TI*MES

Autumn 1995

Contents

All contributions for issue 51 *must* be submitted by December 1st 1995

Chairman Trevor Stevens	249 Southwell Road East, Rainworth, Notts, NG21 OBN	01623 793077
Vice Chairman Mark Wills	20 Cockayne Road, Hartescott Grange, Shrewsbury	01743 350588
Gen. Secretary Richard Twyning	24 Peel Road, Mansfield, Nott, NG19 6HB	01623 27670
Treasurer Alan Rutherford	13 The Circuit, Wilmslow, Chesire, SK9 6DA	01625 524642
Disk Librarian Stephen Shaw	10 Alstone Road, Stockport, Cheshire, SK4 5AH	
Module Librarian Francesco Lama	14 Granville Court, Cheney Ln, Oxford OX3 0HJ	01865 721582
Hardware Gary Smith	55 Boundary Road, Newark, Notts, NG24 4AJ	01636 706787
TI*MES Editor Richard Speed	8 Corfe Close, Horsham, West Sussex, RH13 7XL (Email: rspeed@CIX.COMPULINK.CO.UK)	01403 730836

Welcome to issue 50 of TI*MES - our first half century! As you can probably already see, I've been playing with the magazine format again... this time in an effort to improve readability. After discussing the problem of the faint quality of print within the magazine with our printers I was given two options. To explain these options, let me just explain how the magazine is normally put together:

Putting TI*MES Together

First I receive the articles from you, the members. These articles are then cut down to fit on an A4 page (with about a half an inch gap on the left and right and an inch on the top and bottom to allow me to fit in the page headers). The A4 page is preprinted with the header and footer containing page number, and so on before I actually stick the article on. All the pages are prepared like this and once complete it is then sent off to the printers. The printers reduce the size to A5 and then use the reduced prints to produce the magazine as you're reading it now. As you can see, there are two copying processes - one to reduce the magazine down and one to do the print run, so a faint article is going to get fainter...

Darkening the print

So what are the solutions? The printers can darken the pages as they reduce them, but they charge for that, or all contributors can buy lasers(!). However, while discussing this problem with the printers we came up with a third option. If they are sent the magazine as one big PostScript file on a PC or Mac disk, they can then run it out on their 1200dpi laser printer. Thus all the magazine would come out in a clear dark print including (hopefully) illustrations.

The uses of PCs

So how is this done? Well, I have an IBM compatible PC at home which I use for work. Since most contributors prefer to submit on disk I can transfer the articles from the TI to the PC where I can use a DTP package. Once I've finished the issue I can then print it out to a postscript disk file and send that to the printers. As you'll see in this issue, all the articles are printed in Times Roman with all listings printed in Courier to help readability. I'd welcome all comments and suggestions though...

Diskless?

So where does this leave those members who don't have disk systems? Well, with the help of Walter Allum, we are producing a text entry program that can be used to save text to cassette which I can then load here and use, or I am more than happy to type up anything I'm sent.

Disk Library Listing

You'll also find a hard copy print out of the disk library with this issue of TI*MES. I hope it'll encourage you all to have a look at the disk library. Contact Stephen for the latest information, but please be patient if you don't get a reply immediately (see Rambles for more info.)

The MOBB BBS

Lastly, if you're able, try the BBS - you'll find lots of files there including the disk library list. You can also Email me (and the committee members) with your comments or suggestions.

Strewth, I've waffled on much too long! Enjoy the rest of the mag!

Richard Speed

Rambles

Hello again. Quite a different subject to begin with, and multiple apologies to anyone who does not have access to a PC and is unlikely to in the next couple of years.

Emulating the TI

The topic is EMULATORS- clever little (?) programs that make one computer think it is something else entirely. For the PC you can buy a CD Rom with emulators- and THOUSANDS of programs for them- to keep your Spectrum or Commodore 64 programs.

For the TI enthusiast, you can go to your CD Rom supplier and ask for a copy of Night Owl 13 (August 1994) and there you will find a ready to run TI Emulator with Extended Basic, TI Logo and Hunt the Wumpus. This is an early version of the TI Emulator written by Edward Swartz, and right now it is totally unsupported and registrations are not accepted.

Moving on a little, I also have a PC disk with Version 5.01 of Edwards emulator, again a ready to run program and with a little more TI stuff working. The principal things you cannot do with it...

- **`Call Coinc(ALL)`** does not function
- Sound noise channel (4th channel) does not function and may generate odd sounds

Many modules avoided TI's standards, notably Atarisoft and regrettably Scott Foresman, and a number of machine code programs used odd keyboard scans or disk access. For this reason not everything will function. The PC mouse and joystick are VERY different devices to the TI joystick and while the emulator will respond to them, you may prefer like me to stick to the keyboard - and for programs that insist on joystick control, the emulator allows YOU to insist on keyboard operation!

Emulator Requirements

The emulator requires a 386 processor, preferably 33MHz or better. A few programs may run very fast but can be slowed down easily. I have put together FOUR PC HD 3.5" disks with archived TI material, ready to run programs and text files. These expand to a little over 8 megabytes for your PC hard disk. They are tied together with a DOS menu system called ADMAST which makes selection and operation that much easier (The emulator only allows a maximum of 32 modules in a selection list so we use dos to switch between lists and to set which dos directories to use for data and so on).

I have also added the archive program for anyone who does not have it - it is needed to unarchive the files! (I have used ARJ). I have also put in a batch file to simplify installation to C:\TI but that is untested! Please note that THIS version includes code which is copyright by TI and can only be supplied to TI owners who have the original chips! Also note that Edward has requested that this version is not distributed, is not accepting registrations, and is offering no support.

All together I have over a thousand files, with lots of modules, and many TI Basic, Extended Basic and TI Logo programs. Some of the more interesting contents include TI Base, TI Artist (Vn2), Multiplan, Triton Super Extended Basic, Mancala!, Adventure and Tunnels of Doom (with databases), PRK,PRG, and lots of others. I have not tried writing 9900 machine code with EdAs yet...

