J+16),K+32,CH
R$(J+32),K+48,CHR$(J+48),K+6
4,CHR$(J+64),K+80,CHR$(J+80)
:: NEXT J
160 IF FLAG=1 THEN STOP ELSE
 FLAG=1 :: PRINT #1:"":"":P$
:"TI-WRITER CODES FOR GEMINI
 SPECIAL CHARACTERS":P$ :: X
=128 :: GOTO 140
```

Another one that just looks pretty -

```
100 !KALEIDOSPRITES by Jim P
eterson
110 CALL CLEAR :: FOR CH=100
 TO 128 STEP 4 :: FOR L=1 TO
 4 :: RANDOMIZE :: X$=SEG$("
0018243C425A667E8199A5BDC3DB
E7FF",INT(16*RND+1)*2-1,2)
120 B$=B$&X$ :: C$=X$&C$ ::
NEXT L :: CALL CHAR(CH,RPT$(
B$&C$,4)):: B$,C$="" :: NEXT
 CH :: Z=2 :: CALL SCREEN(5)
130 CALL MAGNIFY(Z):: K=1 ::
 FOR J=1 TO 7 :: S=96+4*J ::
 R=16*J :: C=100*RND+20
140 IF J>5 AND Z=4 THEN T=5
```

```
:: GOTO 160
150 T=INT(15*RND+2):: IF T=5
 THEN 150
160 CALL SPRITE(#K,S,T,R,C,#
K+1,S,T,177-R,C,#K+2,S,T,R,2
41-C,#K+3,S,T,177-R,241-C)::
 K=K+4 :: NEXT J
170 Z=INT(2*RND+1)*2 :: GOTO
 130
```

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

```
100 !DISK MATCHER by Jim Pet
erson
110 DISPLAY AT(8,9)ERASE ALL
:"DISK MATCHER": : : :" To c
ompare a backup disk":"with
a master and list any":"file
s found on one but not"
120 DISPLAY AT(15,1):"on the
 other.": : : :"         Press
 any key"
130 CALL KEY(0,K,S):: IF S=0
 THEN 130
140 DISPLAY AT(12,1)ERASE AL
L:"INSERT MASTER - PRESS ENT
ER" :: CALL KEY(0,K,S):: IF
S=0 THEN 140
150 OPEN #1:"DSK1.",INPUT ,R
ELATIVE,INTERNAL :: INPUT #1
:D1$,A,J,K :: DIM F1$(127)
160 X=X+1 :: INPUT #1:F1$(X)
,A,J,B :: IF LEN(F1$(X))<>0
THEN 160 ELSE CLOSE #1
170 DISPLAY AT(12,1)ERASE AL
L:"INSERT BACKUP DISK": :"PR
ESS ENTER" :: CALL KEY(0,K,S
):: IF S=0 THEN 170
180 OPEN #1:"DSK1.",INPUT ,R
ELATIVE,INTERNAL :: INPUT #1
:D2$,A,J,K :: DIM F2$(127)
190 Y=Y+1 :: INPUT #1:F2$(Y)
,A,J,B :: IF LEN(F2$(Y))<>0
THEN 190 ELSE CLOSE #1
200 DIM F(127):: FOR J=1 TO
X :: FOR L=1 TO Y :: IF F2$(
L)=F1$(J)THEN F(L)=1 :: GOTO
 220
210 NEXT L :: PRINT F1$(J);"
 NOT ON BACKUP"
220 NEXT J
230 FOR M=1 TO Y :: IF F(M)=
0 THEN PRINT F2$(M);" NOT ON
 MASTER"
240 NEXT M :: END
```

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

A very useful tip from Jim Swedlow, in the Orange County ROM newsletter - INPUT respects any trailing print separator on a preceding PRINT command. Try it -

```
100 PRINT TAB(20);:: INPUT B
```

$

## TIPS FROM THE TIGERCUB #30

Copyright 1986

Here's another Tigercub Challenge - can you run this and get these results?

```
>LIST
100 PRINT PI
110 PRINT MAX
120 PRINT PI
130 PRINT MAX
>RUN
 0
 0
 3.141592654

* SYNTAX ERROR IN 130
```

Some of you sharp-eyed newsletter editors may have noticed that this text is being hyphenated to avoid some of those gaping blanks that occur when only a few long words will fit on a right-justified line. The only way that I have found to accomplish this is to set the TI-Writer right tab for the actual column width to be printed and then, whenever a word is hyphen-ated, backspace and replace the blanks on that line with carets, adding enough extra carets to justify the line - like this -

whenever^a^word^^is^^hyphen-

It helps to go into fixed mode with CTRL 0 when you are inserting extra carets.

When using this method, it is also necessary to set the paragraph indentation with IN 0 on the command line; if indentations are desired, they can be filled with caret signs, like this:
^^When using this method,

I am told that my old 3D Sprite Routine made it to the Golden Quickies section of CompuServe, so here is an updated version. I have found that sprites can be controlled much more easily (although not moved as rapidly) with CALL LOCATE, rather than turning them loose with CALL MOTION and then trying to catch up with them!