Oh yes- TI Writer AND Funlweb 5.01. The emulator saves disk files in TI format (eg with headers for TI programs to read) and you cannot read a file saved from TI Writer with a dos text editor.
Therefore I have added a dos utility from Buttonware (originators of the shareware concept) which acts as a shell to redirect printer output to a disk file (using append mode). This is NOT a TSR but is used to run TI Emulate and anything you print to PIO will be sent to a specified dos text file which any word processor can pick up. Documentation includes the dos-readable text of my 1983 book.

I am in great debt to Mike Poskitt for the transfers from TI to PC format.

The four PC disks are available at "standard" PC shareware library prices- that is all four disks for TEN POUNDS including disks and postage. Please don't send blank disks, PCs can catch things.

Moving on a little....

I also now have on one PC HD disk (3.5") the very latest version 6 of Edwards TI Emulator, and boy have things changed. The "compatibility" claimed is only true if you amend your previous files - and I have over a thousand of em! Also there is absolutely NO TI copyright code this time. So if you want a TI system or modules, you either supply them OR buy them from Edward, who will pay TI a royalty. Also, please note that Edward has clearly stipulated that he will not offer any support at all for this version after December 1995.

The documentation refers to the fourth sound channel being available under certain circumstances but I cannot verify that nor that CALL COINC(ALL now works. It is also suggested that more modules should operate under Vn 6.

Now comes the hard bit - with Vn 5.01, I had it up and running immediately. I cannot make Vn 6 operate at all on my system. Utilities are provided to modify your Vn 5 files but again, I could not get these to work, although they did manage to trash my (backed up!) original 5.01 files.

The set up of Vn 6 appears to me to be quite a bit more complex than Funlweb, so if you had any problems with configuring Funlweb, you are likely to have problems with this one! I will however copy the disk for anyone who wants it for THREE POUNDS including disk and postage. And if anyone can set it up to work WITH module files (Extended Basic and Ed/As as a minimum!), and suitable to run from a floppy on my system, you can have a refund! And if you cannot make it work - neither can I!

Of course there is also PC99, a commercial program also paying royalties to TI. This could cost you (in bare) as much as emulators for a half dozen other machines and thousands of programs for them. But it is the only other available if you want to emulate a TI99/4A on a PC. I have not seen a copy but have heard that the latest version is about as fast as a TI on a fast PC.

I fully anticipate that my TI will go on working longer than my new PC... but when my TI finally ceases to operate I will be able to buy a good 486 PC for about fifty quid and get a full TI system to use on it....

Answers to critics time....

The disk library operated at a loss last year and ended the year in deficit. To obtain disks from the USA I consider it fair that we pay AT LEAST the cost of supply, disk, disk mailer and postage. And new disks need to have much larger take up if the cost of obtaining them is to be met from funds generated by the library.

The library costs are not met by current charges. I am entirely happy to give disks away for free- but SOMEBODY has to pay for them and it is not going to be me!

I must also mention that other Group members appear to be obtaining huge amounts of software which they are NOT sharing with the group library - and complaining of library charges which they need not pay if they share more generously!

At present I have a large number of disks available to obtain (for example the entire collection of Jim Peterson disks) but lack the funds to obtain them and also any indication of interest should they be obtained! It really is down to group members (and officers) to decide what they want. And pay for it.

With under a hundred members it is unlikely that any new disk is going to be "wanted" by more than a handful of members, who therefore somehow have to cover the costs of obtaining an original copy of the software and also the costs of UK distribution. A large quantity of new software is dedicated to quite obscure hardware, (TI's GRAM CARD for example!), or is merely a minor rehash of what has gone before.

PC Shareware libraries typically charge from 2.25 to 2.80 per disk and while we are talking 1.44M disks, we are also talking bigger programs! There is usually an extra pound for postage.

Looking at a current advert I see that ONE dos game (an astonishing 3.6Mb) can be obtained from one shareware library for £5.50. (I got mine free with a PC magazine but that doesn't mean the shareware library supplies it free!).

I am yours to command - but I am not going to continue to subsidise the disk library, and diminishing numbers, usage, and software, means costs RISE. Oh- and those of you with CD Rom drives attached to your TI's, the group would have to pay something over forty pounds to have a CD Rom filled with stuff from the disk library. Plus the postage on the disks going to the person putting the data on the CD. A CD with TI*MES.... of course for a magazine with a circulation of 40,000 the cost is more akin to 50p a CD.

I hate to have to refer to all this again but some members do not seem to be reading the magazine. I now have the care of my blind insulin dependant diabetic father. (Listening out there?). This takes up time. I am in the throes of major changes at work (no I did not keep my job and have to learn a new one). And my son is mildly troubled with one of those annoying developmental syndromes... in a nutshell, I no longer have absolutely oodles of time for disk copying (or article writing, sorry).

If everyone sent in a major order to the disk library, everyone would be kept waiting for a very long time. Can we keep it sensible please???? If you obtain lots of programs at once you will take ages to look through them all. Just get a few at a time and take it easy and the disk library can offer a reasonable service (TARGET is to get your disks back to you within a

fortnight but very large orders wait until they are done!).

Back to IBM graphics (or ANSI graphics as referred to in last issue). Funlweb Vn 5.01 offers full support for those single and double line characters for drawing boxes with! In the EDITOR. No need to use 'orrible transliterate files.

My very ancient Epson FX80 printer does NOT have IBM Graphics - the codes are used for the italics and foreign characters. However, using Funlweb on the PC I can use AND print IBM graphics by using a little dos TSR which converts the characters to Epson Graphics commands.

You may have gathered that I have been spending rather a lot of time putting together my TI Emulator package on my new (?) PC, so that it all works easily and neatly from a simple menu structure. This includes testing every TI module for compatibility, and also adding documentation files. My actual use of my TI has been limited therefore. But it is all in a good cause - continued use of my TI!

Older RAMBLES articles were triggered by outside suggestions and queries - and with almost no other user group newsletters these days, that means it is up to you dear reader to ask me questions (or make suggestions) appropriate to my field of knowledge - Basic programming. Machine code and hardware I leave to all those other clever folks!!

Our editor has chosen to omit the TIPS FROM THE TIGERCUB - do you wish to see them again, we have several more you haven't seen yet! Write in to Richard and ask POLITELY please.

Easy puzzle

To solve this take no short cuts but have your program solve it. Area is length times height. A St George flag has a red cross on a white background. The area of red is the same as the area of white and the flag measures 7 feet by 9 feet. How wide are the arms of the cross?
(Hint- break the cross up into rectangles)

Harder Puzzle

A class at school wanted to know their marks in an exam. The teacher gave then this data, which is sufficient to determine the size of the class and the marks awarded (though not to whom).

1. Maximum mark is 50
2. Each pupil had a different integer total (no fractional marks!)
3. The highest mark times the lowest mark = 1012
4. Six times the average of the highest and lowest marks is the same as the sum of all of the marks.
5. The average of the second best and third best marks is the same as twice the difference between the highest and third best mark.

You can make logical deductions but a program will be required to find the answer. Solutions - programs and answers and timings - to the Editor for publication please!

Not too long - nobody is writing to me about Basic these days! - and in addition to a week at the seaside, there are the problems listed many times already plus remarkably warm weather!

Best wishes to all and HAPPY TIing.

Stephen Shaw

Dear computing enthusiasts,
Why have I opened without saying "Dear TI'ers"??? Well, I thought I would use something that I couldn't possibly have used had we been in an IBM user group!

IBM user groups are mostly profit driven rip-off's, and like most of my fellow students on BSc. Computing Systems, are only out for their own gain. You can tell I'm not very pleased with Nottingham Trent University after the way I feel I've been treated.

If my article turns out to be very short, then I will apologize now, but the course and the university has left me completely demoralized and I'm currently appealing for my result, and I don't know whether to finish my final year project or not!! They constantly contradict themselves. They say you are supposed to show initiative, and form your own opinions, but if you do this you are outcast. But, if you behave like a sheep and swallow all their crap and copy it word for word in an exam, then you're given a 1st!!!!

At the moment I don't really care about a career in computing. I just want to do something I enjoy that will pay the bills so I can do the important things in life, which is of course support the group and write programs for the 4A.

When I say I'm going to write a program such as my CAD system, it's because I want to add to the list of things that can't be done on the 4A.

There was something I read ages ago which I think might have even been in Micropendium, and it said something like "We can't expect to buy a database program as powerful as DBASE for the 4A and pay peanuts for it!"

This was before TI-BASE was released, but it proves that nothing is impossible on the 4A. Please don't use PRESS as an example! PRESS was to be the ultimate Desk Top Publishing program for the 4A and Geneve, but they ran into problems with it. One of its authors was Charles Earl, who wrote Telco, and co-wrote Batch-It.

Therefore, I have two reasons for writing my CAD program. I want to write it to prove that it's not impossible, and I also want to bring software to the 4A to encourage users to stay with it, to prove that you don't need PC's to produce decent output. PC's just over complicate things. I could never contemplate drawing a serious picture with a PC or the AMIGA. It's just not a comfortable thing to do. There is something about our drawing programs such as GRAPHX, TI-Artist, and YAPP that makes them seem much more precise.

There is one problem with them, which I am sure you will agree with. They are much easier to draw with, but their resolutions do look a bit aged when compared to today's machines.

The original reason I wanted to write the CAD program was because I'd used AutoCAD at West Nottinghamshire college (we had no such luxury at university!) and I found it quite stimulating. It wasn't stimulating because of the pictures I drew with it, but because

of the ways it offered of being able to draw pictures.

With a CAD program it is much easier to overlay different seperate objects, and then delete them seperately, bit if you're drawing something with a bit-mapped graphics program, which all of our drawing programs are, then if you draw a line over an object and decide you want to erase it, then you had better hope that the program has got an undo feature. If it hasn't got an undo feature, which TI-Artist and GRAPHX haven't, then you've got to "draw out" the line by hand a pixel at a time!

With a CAD program however, the line is just stored as four variables (for a 2D system), so since the line is an individual object, it can just be instantly deleted, and the object which the line was drawn over can be redrawn again perfectly as it was without the line.

There are some so-called CAD programs for the 4A that just don't live up to the expectations of what a true CAD program should be. After using AutoCAD at college,I knew that if there was such a program on the 4A, then it might just encourage a few extra members to stay with the machine, and the group. Also, I know how the program is likely to turn out, and I know that if it does turn out as I hope, then it will show people what is possible with the machine. Some people are starting to doubt what the machine is capable of, and Nottingham Trent University and others almost had me doubting it too, but there's life left in it yet, and while ever it takes a mere six million bytes to load an IBM PC word processor, there will still be life in the 99/4A. I like to think what I lack in programming skills, I make up in vision. I can visualize my CAD program, and it's almost as though I can see it running in my head. Hopefully, someone who is more skilled in Assembly Language can see what I'm getting at, and convert it into raw 9900.

It was back in 1990 that I used AutoCAD for the first time, and after using it only a couple of times I decided that I just had to have something similar on the 4A, and I started studying how the program worked, such as delays when you select an object to be deleted, because the system is searching its database and trying to find an object that might be under the location of the cursor.

There is no problem with writing the CAD program, I have been thinking about it and planning it for so long that I've worked out most of the problems. The problem with it is though, what language to use. The best thing of course, would be to use Assembly Language, which would be the best for speed, and for maximizing memory, but my Assembly Language experience is just not up to it.

I would need somebody to give me a tutorial on handling floating point numbers, and I would need some decent routines for produding bit-mapped lines and circles, and of course, file handling.

It must be four or five years since I attempted playing with Peripheral Access Blocks after working through Ralph Molesworth's book. My experiment didn't work, which wasn't surprising, since I was trying to play with the DSR Load and Save routines to try and write a file copying routine that could be LINK'ed from Extended BASIC, but the routine locked up and left me a bit disillusioned with continuing. I suppose know that the problem could have been because I had written the routine as an extension to a

slightly out of date version of the XHI source code. I had to do this because I needed to use a DSRLNK that was compatible with XHI, since XHI plays around with VDP quite a bit, so that it can display the high resolution screens from XB. The reason it had to be compatible with XHI was because I was hoping that I could write a file copying routine that I could LINK from Workspace, but it didn't work, so that was the end of that!