```
100 CALL CLEAR :: CALL SCREE
N(5):: FOR SET=2 TO 8 :: CAL
L COLOR(SET,8,5):: NEXT SET
:: DISPLAY AT(3,12):"3-D SPR
ITE DEMO"
110 DISPLAY AT(22,1):"BY TIG
ERCUB" :: CALL CHAR(40,"FF81
818181818181FF8181818181818181FF
FF010101010101FF010101010101
01FF")
120 CALL CHAR(36,RPT$("F",64
)):: CALL MAGNIFY(4):: FOR X
=2 TO 22 STEP 2 :: CALL SPRI
TE(#X,36,X/2+1-(X>7)-(X>13),
32+X*6,40+X*6):: NEXT X
130 S=1 :: CALL SPRITE(#S,40
,16,46,7):: FOR C=6 TO 42 ST
EP 2 :: CALL LOCATE(#S,46,C)
:: NEXT C :: FC=44 :: FR=46
:: Y=0
140 FOR C=FC TO FC+44 STEP 2
 :: CALL LOCATE(#S,FR,C):: N
EXT C :: FC=FC+44 :: CALL SP
RITE(#S+2,40,16,FR,FC):: CAL
L DELSPRITE(#S):: TC=FC-32
150 FOR C=FC TO TC STEP -2 :
: CALL LOCATE(#S+2,FR,C):: N
EXT C :: TR=FR+34 :: FOR R=F
R TO TR STEP 2 :: CALL LOCAT
E(#S+2,R,TC):: NEXT R
160 CALL SPRITE(#S,40,16,TR,
TC):: CALL DELSPRITE(#S+2)::
 FR=TR :: TR=FR-72 :: FOR R=
FR TO TR STEP -2 :: CALL LOC
ATE(#S,R,TC):: NEXT R
170 CALL SPRITE(#S+2,40,16,T
R,TC):: CALL DELSPRITE(#S)::
 FR=TR :: TR=FR+50 :: FOR R=
FR TO TR STEP 2 :: CALL LOCA
TE(#S+2,R,TC):: NEXT R
180 Y=Y+1 :: IF Y=11 THEN CA
LL DELSPRITE(#S+2):: GOTO 13
0 ELSE S=S+2 :: FC=TC :: FR=
TR :: GOTO 140
```

Ian Swales in Belgium can write some of the most intricate routines, and pull them into the tightest knot. I had searched everywhere for a sorting routine for 2-dimensional arrays, and invented some ridiculous ones, before Ian sent me this jewel.

```
100 !DEMO of two-dimensional
 sorting routine
110 !Set up array to be sort
ed
120 CALL CLEAR :: DIM A$(20,
4):: RANDOMIZE :: DEF X$=CHR
$(26*RND+65)
130 FOR J=1 TO 20 :: A$(J,1)
=X$&X$&X$ :: A$(J,2)=STR$(IN
T(100*RND+1):: A$(J,3)=X$&ST
R$(INT(10*RND)):: A$(J,4)=IN
T(10*RND))&X$ :: NEXT J
140 INPUT "SORT BY?(1-4)":K
150 J=20 !2-dimensional arra
y sorting routine by Ian Swa
les
160 DIM Q(20):: FOR X=1 TO 2
0 :: Q(X)=X :: NEXT X
170 M=0
180 FOR X=1 TO J-1 :: IF A$(
Q(X),K)<=A$(Q(X+1),K)THEN 21
0
190 M=-1
200 T=Q(X):: Q(X)=Q(X+1):: Q
(X+1)=T
210 NEXT X
220 IF M THEN 170
230 FOR X=1 TO 20 :: FOR L=1
 TO 4 :: PRINT A$(Q(X),L);"
";:: NEXT L :: PRINT :: NEXT
 X :: GOTO 140
```

Did you ever need a routine that would accept either a string or a numeric value? Try this -

```
100 N=0 :: ON ERROR 110 :: A
CCEPT M$ :: N=VAL(M$):: GOTO
 120
110 ON ERROR STOP :: RETURN
120
120 ON (N=0)+2 GOTO 130,140
130 PRINT M$ :: GOTO 100
140 PRINT N :: GOTO 100
```

A useful tip from Stephen Shaw in England - if you have a long program which will run only in Basic, and which will load from disk with CALL FILES(1) but runs out of memory when you try to run it; and if you have the MiniMemory module -
Insert MiniMemory module, select Basic, enter CALL FILES(1), Enter NEW, enter OLD DSK1.(filename). When loaded, enter SAVE EXPMEM2.

When SAVEd, enter CALL LOAD(-31888,63,255), enter NEW, enter OLD EXPMEM2, and enter RUN. That is still a lot faster than loading a long program from tape!

Another reason for never using the default mode of so-called UPDATE when opening a file (without specifying INPUT or OUTPUT) is that you will get an I/O ERROR 01 if the file is write-protected.

To make a note to yourself while programming, just type 1! and whatever you want to make note of, then LIST "PIO":1, and then type 1 and enter to delete the line.

One of the very best tips for this month comes from Paul A. Meadows, in the September 85 newsletter of T.I.N.S. (Nova Scotia, Canada) -
How to print up to 132 characters in a line (condensed print, of course) out of TI-Writer! Just prepare your file as usual but in line 0001 put formatter commands such as .LM 10;RM 132; IN +5;FI;AD . The Fill and Adjust are necessary, the Indent is up to you, as are the left and right margins - but notice that right margin set way over at 132?
Now, instead of saving the file with SF, type PF and then C DSK1.(filename) to print to the disk. This not only strips out the control C characters, it also erases the TI-Writer tab line that was applied to the last line of the file.
So now, with your printer opened and initialized for condensed print, go into the TI-Writer formatter mode and print your file!

I have made the following changes to my working copy of the Tigercub Menuloader. This sets up my Gemini printer to skip over the perforations and print full page width in elite print with a wide left margin for ring-binder punching. Other printers may need changes in these codes.

```
620 DISPLAY AT(12,1)ERASE AL
L:"PRINTER? PIO" :: ACCEPT A
T(12,10)SIZE(-18):P$ :: GOSU
B 895 :: PP=3
840 DISPLAY AT(24,1):"PRINTE
R NAME? PIO" :: ACCEPT AT(24
,15)SIZE(-14):PP$ :: GOSUB 8
95 :: PRINT #2:SEG$(D$,1,4)&
" - Diskname= "&N$
895 OPEN #3:P$,VARIABLE 132
:: PRINT #3:CHR$(27);"B";CHR
$(2);CHR$(27);"M";CHR$(10);C
HR$(27);"N";CHR$(6):: RETURN
```

I always keep a backup of everything, on the flipped side of another disk, and I often want to verify that the backup has everything that is on the master, and vice versa.