The next possibility of writing my CAD program is to use C99, but there are one or two problems. The first problem is that floating point numbers are not properly supported, and someone else wrote a complete set of functions for supporting floating point numbers. It's quite amazing how the routines work, and I have mentioned them in a previous article, and I think I even included a sample program that I wrote as a practice.

Even though the routines are written as an extension to C99, and they're used in a program with an #include directive, the floating point numbers are defined with a FLOAT directive (which is fiddled by using a #define directive). The most important feature of them though, is that I needed the floating point numbers to be able to be passed in and out of functions, and after writing the little program to pass the numbers in and out, I discovered that it worked perfectly.

The only disadvantage with the floating point routines, since they are an extension, is that the numbers can only be operated on by the functions included, so you can't use them in equations. Each calculation has to be done a step at a time because there's a seperate function for adding, and seperate ones for subtracting, multiplying, and dividing.

I will have to break each operation down into loads of individual calculations, which is not impossible, but is very time consuming, since these calculations have to be done at every stage, even when just displaying things to the screen, because everything is scaled to the screen depending upon the currently selected zoom area and size etc. The advantage of a CAD program on the 4A is that the quality of anything produced with it shouldn't be too adversely affected by our resolution, because the program will allow any point of an image to be displayed full screen, including a complete view of the entire image. If there is a very detailed part of the picture that needs to be worked on, then this area can be displayed full screen. Even if the area you want to work on takes up only two pixels if you are viewing the full image, then even this small area can be scaled up to fill the entire screen. This will give an advantage when printing out the completed image, because the increased resolution on the printer will mean that the smallest details will be sharp on the printer because standard 9-pin printers have a maximum resolution of 1920 pixels across the page.

Another disadvantage of using C99 is that the bit map graphics routines do not include anything to handle sprites, and I'll need a sprite to use as a cursor, and as markers for the outline of windows or objects etc.

It would mean writing an extension to the routines to handle a sprite, and then there's also a problem of displaying text of varying sizes for menu's and labels etc.

One advantage though with writing it in C99 is that I can include Assembly Language straight into it and therefore have routines that instantly redraw menus and define character sets.

I could also possibly make use of routines for handling strings etc., or anything else I possibly can to speed up the program. Much of the time, any language would be able to support it. The only slow parts of the program are when it's redrawing an image, because it's got to step through the entire image database and scale every object to the current zoom size.

This gets worse when you want to edit or delete an object, because when you try and select something on screen, the program not only has to scan the entire database, but has to also step through every object. If it's a line, then it's got to check every pixel of the line to see if the cursor was over it when the selection was made, and if it's a circle, it's got to step around the entire circumference of the circle! I've just thought of a solution to the problem while typing this! If I have a visual marker (a sprite) on screen which marks the pixel that the program's currently scanning.

If the user can clearly see that the object being scanned is not the one that needs editing, then he/she can press a key to escape from this object, and the next, and the next, until the object requiring selection is scanned. This will speed things up alot, but there's still the problem of objects that don't appear on the screen because the area currently being displayed is only a small part of the entire image. I've got to write routines that check each different object, such as lines and circles etc., and if an object is entirely off of the screen, then it can be ignored, which will hopefully speed things up again, because for a line I only need to check the end points, and for a circle I only need to check four points around the circumference. See figure 1.

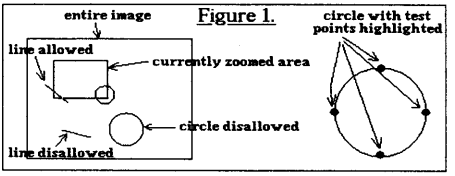


Figure 1.

From figure 1 you will see that the test points start from 0 degrees, and are 90 degrees apart. Circles which have part of their circumference crossing the zoom area must still be allowed.

After talking to Mark about the CAD program, he suggested that I write the first version using The Missing Link. It will be a quick way of testing out my theories, and a way of getting a head start when I finally attempt to write a version in C99. I can think more clearly about a program if I write it in Extended BASIC first, and when I was still at the Hell Hole (Nottingham Trent University!) I even wrote all of my programming assignments in Extended BASIC first, including a lexical analyser that was supposed to recognize individual C language statements and operators. My XB version was about ten lines long, and used a RELATIVE file on disk to control the recognition. The PASCAL version on the IBM PC was about 300 lines long, and I used XB to generate the PASCAL program!

I originally decided not to write the program for The Missing Link, because I thought I wouldn't be able to erase pixels. I thought that The Missing Link only allowed pixels to be re-displayed in a new colour, and not deleted completely, but according to Mark, two colour mode of Missing Link does allow pixels to be erased again completely. After thinking about it though, PEN ERASE should also allow pixels to be entirely deleted in 16 colour mode. I will have to experiment

with it. I think that before when I experimented with Missing Link I might have erased some pixels that had been set to a certain colour, and then overdrawn the area with more pixels without specifying a colour. The pixels would have been erased from the pattern table, but the colour table would not have been erased, so the old colour would appear again.

I'm not making any promises on when I will start work on the program again, but I definitely will, and if anything goes wrong with it, I'll start again from scratch, and I'll only leave the project when the front cover of TI*MES is produced with the program!

O.K., what's next? A little update on proceedings following the AGM. Those who attended will know that Francesco brought Richard Sierakowski's collection of hardware and software, and I was surprised that Francesco was leaving me in charge of dealing with it all, and Little Nellie definitely used more petrol on the return journey to Mansfield by the time she was loaded up with everything.

I sold one or two things to various people at the AGM, which was mainly books and disks, but the rest of the gear is cluttering up our hallway and my bedroom!!!!

This pile of equipment increased on the 25th of August, when Francesco made another welcome visit to Mansfield. This time he brought Peter Brooks' entire collection of equipment, which includes a very interesting find of an EPROM programmer that fits in the console cartridge port. Also in this collection I have also discovered twelve (YES! 12) full height floppy drives. I don't know if these are double sided or single sided, but if anyone is interested, then let me know. I've also got more ROMOX's than you can imagine. I thought I was doing well with my original collection of ROMOX's, but now it's beyond belief! If anyone wants one (or some!) then they're £2 each.

Also, there's a daisy wheel printer which I will have to test, but if anyone is interested, or is also interested in an Expansion Box complete with Memory Expansion, Disk Controller, RS232 card, then give me a call.

Also, don't jump out of your seats when I say that there is a MODEM in the collection, but it's only a 1275 split baud rate MODEM. It doesn't even do 300 baud, but only does 75bps receive and 1200bps transmit and vice versa.

The disadvantage of this is that it cannot be used to access the bulletin board, but you could use it to connect to other members who have MODEMs that support 1275/7512.

My Miracom will, Richard Speed's definitely should do (it's an actual Hayes!), Gary's does, and I'm sure Mark's does.

If you require any software, then it allows you to contact one of us, or anyone else with the capability of the same speed, and you can download any piece of software you might need, or you could use it to instantly upload magazine articles to Mr. Editor! You can either receive 1200 baud, or transmit 1200 baud, so for the use of XMODEM transfer on TELCO, you've virtually got the equivalent of a full 1200 baud MODEM.

So, what have I been up to since the AGM???

Well, I went on my first holiday in six years! It was a camping holiday, and my last holiday in 1989 was a camping trip to

Youlgreave in Derbyshire. This time I wanted to be a bit more adventurous, and my three college friends and I, travelled at a steady warp six for about ten hours (including stops!) and eventually arrived at the Loch Ness Caravan and Camping site in Invermorriston. We travelled up over night on the Friday, and arrived at Fort Augustus at around 8am. We were only there for four days, and we travelled back on the fifth day, but I recorded 6½ hours of video, including the journey and the amazing scenery. I saw Derek Hayward before I went up, and he said how good the scenery would be on the A82 which starts at Glasgow and ends at Inverness.

Anyone who has been up there will know how good it is. It is sometimes difficult to imagine that we are still in the British Isles as you drive round a bend and through the middle of a cloud!!! Then, as you look upwards, you see a mountain above the cloud and towering above you. When I was editing the video and choosing a soundtrack to accompany the mountainous scenery, there is only one which is suitable, and one that seems specially written for the purpose, and that is the JURASSIC PARK soundtrack!!!

I wanted to have a decent video of the event that my friends and I could look back on in a few years time, and this meant doing it professionally, and adding titles to it.

As you know, I've got an AMIGA, and I've got a video titling program for it, so I thought that if a job's worth doing, then it's worth doing well, so I immediately dug out my STAR TREK II: THE WRATH OF KHAN video and grabbed some graph paper! My friends and I are all into STAR TREK, so I thought it would be good to have titles that were done in the style of STAR TREK. So, armed with a sheet of graph paper and STAR TREK 2, I drew out 16*16 pixel boxes for each large character and drew the characters that I could copy from the screen after pausing the video at the main titles of the film! Since there are only a few characters of the alphabet, I had to fill in the extensive gaps myself by drawing the rest of the alphabet how I thought myself that they should look and I'm quite pleased with the effect, and it looks amazing on screen.

O.K., if it looks so amazing, what system did I use? Well, I said that if a job's worth doing, then it's worth doing well, so I also dug out a spare console and a tape recorder and set it up next to my video in the loft. I keep my SONY up there out of sight, and out of mind, and keep my cheap and nasty MATSUI video downstairs in my bedroom.

So, how did I plug the 4A into the video? Easy. You know that the console's video port doesn't give out proper Composite Video, but produces direct YUV which is what only professional video systems use, such as the large video cameras in television studios, and YUV is the way that T.V. pictures are transmitted. The letters Y, U, and V, stand for different parts of the signal, just as RGB does in that system. Gary will correct me if I'm wrong, but since it's ages ago when I wired my plug to allow me to plug into composite video, but I'm sure that it's the Y and Ground pins that work as composite video.

But there must be some catch? Yes there is. Since it's only brightness and the colour component is seperate. Therefore, you can only get monochrome from it, which doesn't really affect the final result of the titles, but you will remember that a couple of issues ago, I re-submitted the instructions on how to modify the

modulator to give true Composite Video that I found when browsing the early issues of TI*MES for my SOMEWHERE IN TI*MES feature.

This also reminds me that I've not kept up with this for quite a few issues, but you never know, it might well return in the Winter issue (if I have time!). I will start my Winter article earlier, and I should get the majority of completed, especially if the experiment continues. What? Read on and find out!

Right. Back to the video titling system. Well, the video titling system on the AMIGA might look very professional on the surface, but I don't think it's very useable, and like all software on other machines, has convoluted menus and functions that are just not laid out clearly at all, so I decided to make life easier in the way of achieving a predictable result, but it was less easy, due to the fact of all the drawing I had to do, and then typing in the character definitions, but it was well worth it, and because I did it with Extended BASIC I even made a moving star field with sprites to complete the effect, and it looks incredible, just as though we were watching the start of an actual STAR TREK film!!! I'll let you judge for yourselves though. You will see that the program includes a small character set as well as a large character set, and it uses a double set of SUB PROGRAMS, which are BIG and MBIG for the large character set, and SMALL and MSMALL, for the small character set.

The BIG subprograms are used as follows:

CALL BIGTEXT
Defines large character set.

CALL BIG(screen_row, "string")
Prints the string at the specified screen row. The string is automatically centred left and right.

CALL MBIG(number_of_rows,"","","","","","","","")
Prints the strings on the screen centred left, right, top, and bottom. Number_of_rows specifies the number of strings that are displayed. 3 means display the first three, and 8 is the maximum.

CALL SMALLTEXT
Defines small character set.

CALL SMALL(screen_row, "string")
Prints the string at the specified screen row. The string is automatically centred left and right.