```
100 DISPLAY AT(3,6)ERASE ALL
:"TIGERCUB DOUBLECAT": :" To
 compare the contents of": :
"a disk with a backup." !by
Jim Peterson
110 DISPLAY AT(12,1):"INSERT
 MASTER DISK": :"PRESS ENTER
"
120 CALL KEY(0,K,S):: IF S=0
 THEN 120
130 DATA DF,DV,IF,IV,P
140 RESTORE :: FOR I=1 TO 5
:: READ T$(I):: NEXT I
150 DIM F$(127):: OPEN #1:"D
SK1.",INPUT ,RELATIVE,INTERN
AL :: INPUT #1:A$,J,J,K :: F
$(0)=A$&" "&STR$(K)
160 X=X+1 :: INPUT #1:F$(X),
I,J,K :: IF F$(X)="" THEN 17
0 :: F$(X)=F$(X)&" "&T$(ABS(
I)):: GOTO 160
170 X=X-1 :: CLOSE #1 :: DIS
PLAY AT(12,1)ERASE ALL:"REMO
VE MASTER DISK": :"INSERT BA
CKUP DISK": :"PRESS ENTER"
180 CALL KEY(0,K,S):: IF S=0
 THEN 180
190 OPEN #1:"DSK1.",INPUT ,R
ELATIVE,INTERNAL :: INPUT #1
:A$,J,J,K :: DISPLAY AT(1,1)
ERASE ALL:F$(0);:: DISPLAY A
T(1,15):A$&" "&STR$(K);
200 Y=Y+1 :: R=R+1 :: GOSUB
290 :: INPUT #1:A$,I,J,K ::
```

```
IF A$="" THEN 260 :: K$=A$&"
 "&T$(ABS(I))
210 IF K$=F$(Y)THEN DISPLAY
AT(R+1,1):F$(Y);:: DISPLAY A
T(R+1,15):K$;:: GOTO 250
220 IF K$<F$(Y)THEN DISPLAY
AT(R+1,15):K$;:: Y=Y-1 :: GO
TO 250
230 DISPLAY AT(R+1,1):F$(Y);
:: R=R+1 :: GOSUB 290 :: Y=Y
+1
240 IF K$=F$(Y)THEN 210 ELSE
 IF K$<F$(Y)THEN 220 ELSE IF
 Y<X THEN 230 ELSE DISPLAY A
T(R,15):K$;
250 GOTO 200
260 IF Y>X THEN 280
270 R=R+1 :: GOSUB 290 :: FO
R J=Y TO X :: DISPLAY AT(R,1
):F$(J):: R=R+1 :: GOSUB 290
 :: NEXT J
280 DISPLAY AT(24,1):"      P
RESS ANY KEY" :: CALL KEY(0,
K,S):: IF S=0 THEN 280 ELSE
CLOSE #1 :: END
290 IF R<23 THEN RETURN
300 DISPLAY AT(24,1):"PRESS
ANY KEY" :: DISPLAY AT(24,1)
:" " :: CALL KEY(0,K,S):: IF
 S=0 THEN 300
310 CALL CLEAR :: R=1 :: RET
URN
```

And that is just about

MEMORY FULL!

Jim Peterson

## TIPS FROM THE TIGERCUB No. 68

Tigercub Software
156 Collingwood Ave.
Columbus, OH 43213
*********

My three Nuts & Bolts disks, each containing 100 or more subprograms, have been reduced to $5.00 each. I am out of printed documentation so it will be supplied on disk.

My TI-PD library now has almost 600 disks of fairware (by author's permission only) and public domain, all arranged by category and as full as possible, provided with loaders by full program name rather than filename, Basic programs converted to XBasic, etc. The price is just $1.50 per disk(!), post paid if at least eight are ordered. TI-PD catalog #5 and the latest supplement is available for $1 which is deductible from the first order.

When I have finished reading Barry Traver's column in Computer Monthly, I like to take a look at whatever Dr. Michael Ecker is up to in his "Recreational Computing" column, although much of his math is beyond me and I can't always translate his GW Basic into TI Basic.

In the February issue, he had a routine to play Fibonacci modular music. This is the TI version; it is not very musical, but the notes are in the chromatic scale.

```
100 A=0 :: B=1 :: M=51
110 C=A+B :: C=C-M*INT(C/M):
: CALL SOUND(-100,110*2^(C/1
2),5):: A=B :: B=C :: GOTO 1
10
```

Dr. Ecker also had a challenge to swap two numbers without using a third variable or the SWAP command - which TI Basic doesn't have anyway. The practical way, of course, is to use the 3rd variable, T=A :: A=B :: B=T, but just for the fun of it, if we are dealing with one-digit numbers -

```
100 A=1 :: B=2 :: A=A+B/10 :
: B=INT(A):: A=(A-INT(A))*10
 :: PRINT A;B
```

But suppose we are dealing with numbers of any length - we can still do it with a one-liner, or a two-liner if we want to input the numbers from the keyboard -

```
100 INPUT A :: INPUT B
110 B=B/10^(LEN(STR$(B)))::
A=A+B :: B=INT(A):: A=A-INT(
A):: A=A*10^(LEN(STR$(A))-1)
```

```
:: PRINT A;B :: GOTO 110
```

So you got smart and tried a negative number or a decimal? OK, how about this -

```
100 INPUT A$ :: INPUT B$
110 A$=A$&" "&B$ :: B$=SEG$(
A$,1,POS(A$," ",1)-1):: A$=S
EG$(A$,POS(A$," ",1)+1,255):
: PRINT A$;" ";B$ :: GOTO 11
0
```

And another challenge was to alternately assign X the value of A and B, without using IF...THEN or any outside help. That seems to require a two-liner -

```
100 A,X=77 :: B=132
110 X=ABS(X=A)*B+ABS(X=B)*A
:: PRINT X :: GOTO 110
```

Thanks to Bruce Harrison, here is a neat subprogram to sort strings into sequence as they are entered -

```
100 CALL CLEAR :: DIM W$(100
)
110 FOR J=1 TO N :: W$(J)=""
 :: NEXT J :: INPUT "N=? ":N
120 INPUT I$ :: IF I$="" THE
N 130 ELSE CALL INSORT(W$(),
I$,N):: GOTO 120
130 FOR J=1 TO N :: PRINT W$
(J):: NEXT J :: GOTO 110
30020 SUB INSORT(W$(),I$,N):
: FOR T=1 TO N :: IF I$>W$(T
)THEN 30030 ELSE 30040
30030 NEXT T :: GOTO 30050
30040 FOR J=N TO T STEP -1 :
: W$(J+1)=W$(J):: NEXT J
30050 W$(T)=I$ :: N=N+1 :: S
UBEND
```

In the test routine in lines 100-130, give N the value of 0, input some words and then just press enter.

To start a new array, use FOR J=1 TO N :: W$(J)="" :: NEXT J, then reset N to 0. If you want to sort in reverse sequence, change the > to <. If you need to sort numbers, delete all the $, change the "" in line 120 to 0, and input a 0 when you are when finished inputting.

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

Someone sent me a program to figure days between dates but it would not count leap dates, so I decided to write one that would.