CALL MSMALL(number_of_rows,"","","","","","","","")
Prints the strings on the screen centred left, right, top, and bottom. Number_of_rows specifies the number of strings that are displayed. 3 means display the first three, and 8 is the maximum.

There is a complication with the characters that you need to use in the subprograms though.

In the **BIG** routines you should just use **UPPER CASE**, and in the **SMALL** routines, lower case will give you normal 8*8 characters, and upper case will give you double height 16*8 characters.
In both sets of routines you cannot use a space character, but must use an at sign "@" when you want to generate a space in the final display.

In the **BIG** routines, you can generate a full stop character by substituting a "[" character in your program.

In the **SMALL** routines, the following substitutions can be made:

? substituted for . ; substituted for ,
= substituted for - < substituted for (
> substituted for) : substituted for :

In the **SMALL** routines, as you will see from the program, you can also use the digits 0 to 9. Here's the program...

```
1 ! Video Titling - By Richar
d Twyning 27-7-95
2 CALL SMALLTEXT :: FOR D=0 T
O 14 :: CALL COLOR(D,16,1)::
NEXT D :: CALL SCREEN(2)
3 CALL CLEAR :: FOR D=1 TO 28
 :: RANDOMIZE :: CALL SPRITE(
#D,143,15,INT(RND*192)+1,1,0,
INT(RND*19)+1):: NEXT D :: CA
LL MAGNIFY(1):: CALL D
4 CALL MSMALL(3,"TWYNING@HOME
@ENTERTAINMENTS","@","present
s?@?@?","","","","","","):: CAL
L D
5 CALL CLEAR :: CALL MSMALL(3
,"a","RICHARD@TWYNING@","film
","","","","","","):: CALL D
6 CALL CLEAR :: CALL BIGTEXT
:: CALL D :: CALL MBIG(1,"LOC
H@NESS","","","","","","","")
:: CALL D
7 CALL CLEAR :: CALL MBIG(4,"
STARRING","IN","ALPHABETICAL"
,"ORDER","","","","","):: CALL
D
8 CALL CLEAR :: CALL MBIG(2,"
DARREN","BLACKBAND","","","",
"","","","):: CALL D
9 CALL CLEAR :: CALL MBIG(2,"
PAUL","BODSWORTH","","","","",
"","","):: CALL D
10 CALL CLEAR :: CALL MBIG(2,
"KEVIN","GOODWIN","","","","",
"","","):: CALL D
11 CALL CLEAR :: CALL MBIG(2,
"RICHARD","TWYNING","","","",
"","","):: CALL D :: CALL CL
EAR :: CALL SMALLTEXT
12 CALL CLEAR :: CALL MSMALL(
5,"and@a@special","guest@appe
arance","by","STEVE@FELTHAM;"
,"monster@hunter?","","","","):
: CALL D
13 CALL CLEAR :: CALL MSMALL(
3,"Visual@and@Audio@effects",
"by","RICHARD@TWYNING","","",
"","","","):: CALL D
14 CALL CLEAR :: CALL MSMALL(
3,"music@arranged","by","RICH
ARD@TWYNING","","","","","","):
: CALL D
15 CALL CLEAR :: CALL MSMALL(
3,"principal@music","composed
@by","JOHN@WILLIAMS","","","",
"","","):: CALL D
16 CALL CLEAR :: CALL MSMALL(
6,"additional@music","by","JA
MES@HORNER","ELLIOT@GOLDENTHA
L","DANNY@ELFMAN","ALAN@SILVE
STRI","","","):: CALL D
17 CALL CLEAR :: CALL MSMALL(
6,"Tri=logic","Optimum@Pictur
e@Control;","Digital@Stereo@S
ound;","and@DYNAMIC@SIGNAL@FI
LTERING","by","SONY@Inc?@JAPA
N?","","","):: CALL D
18 !
19 CALL CLEAR :: CALL MSMALL(
5,"camera@from","@","Matsushi
ta@Corporation;","<PANASONIC>
;","JAPAN?","","","","):: CALL
D
20 CALL CLEAR
21 CALL MSMALL(8,"filming","D
ARREN@BLACKBAND","PAUL@BODSWO
RTH","KEVIN@GOODWIN","RICHARD
@TWYNING","@","additional@fil
ming@by","RICHARD@GOODWIN"):::
 CALL D
22 CALL CLEAR :: CALL MSMALL(
4,"location@consultants","@",
"DARREN@BLACKBAND","RICHARD@T
WYNING","","","","","):: CALL D
23 CALL CLEAR :: CALL MSMALL(
4,"transportation","@","KEVIN
```

```
@GOODWIN","RICHARD@TWYNING","
","","","",""):: CALL D
24 CALL CLEAR :: CALL MSMALL(
4,"executive@producers","@","
PAUL@BODSWORTH","KEVIN@GOODWI
N","","","","",""):: CALL D
25 CALL CLEAR :: CALL MSMALL(
5,"produced;@edited;","and@di
rected","by","@","RICHARD@TWY
NING","","","",""):: CALL D
26 CALL CLEAR :: CALL BIGTEXT
 :: CALL CLEAR :: CALL MBIG(4
,"LATE","IN@THE","TWENTIETH",
"CENTURY[[[[","","","","","")
27 CALL D :: CALL CLEAR :: CA
LL SMALLTEXT :: CALL DELSPRIT
E(ALL):: CALL MSMALL(1,"JUST
AFTER 7pm","","","","","","",
"",""):: CALL D
28 CALL CLEAR :: CALL MSMALL(
1,"EARTHDATE:@28=7=95","","",
"","","","","",""):: CALL D
10000 SUB D
10001 CALL KEY(0,K,S):: IF S=
0 THEN 10001
10002 SUBEND
20000 SUB BIG(ROW,T$):: LC=1+
((32-LEN(T$)*2)/2):: FOR D=1
TO LEN(T$):: C=32+((ASC(SEG$(
T$,D,1))-64)*4)
20001 CALL HCHAR(ROW+1,LC+1,C
+3):: CALL HCHAR(ROW,LC,C)::
CALL HCHAR(ROW,LC+1,C+2):: CA
LL HCHAR(ROW+1,LC,C+1):: LC=L
C+2 :: NEXT D :: SUBEND
20002 SUB MBIG(ROWS,A$,B$,C$,
D$,E$,F$,G$,H$):: TR=(24-((RO
WS*2)+ROWS-1))/2 :: CALL BIG(
TR,A$):: IF ROWS=1 THEN SUBEX
IT
20003 TR=TR+3 :: CALL BIG(TR,
B$):: IF ROWS=2 THEN SUBEXIT
ELSE TR=TR+3 :: CALL BIG(TR,C
$):: IF ROWS=3 THEN SUBEXIT E
LSE TR=TR+3 :: CALL BIG(TR,D$
)
20004 IF ROWS=4 THEN SUBEXIT
ELSE TR=TR+3 :: CALL BIG(TR,E
$):: IF ROWS=5 THEN SUBEXIT E
LSE TR=TR+3 :: CALL BIG(TR,F$
):: IF ROWS=6 THEN SUBEXIT
20005 TR=TR+3 :: CALL BIG(TR,
G$):: IF ROWS=7 THEN SUBEXIT
ELSE TR=TR+3 :: CALL BIG(TR,H
$)
20006 SUBEND
20007 SUB SMALL(ROW,T$):: LC=
INT(((32-LEN(T$))/2)+1):: FOR
 D=1 TO LEN(T$):: C=ASC(SEG$(
T$,D,1))
20008 IF C>63 AND C<91 THEN C
=((C-64)*2)+32 :: CALL HCHAR(
ROW+1,LC,C+1):: CALL HCHAR(RO
W,LC,C):: GOTO 20010 ELSE IF
C>96 AND C<123 THEN CALL HCHA
R(ROW+1,LC,C-11):: GOTO 20010
20009 IF C>47 AND C<64 THEN C
=((C-47)*2)+110 :: CALL HCHAR
(ROW+1,LC,C+1):: CALL HCHAR(R
OW,LC,C):: GOTO 20010
20010 LC=LC+1 :: NEXT D :: SU
BEND
20011 SUB MSMALL(N,A$,B$,C$,D
$,E$,F$,G$,H$):: SR=INT((24-(
(N*2)+N))/2):: CALL SMALL(SR,
A$):: IF N<2 THEN SUBEXIT ELS
E SR=SR+3 :: CALL SMALL(SR,B$
)
20012 IF N<3 THEN SUBEXIT ELS
E SR=SR+3 :: CALL SMALL(SR,C$
):: IF N<4 THEN SUBEXIT ELSE
SR=SR+3 :: CALL SMALL(SR,D$):
: IF N<5 THEN SUBEXIT ELSE SR
=SR+3
20013 CALL SMALL(SR,E$):: IF
N<6 THEN SUBEXIT ELSE SR=SR+3
 :: CALL SMALL(SR,F$):: IF N<
7 THEN SUBEXIT ELSE SR=SR+3 :
: CALL SMALL(SR,G$)
20014 IF N<8 THEN SUBEXIT ELS
E SR=SR+3 :: CALL SMALL(SR,H$
)
20015 SUBEND
20016 SUB PR(A$):: PRINT RPT$
(" ",INT((28-LEN(A$))/2))&A$
:: SUBEND
30000 SUB BIGTEXT :: CALL CHA
R(32,RPT$("0",64))
```

```
30001 CALL CHAR(36,"030404050
909091212132525246848780202 0
A09090904848C80404E412121E")
30002 CALL CHAR(40,"FF4040272
424272525242424274040FFFC0201
F10912E40402F10909F10204F8")
30003 CALL CHAR(44,"1F204047"
&RPT$("48",8)&"4740201FFF0202
FC"&RPT$("0",16)&"FF0101FF")
30004 CALL CHAR(48,"FF4040272
424242424242424274040FFF80402
F10909090909090909F10204F8")
30005 CALL CHAR(52,"1F2040474
8484F4A4A4948484740201FFE0202
FE0000F80404FE0000FE0202FE")
30006 CALL CHAR(56,"1F2040474
8484F4A4A49484848484878FE0202
FE0000F80404FE")
30007 CALL CHAR(60,"1F2040474
848484848494848474020 1FFC0202
FF00007F8181F31212E20204F8")
30008 CALL CHAR(64,"407048484
8484F4A4A494848484848781E1212
121212F20202F2121212120E02")
30009 CALL CHAR(68,"3F10100E0
2020202020202023E10100FF80404
7E4040404040404040780404FE")
30010 CALL CHAR(72,"070202010
000000000007844232018 07FF0101
F712121212121212E20204F8")
30011 CALL CHAR(76,"784848484
8484F40404F4848484848780F0912
244890204040209048241209 0F")
30012 CALL CHAR(80,"78"&RPT$(
"48",11)&"4F40407F"&RPT$("0",
24)&"FC0202FF")
30013 CALL CHAR(84,"406050484
442494C4A4948484848704002060A
122242923252921212121212121E")
30014 CALL CHAR(88,"406050484
44A4D4A4948484848487040020E12
121212129252B25222120A0602")
30015 CALL CHAR(92,"071820234
448484848484844232018 07E01804
C422121212121222C40418E0")
30016 CALL CHAR(96,"7F40404F4
8484F4A4A49484848484878FC0202
F10909F10202FC00")
30017 CALL CHAR(100,"07182023
44484848484B494423201807E0180
4C4221212121292528B28C021AEF")
30018 CALL CHAR(104,"7F40404F
48484F4A4A494848484848 78FC020
2F10909F10202CC22121109090F")
30019 CALL CHAR(108,"0F102047
48484720100F0000FF40403FFC020
2FF0000F80402F10909F10204F8")
30020 CALL CHAR(112,"7F40407F
020302020202020202020203FE020
2FE008040404040404040404 0C")
30021 CALL CHAR(116,"4070"&RP
T$("48",9)&"444320100F101E"&R
PT$("12",9)&"22C20408F0")
30022 CALL CHAR(120,"40704824
242412121209090905040403 1E121
224242448484890909 0A02020C0")
30023 CALL CHAR(124,"F0909090
484848482525262411121121C0F090
90912121212A4A4642488484838")
30024 CALL CHAR(128,"78484824
241213080813122424484878 1E121
2242448C81010C848242412121E")
30025 CALL CHAR(132,"78484824
241212090402020202020231E121
224244848902040404040404 0C")
30026 CALL CHAR(136,"3F2020FF
000102040408112247408 0FFFF020
4C4881020404080000FF0202FC")
30027 CALL CHAR(140,RPT$("0",
24)&"3E11110F"&RPT$("0",30)&"
80")
30028 SUBEND
30030 SUB SMALLTEXT :: CALL C
HAR(32,RPT$("0",32))
30031 CALL CHAR(34,"101028282
8282844447C448282828282F84442
424242445844424242424244F8")
30032 CALL CHAR(38,"3E40"&RPT
$("80",12)&"403EF844"&RPT$("4
2",12)&"44F8")
30033 CALL CHAR(42,"FE8080808
080F88080808080808 0FEFE8080
808080F8"&RPT$("80",9))
30034 CALL CHAR(46,"3E4080808
0808F828282828282824438828282
828282FE828282828282828282")
```