```
100 DISPLAY AT(2,5)ERASE ALL
:"DAYS BETWEEN DATES":"":"
including leap year days" ::
 M$(1)="From" :: M$(2)="To
" :: R=13
110 DATA 31,28,31,30,31,30,3
1,31,30,31,30,31
120 DIM L(12):: FOR J=1 TO 1
2 :: READ L(J):: NEXT J
130 FOR J=1 TO 2 :: DISPLAY
AT(R-1,1):M$(J):"year      m
onth    day  " :: ACCEPT AT(
R,6)VALIDATE(DIGIT)SIZE(4):Y
(J)
140 ACCEPT AT(R,17)VALIDATE(
DIGIT)SIZE(2):M(J):: IF M(J)
<1 OR M(J)>12 THEN 140
150 ACCEPT AT(R,24)VALIDATE(
DIGIT)SIZE(2):D(J):: IF D(J)
<1 OR D(J)>31 THEN 150
160 CALL LEAP(Y(J),X):: L(2)
=L(2)-X :: IF D(J)>L(M(J))TH
EN 150
170 L(2)=28 :: R=R+3 :: NEXT
 J :: R=13 :: IF Y(1)>Y(2)TH
EN T=Y(1):: Y(1)=Y(2):: Y(2)
=T :: T=M(1):: M(1)=M(2):: M
(2)=T :: T=D(1):: D(1)=D(2):
: D(2)=T
180 IF Y(1)=Y(2)AND M(1)>M(2
)THEN T=M(1):: M(1)=M(2):: M
(2)=T :: T=D(1):: D(1)=D(2):
: D(2)=T
190 L(2)=28 :: IF Y(2)>Y(1)T
HEN 220
200 IF M(1)=M(2)THEN B=ABS(D
(2)-D(1)):: GOTO 260
210 CALL LEAP(Y(1),X):: FOR
J=M(1)+1 TO M(2)-1 :: B=B+L(
J)+X*(M(1)=2):: NEXT J :: B=
B+L(M(1))+X*(M(1)=2)-D(1)+D(
2):: GOTO 260
220 CALL LEAP(Y(1),X):: B=L(
M(1))-D(1)+X*(M(1)=2)
230 FOR J=M(1)+1 TO 12 :: B=
B+L(J)+X*(J=2):: NEXT J
240 FOR J=Y(1)+1 TO Y(2)-1 :
: CALL LEAP(J,X):: B=B+365-X
 :: NEXT J
250 FOR J=1 TO M(2)-1 :: CAL
L LEAP(Y(2),X):: B=B+L(J)+X*
(J=2):: NEXT J :: B=B+D(2)
260 DISPLAY AT(20,1):B;"days
 between" :: B=0 :: GOTO 130
270 SUB LEAP(Y,X):: X=(Y/400
=INT(Y/400)):: IF X=-1 THEN
```

```
SUBEXIT ELSE X=(Y/4=INT(Y/4)
):: IF X=0 THEN SUBEXIT ELSE
 X=(Y/100<>INT(Y/100))
280 SUBEND
```

A leap year is a year that is evenly divisible by 4 unless it is evenly divisible by 100 but not evenly divisible by 400. The subprogram in lines 270-280 will give X a value of -1 if Y is a leap year.
================

Gene Hitz of Arcade Action Software reports another undocumented feature of TI Extended Basic. The manual says that you can only enter a subprogram by a CALL and only leave it by a SUBEXIT or SUBEND, but the manual is wrong. You can GOSUB to a subroutine within a subprogram, providing it does not contain a SUBEXIT, and return; and you can GOSUB from within a subprogram to a subroutine in the main program, and return. In this way, you can transfer variables in and out of a subprogram without putting them in a parameter list. See for yourself -

```
100 CALL CLEAR
110 INPUT M$ :: CALL SUB(M$)
:: PRINT M$ :: GOSUB 140 ::
PRINT "M$ IS";X;"CHARACTERS
LONG" :: GOTO 110
120 M$="SEE WHAT I TOLD YOU?
" :: RETURN
130 SUB SUB(M$):: GOSUB 120
:: GOSUB 140 :: SUBEXIT
140 X=LEN(M$):: RETURN
150 SUBEND
```

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

Another user asked me if there was anyway to key in the ASCII above 127 into TI-Writer's Editor. Many of those ASCII can be entered from the keyboard by using the CTRL and FCTN keys - try this -

```
100 INPUT N$ :: PRINT ASC(N$
):: GOTO 100
```

- but the Editor has been programmed to refuse them because so many of those FCTN and CTRL combinations are used as edit commands.

I had a bright idea - I thought. I wrote a little program to create 127 files, named 128 through 255, each containing just the ASCII of the same number. Now, I thought, when I want to put in such an ASCII I will just LF that file into the next line and CTR 2 to pop it into place. But the Editor refused to even load a file that began with an ASCII above 127!

I'll fool you, I thought. I created those files again, but with an asterisk before the high ASCII. Now they loaded alright - but each ASCII above 127 became an ASCII 128 numbers lower! It is too bad that the Editor does not have a command to add 127 to an ASCII, just as CTRL U subtracts 64, but if you want those graphics characters in your text you will just have to transliterate them and print through the Formatter.

Folks take it for granted that my Nuts & Bolts disks are only useful for programmers, but they contain many routines so simple to use that anyone can use them to dress up their favorite program. For instance -

```
20083 SUB TITLE(S,T$):: CALL
 SCREEN(S):: L=LEN(T$):: CAL
L MAGNIFY(2)
20084 FOR J=1 TO L :: CALL S
PRITE(#J,ASC(SEG$(T$,J,1)),J
+1-(J+1=S)+(J+1=S+13)+(J>14)
*13,J*(170/L),10+J*(200/L)):
: NEXT J
20085 SUBEND
```

Key that in and save it by SAVE DSK1.TITLE,MERGE . Load your favorite program. Enter MERGE DSK1.TITLE . Make sure your program does not have a line 1 or 2 - if so, RES it. Type in -

```
1 CALL CLEAR :: CALL TITLE(5
,"MY PROGRAM")
2 FOR D=1 TO 1000 :: NEXT D
```

```
:: CALL DELSPRITE(ALL)
```

And try it. Instead of "MY PROGRAM", put the name of your program. Instead of 5, put the number of whatever screen color you would like, from 2 to 16 - check your Basic manual. Change 1000 to whatever delay you want - if you have selected a screen color that will leave text legible, use -

```
2 DISPLAY AT(24,1):"PRESS AN
Y KEY" :: DISPLAY AT(24,1):"
press any key" :: CALL KEY(0
,K,S):: IF S=0 THEN 2 ELSE C
ALL DELSPRITE(ALL)
```

You might also need a CALL SCREEN(8) to restore normal screen color.

Oops! Memory full! - Jim P

## TI WRITER FOR NOVICES

In the last issue of TI*MES we carried a summary of TI Writer commands. This article carries on from there. There will be a short break and then from Issue 41 we shall carry on with more info on TI Writer - and of course its clones such as Funlweb.

This article is for you if you have TI WRITER and no manual...

The menu choice varies depending upon which version you have but is usually 1. EDITOR and 2,FORMATTER. Other options should be covered in text files on the disks. (These are DV80 or DIS/VAR 80 etc).

First, EDITOR. This creates a full screen editor, on which you create your text. The screen "paper" is 80 columns wide, and is shown to you 40 columns at a time.

You do not have a full single character horizontal scroll - the screen is split into three columns of 40 characters: the leftmost screen display is the first 40 columns (columns 1 to 34 of your text if you have the line numbers displayed). Then if you move the cursor to the right, you will trigger a switch to display columns 21 to 60, and finally 41 to 80.

When you select EDITOR you will note the cursor appears at the top of the screen, on what is called the COMMAND LINE. The use of this line is described in the section below on TEXT EDITOR COMMANDS.

First, lets create some words!

See at the top of the screen some words, with some letters in CAPITALS? For instance Edit... the capital E means that if you ENTER an E on this line you go into EDIT mode... so ENTER a letter E.

Did you hold shift down or have ALPHA LOCK on? No need to when in this area: even with ALPHA LOCK off, capital letters will be entered. Entering E causes the COMMAND line
to leave the screen and you are presented with the start of your paper, on which you can type your letter.

To return to COMMAND line, you press the keys FCTN and 9 ("BACK"). First though, lets look at all the instructions you can give to the computer while staying in the Edit mode....

SHIFT and ALPHA LOCK have their usual uses! And you have an auto-repeat on the keys. If you need to auto repeat a character using the SHIFT key, you can release the SHIFT key when auto repeat has started and just hold the main key down - SHIFT will be assumed to continue until you release the key.

ENTER will place an odd character on the screen, which looks like a small C over a small R - this is the Carriage Return symbol, and is NOT printed.

It is important when REFORMATTING - more later!

When you come to the end of a line and keep typing, WORD WRAP will move you to a new line automatically, and also ensure that you do not have a word cut in half in the process.
   Unfortunately, word wrap takes a finite time, and many even moderate typists will find that it pays to check the first word at the start of the wrapped line for missing letters - our console lacks a keyboard buffer, and any keys pressed while word wrap is in progress are ignored.
   To end a paragraph, press ENTER and you will move to a new line, and a CR will be inserted at the end of the previous text.
Before we move on... TI WRITER is key-compatible with Wordstar, should you use that program on another computer! However, in this article I shall not deal with the Wordstar keys, but rather with the more convenient use of the TI Function keys. Funlweb has some extra function keys also.
As space is limited, each Edit mode command can only be described briefly here, but the following should help you make progress:
 The Arrow keys: FCTN E S D and X move the cursor one space in the appropriate direction.
CTRL L moves the cursor to the top Left of the screen, but keeps the screen display the same.
CTRL 6 moves the cursor to the first word in the paragraph it is in the middle of, AND moves it to the top left of the screen - therefore moving the text on screen, usually upwards!

  PARAGRAPHS are collections of words between CR symbols. That is, each CR marks the end of a paragraph.

CTRL 8 is New Paragraph- it has the same effect as ENTER, it adds a CR to the end of the current line and moves the cursor to the next line.
CTRL V moves the cursor to the start of its current line.
CTRL 9 is New Page- it inserts not only a CR but also a PA, which is also not printed- the PA symbol will cause your printer to move to a New Page.
CTRL 4 is a tricky one- NEXT PARAGRAPH. When you type CTRL and 4, the text moves up off the screen and the cursor moves to top left. However, the line of text that your cursor was on does NOT have a CR added to it!
FCTN 5 is Next Window and enables you to quickly flick through the three columns of page. It is cyclic - from far right you go back to far left.
FCTN 4 is Roll Down - the cursor moves down 24 lines ( having the appearance of moving the text up 24 lines- the cursor keeps its position on screen!). If there are not 24 lines below the cursor, it moves to the end. FCTN 6 is Roll Up and moves the cursor up 24 lines.
FCTN 7 is TAB (more later) and moves the cursor to the next tab setting on the right, while
CTRL T moves the cursor to the next tab position to the LEFT.
CTRL 7 is interesting - it is the Word Tab. If there is no text after the cursor, the cursor will move one space right, otherwise it will move to the start of the next word.

  All those commands move the cursor around - and for speed, remember that you have an auto-repeat function on the keys!
Other keys you may use in Edit mode are:
FCTN 9 (or FCTN +) to go back to the COMMAND LINE.
CTRL 1 is your OOPS key... in the commands below, if you press the keys in error you can recover by immediately pressing CTRL 1. NOTE that word IMMEDIATELY - I dont mean quickly! but rather that pressing any key between the commands listed below and Oops, will stop Oops working!

FCTN 1 - deletes character cursor is sat on
CTRL K - deletes all text to the right of the cursor
FCTN 3 - Not only deletes text but deletes the actual line!
CTRL 5 - really useful this one, it duplicates the line above! - HOWEVER it will also delete the line the cursor is on, so dont use it if the cursor is sitting on text you wish to keep!
Thats the end of the commands Oops can reverse. Now for some more...

FCTN 0 is a toggle which enables you to display or not display the line numbers on the left side of the page - they are not printed anyway.
FCTN 2 is INSERT CHARACTER. Under normal circumstances, it opens up a line for text to be typed in. When done, remembering to end with a space! if one is needed- press CTRL 2, which is REFORMAT.
FCTN 8 is INSERT LINE, and works by moving the line the cursor is on DOWN, leaving the cursor on a blank line.

CTRL 3 changes the screen colour combinations - not very many choices but better than none! _

With WORD WRAP off (toggled with CONTROL 0), we are in FIXED MODE and the following key commands alter:

INSERT CHAR (Fctn 2) will merely push the text to the right as you enter the inserted material - very like using INS when entering a Basic program. And when the text is pushed to the right hand side of the screen, it starts getting deleted, so careful!
REFORMAT (Ctrl 2) is used to terminate insert mode, also terminated by use of the other cursor movement keys.
New Paragraph, Last Paragraph, and
Next Paragraph do not function in fixed mode.

Those are the directly active keys.

You can also insert commands to your printer into the text, using CONTROL MODE.
CONTROL MODE makes available from the keyboard, ASCII characters 0 to 31 , so that you can send those codes to the printer : they are NOT printed, unless that is a part of your printer instruction set : see your printer manual for details.
You enter control mode by pressing CTRL U, which causes the cursor to become an UNDERLINE ( Notice the cursor shape always tells you which mode you are in: Word Wrap, Fixed, or Control).

With the UNDERLINE cursor, you have access to the lower ASCII codes by pressing the following key combinations: ASCII 1 to ASCII 26 are simple SHIFTED A to Z - thats easy to remember!
ASCII 0 (zero) is a SHIFTED ZERO - thats easy to remember!
Then you'll need to write these down:
ASCII 27 is FCTN R
ASCII 28 is FCTN Z
ASCII 29 is FCTN T
ASCII 30 is SHIFT 6
ASCII 31 is FCTN U
As you enter these low codes, odd characters will appear on the screen - they will not be printed! - you will get used to their appearance in time. They are based on the HEXADECIMAL equivalent of the codes. Remember to switch OUT of control code to use ordinary keys- toggle with CTRL U.

Your printer may for instance require a character 15 to switch to condensed print mode. To insert a character 15 in your text, you need to key: CTRL U then SHIFT O then CTRL U again.
ESC is short for ESCAPE and is the
ASCII value 27, or FCTN R
Consult your printer manual for details of the codes your printer needs.
Note that TI Writer and your printer may have similar codes: it is easy to be confused with the TAB settings on TI Writer and those of your printer: but they are different things! It is usually easier to use TI Writer TABs but for some difficult jobs it may be better to ignore TIW TABS and set and use TABS on your printer - see your printer manual!
One example of compatible but different commands is the UNDERLINE: the keyboard has an underline as FCTN U - but my printer has an underline function availaboo by using ESC - 1 and ESC - 0. If I use both, the printer prints a continuous underline, with a broken underline one pixel above it!

PAGE START: When you switch your printer on, wherever the platten is - and the paper held by it - is marked in printer memory as the start of the page. The printer then keeps count of the number of lines printed. If your printer has a default page length of 66 lines, and after 40 lines to send a PA symbol, or the standard page feed chgracter, ASCII 12, then the printer will move the paper up 66-40=26 lines.
If you manually move the paper up or down, the printer does NOT count that movement! So if you are using either Form Feed command, take care to avoid all manual paper adjustments!
To print out your text, go back to
the command line, enter PF (Print File) and then the printer name (eg PIO) and off it goes. To save text to disk, enter SF (Save File) and then DSK1.FILENAME or whatever. To load text you use LF (LoadFile) then DSK1.FILE.
There is much more to TI Writer- we have barely covered the Editor portion (the manual is VERY thick!) so until Issue 41...

I use an expensive IBM word processor at work- and there are some things which are fast and simple on TI Writer that the IBM program either finds VERY hard, or sometimes cannot do at all. Of course it can do a couple of things TI Writer cannot do, but TI Writer is not to be sneezed at- I wrote a book with it, and all these items in TI*MES have been produced with it!

2022 note: IBM program was Displaywrite 4- which I still have... stephen

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

PIXEL GRAPHICS USING XB ONLY

TI*MES #38 p31
TI*MES #27 p27
I had an enquiry from a user with Console and XB only, having problems applying the XB pixel plotting routine from issue 27. As others may have similar difficulty, here is my response...

When you replace:
```
CALL LINK("PIXEL",XO+SC*X,94+SC*Y)
```
with:
```
CALL PLOT(R,C,S)
```
there is a very important step which I think you have omitted:
you must set the variables R and C to the row and column numbers you wish to turn the pixel on at.
eg:
```
R=XO+SC*X
C=94+SC*Y
```
--->more

There are two further points to consider:
SCALE: The Missing Link allows you to plot to a screen composed of 190*255 pixels. Using the Extended Basic plotter you are limited to redefining the characters available in XB (32 to 143= 111 chars). These can be anywhere on the screen of course.
In a very dense plot you would be limited to an area only 11 x 10 characters of 8x8 pixels, that is R from 1 to 80 and C from 1 to 88, so the SCale factor needs to be reduced by about a third. (SC=SC/3).
If- as often happens- the plot is not completely dense, with character places in the middle of the plot with no used pixels, you can often get away with using a slightly larger area of the screen.
A good starting point would be to try amending the scaling factor, immediately after it has been set in the printed program: SC=SC/2.5

When the XB routine runs out of characters to redefine it just goes into a closed loop.

SCREEN SIZE:
The Missing Link is also entirely happy plotting OFF screen, something the XB routine cannot handle, so you will need to add delimiters after R and C have been set:

```
IF R<1 THEN R=1 :: IF R>190 THEN R=[illegible] 190
IF C<1 THEN C=1 :: IF C>250 THEN C=250
```

The variable S is set at the start (as shown in #27) to the starting character code and is then used to keep track of which character is being redefined.
======================================

## ASSEMBLY CODE

The Art of Assembly - Part 2
Starting at the Bottom
By Bruce Harrison
Copyright 1991, Harrison Software

In Part one, we discussed the two approaches to program structure, Top Down and Bottom Up. In this article we'll provide some "primitive" source code sections to provide services. Please note that, in Assembly, there are about as many ways to do any given thing as there are programmers trying to do it. We'll try to provide the rationale for the way we approach things as we go along. In general, our approach is to minimize memory consumption and maximize speed of execution. Those two don't always go together, but in many cases the most memory-efficient code also executes fastest.

Bear in mind that, for the time being, we're working in the environment of an Option 3 (Load and Run) E/A program. Let's start with the matter of providing Workspace Registers. Many programs contain a source statement like:

```
    WS    BSS   32
```

That's fine, but doing this uses 32 bytes of the avilable program memory for your registers. There is an area in low memory designated for User Workspace, at address >20BA. To use that, you can make an equate in the beginning of your source code like this:

```
    WS   EQU  >20BA
```

Now at your program's start point, you can simply LWPI  WS, and your registers will be at >20BA, not taking up 32 bytes of program space. (Please note this should not be done when linking from Extended Basic, unless your program never returns to XB until it's finished.)

Let's quickly move on to another subject, that of a subroutine to clear the screen for you. We've used many different techniques for this, so let's explore a couple of alternatives.

One can do it like this:

```
    CLS   CLR  R0       Point R0 at screen origin
          LI   R2,SCRWID*24  Load R2 with total
          LI   R1,>2000  make left byte of R1 the space
    LOOP  BLWP @VSBW    Write one space
          INC  R0       Increment screen location
          DEC  R2       Decrement counter
          JNE  LOOP     If not zero, repeat operation
          RT            Return to calling program
```

Here you'll see one of our little tricks. Sometimes when starting a program, we don't know for sure whether we want to operate in Graphics mode or in Text mode. Thus in many places in the program we'll use the mnemonic SCRWID, then at the beginning of the program we'll put a value in for SCRWID through an equate like SCRWID EQU 32 or SCRWID EQU 40. This was really a two-barreled trick, because it also lets the assembler do some math for us. In this case, the assembler will multiply 24, the number of rows on the screen, by the number of characters per row (SCRWID) and thus will load R2 with the correct number of spaces to fill the screen. The above method will work, but won't be as fast as a method using VMBW to write whole screen lines to the screen. We can gain some speed by setting aside a block of 32 or 40 characters' space, writing a space into each of those, then writing 24 such lines to the screen. There would need to be a block of bytes reserved, like this:

```
    SCRLI  BSS  SCRWID
```

There's our friend SCRWID again, this time telling the assembler how many bytes to reserve for a screen line full of characters. Now the code to clear the screen gets more complicated and takes more memory, but executes faster:

```
    CLS   LI   R2,SCRWID  Sets R2 to characters in screen line
          LI   R5,>2000   Sets left byte R5 to space
          LI   R3,SCRLI   Point R3 at SCRLI
          MOV  R3,R1      Point R1 at SCRLI also
    LOOP1 MOVB R5,*R3+    Move one byte and increment R3
          DEC  R2         Decrement R2
          JNE  LOOP1      If not zero, repeat
          CLR  R0         Point R0 to screen origin
          LI   R2,SCRWID  Set R2 again
          LI   R4,24      24 rows to clear
    LOOP2 BLWP @VMBW      Write SCRWID bytes to screen
          A    R2,R0      add that many bytes to R0
          DEC  R4         Decrement R4
          JNE  LOOP2      If not zero, repeat
          RT              Return to calling program
```

That block of memory which we set aside as SCRLI can be used for other purposes, as you'll see when we get to some other subroutines. We can, for example, use it to stash strings.

Before we go further with subroutines, we ought to discuss how to properly "nest" them in Assembly. If you're used to programming in Basic or XB, you know that subroutines may include GOSUBs to other subroutines, and that so long as each ends with RETURN, all will be well.

In Assembly, the calling of a subroutine by BL @SUBNAM will work properly only if the subroutine does not call others. To get around this problem, we establish a "stack" to keep track of our subroutine return addresses. To do this, set up a data area somewhere (perhaps at the very end of your program) which will contain the return addresses for nested subroutines.

A simple entry such as:
```
   SUBSTK BSS  24
```
This 24 bytes will suffice to hold 12 levels of nesting. The other requirement is to have a pointer to keep track of position in that stack. We simply dedicate R15 to that purpose. Somewhere in the beginning of the program, we insert LI   R15,SUBSTK, so that before we call any subroutines, R15 points to the beginning of that stack.

Now in any subroutine that calls others before it returns, which we define as a High level subroutine, we place this instruction at the beginning of the subroutine:
```
          MOV R11,*R15+
```
That puts the R11 return address in the location pointed to by R15, and makes R15 point to the next word in the stack area. At the end of one High level subroutine, we place the following code:

```
    SUBRET DECT R15  Point back to previous stack word
           MOV  *R15,R11  Move that word to R11
           RT             Return.
```

Other high level subroutines can return by a simple B   @SUBRET. Note that simple subroutines that do not in turn call others, which we'll call Low level subroutines, need only the RT at their ends to return properly. The stack area can be placed anywhere. We recommended putting it at the very end of a program so it's open-ended, as long as the program doesn't fill all of the computer's memory. Placing it elsewhere is okay so long as you're sure about how many levels of nesting are required. If you underestimate, something important could get overwritten by your stacking.

Let's say that you are writing a High level subroutine which needs to have the screen cleared before it can proceed. The subroutine would look something like this:

```
    BIGSUB MOV  R11,*R15+    Stash R11 on SUBSTK
           BL   @CLS         Clear the screen
           (rest of subroutine)
           B    @SUBRET      Go to the high level return
```

This assumes there is already another High level subroutine which ends with the code shown above at label SUBRET. By this method, subroutines may be stacked to any number of levels without losing track of the return address of any subroutine.

Now we'll move on to a few more handy subroutines, and introduce the idea of multiple entry points. Let's say you'll need an ability to move strings around in memory, and you'll also need the ability to move groups of bytes that are not organized into strings. (For our purposes, a string is merely a group of bytes where the first byte is the length, and the rest of that many bytes is the string. For example, we might have a string initialized in our data section like this:

```
    CPYWRT BYTE 14    Length of text
           TEXT 'Copyright 1991'
```

The first byte is 14, which is the length of the string that follows. Now let's suppose that we want to move that string to another location which we'll call TEMSTR for Temporary String. Assume that at least fifteen bytes of memory have been reserved at that place. We'll be using R9 to point to the origin of the string and R10 to point to the destination address. We can preload registers with the addresses to move from and to, like this:

```
           LI   R9,CPYWRT  Put address of CPYWRT in R9
           LI   R10,TEMSTR Put address of TEMSTR in R10
```

Now that pointers have been set, we can proceed with a BL @MOVSTR, where the subroutine looks like this:

```
MOVSTR MOVB *R9+,R4     Get length byte in R4
       MOVB R4,*R10+    Place that byte at R10 location
       SRL  R4,8        Right-justify length in R4
MOVBTS MOVB *R9+,*R10+  Move one byte, inc pointers
       DEC  R4          Decrement length count
       JNE  MOVBTS      If not zero, repeat
       RT               Else return
```

This subroutine uses R4 as a counter for the loop at MOVBTS. We here at Harrison conventionally use R4 and R5 for loop counters or other temporary numbers. But just for a moment let's assume you have a need to move a group of bytes from one place to another but they're not organized as a string, in that there's no length byte at the beginning. Let's say you have 75 bytes to move from location XYZ to location ZXY. Here you can use the label MOVBTS as a second entry point to the subroutine. You'd do it like this:

```
LI   R9,XYZ      Place source address in R9
LI   R10,ZXY     Place destination in R10
LI   R4,75       Number of bytes in R4
BL   @MOVBTS     Call subroutine MOVBTS
```

This technique has been used many times in our programs, and we've found it very useful, in that it's more efficient in use of memory than having two separate subroutines with such similar functions.

Next, let's look at a very small subroutine which has an important lesson to teach us. Assume that you've got many places in the program that require a single-keystroke entry, such as the answer to a Y/N question. To prepare for such a subroutine, we'll put the equates STATUS EQU >837C, KEYADR EQU >8374 and KEYVAL EQU >8375. Then near the start of our program we'll insure that our key-unit is zero by writing this one line of source code CLR @KEYADR. We'll also need somewhere a byte initialized to the value >20, such as ANYKEY BYTE >20. We can then use the short subroutine like this:

```
KEYLOO CLR  @STATUS      Clear the GPL Status byte
       BLWP @KSCAN       Use utility to scan keyboard
       CB   @ANYKEY,@STATUS  Has a key been struck?
       JNE  KEYLOO       If not, try again
       MOV  @KEYADR,R8   Else put key struck in R8
       RT                Then Return
```

You'll notice that there's an extra instruction in there which moves the word at >8374 intd R8. The left byte of that word will be zero, and the right byte will be the value of the key struck. Thus the register's value will equal the ASCII code for the keystroke. We do this on purpose because, in most cases after we return from this subroutine, we have to do a series of comparisons to the key struck. Having the key's value already in a register makes that process easier, and moving the key value into a register before exiting the subroutine uses less memory than doing it after return. Suppose we had asked a Yes/No question, and want the default answer to be No. Upon return from the above subroutine, we could have:

```
       CI   R8,89        Is answer upper case Y?
       JEQ  YES          If so, Jump
       CI   R8,121       Is answer lower case y?
       JNE  NO           If not, answer is No
YES    (perform action for Yes)
NO     (perform action for No)
```

The activity at label YES may be a simple branching to some other part of the program, and label NO may be a simple continuation of some process, but that's not important to our point. By moving KEYADR into R8 in the subroutine, we'll save many bytes of memory if this kind of comparison needs to be done each time we've used the subroutine. The point is that the content of a subroutine should be considered very carefully. A small added function like we've shown in the above example can add up to significant savings of bytes by incorporating it into the subroutine instead of having to repeatedly perform the operation outside the subroutine.

In this article, we've just scratched the surface of the subject of subroutines. In the next article, we'll go back to the subject of structure for a bit, and discuss some of the minimum required things to get a program started and ended gracefully. In later articles of this series we'll move into more advanced subroutines, some of which will depend on things we presented here.

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

# FOR SALE

32k stand alone ram expansion, complete with leads. 15.00 ono

Speech synthesizer. 15.00

Extended Basic
new in box
25.00

***special !***

TRITON EXTENDED BASIC
( Has Draw'N'Plot built in )
50.00

## TO CLEAR

All modules listed below are priced at **3.00** each including postage. If I loose out on this, too bad.

TI INVADERS
HOUSEHOLD BUDGET MANAGEMENT
MUNCHMAN
PERSONAL RECORD KEEPING
SPEECH EDITOR
SHAMUS
PICNIC PARANOIA
THE ATTACK
AMAZING
MIND CHALLENGERS
HANGMAN
MUSIC MAKER
FATHOM
NUMBER MAGIC
MICROSURGEON
BEGINNING GRAMMAR
VIDEO CHESS
PARSEC
VIDEO GAMES 1
TERMINAL EMULATOR 2
DEFENDER
PROTECTOR 2
RETURN TO PIRATES ISLE
TOUCH TYPING TUTOR
FIVE A SIDE SOCCER
EARLY LEARNING FUN
OTHELLO
BURGERTIME
MS PACMAN
TOMBSTONE CITY

AT 5.00 EACH ARE THE FOLLOWING.
ADVENTURE modules with adventure programs on tape.
TUNNELS OF DOOM module with data files on tape. **8.00**

**If you are interested in more than one module, please ring for special prices!!!**

CONTACT...
**Mike Curtis**
**21 Treliske Road,**
**Roseland Gardens,**
**Redruth,**
**Cornwall**
**TR15 1QE**

M.G.C.S.

MIKE GODDARD COMPUTER SUPPORT

NEW AND USED COMPUTER EQUIPMENT BOUGHT AND SOLD

-------------------

TI-99/4A SPECIALIST

-------------------

SPARES

KITS

REPAIRS

COMPONENTS

BOOKS

PRINTER CABLES

RESISTORS

BATTERIES

DISK DRIVES

POWER SUPPLIES

POWER CABLES

PRINTERS

MONITORS

CAPACITORS

DISK DRIVE CABLES

RIBBON CABLE

FOR LATEST LIST CONTACT:
M.G.C.S.'SARNIA', CEMETERY ROAD, RHOS, WREXHAM
CLWYD, LL14 2BY
Tel:(Ø978)843547. MBX:Ø22212529