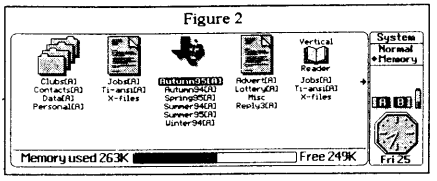
```
30035 CALL CHAR(50,"7C"&RPT$(
"10",14)&"7C3E"&RPT$("08",9)&
"88888888887")
30036 CALL CHAR(54,"82848890A
0A0C0A0A090908888848482"&RPT$
("80",15)&"FE")
30037 CALL CHAR(58,"82C6C6AAA
AAA92929292929292929282C2C2
A2A2A2929292928A8A8A868682")
30038 CALL CHAR(62,"3844"&RPT
$("82",12)&"4438F884828282828
4B8"&RPT$("80",8))
30039 CALL CHAR(66,"384482828
2828282828282928A443BF88482
82828284B88884828282828282")
30040 CALL CHAR(70,"3C4280808
0804038040202020824438FE"&RP
T$("10",15))
30041 CALL CHAR(74,RPT$("82",
14)&"443882828282824444444428
282828101010")
30042 CALL CHAR(78,"828282828
28282829292546C6C6C4444828244
444428281010282844444482282")
30043 CALL CHAR(82,"828244444
42828"&RPT$("10",9)&"FE020404
040808101020204040 80FE")
30044 CALL CHAR(86,"102828447
C828282FC8282BC828282FC7E8080
808080807EFC828282828282FC")
30045 CALL CHAR(90,"FE8080F88
08080FEFE8080F8808080807E8080
9E8282827C828282FE82828282")
30046 CALL CHAR(94,"381010101
01010383F0202020202827C828488
F088848282"&RPT$("80",7)&"FE"
)
30047 CALL CHAR(98,"82C6C6AAA
A92929282C2A2A2928A86827C"&RP
T$("82",6)&"7CFC8282BC8080808
0")
30048 CALL CHAR(102,"7C828282
829A867FFC8282BC828282827E808
07C020202FCFE"&RPT$("10",7))
30049 CALL CHAR(106,"82828282
8282827C828282444428281082828
2445454282882442810102844 82")
30050 CALL CHAR(110,"82442810
10101010FE040810102040FE")
30051 CALL CHAR(112,"38448686
8A8A8A9292A2A2A2C2C244381030"
&RPT$("10",13)&"38")
30052 CALL CHAR(116,"38448202
020404080810102020404 0FE38448
2020202041808040202028244 38")
30053 CALL CHAR(120,"08181828
2848488888FE08080808081CFE808
0808080403008040202028244 38")
30054 CALL CHAR(124,"38408080
80808080B8C48282828244 38FE020
2040404040808080810101010 1")
30055 CALL CHAR(128,"38448282
8282443844828282828244383844 8
2828282463A0202020202020438")
30056 CALL CHAR(132,"00000018
180000000000001818"):: CALL C
HAR(134,RPT$("0",24)&"1808081
0")
30057 CALL CHAR(136,"10102020
204040404040402020201010"&RPT
$("0",14)&"FE000")
30058 CALL CHAR(140,"10100808
080404040404040808081010"&RPT
$("0",28)&"1818")
30059 SUBEND
```

Happy tapping. Apologies if I've missed out any substitution characters, but I suppose it adds to the fun!!! You will have to experiment!

Well what's this about an experiment? Well, Richard Speed has asked me to substitute my article in PC form! Ah! "Religious Sacrilage!!" you're all shouting, but my article this quarter has not touched my GENEVE, but it also hasn't been produced on a PC. It's been produced on a multi-tasking operating system that is 1000 times more powerful than Windows 95, and that's, on my organizer! Yes, even the diagrams. I'm even able to create a dedicated icon (See Figure 2.) and file-list dedicated to my articles, since I normally type them on my organizer anyway, and then upload them to the GENEVE for proper formatting and

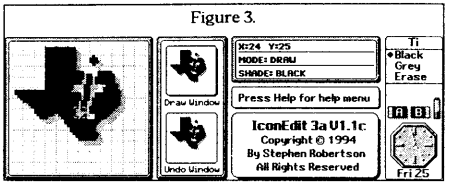
printing. This time though it's been done entirely with my organizer.

Figure 2



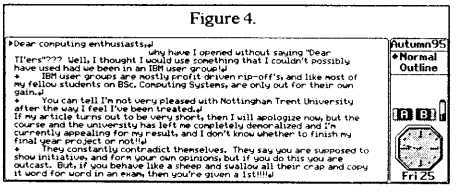
did I draw it? I used the icon designer that also runs on my organizer, and is shown in Figure 3!!!

Figure 3.



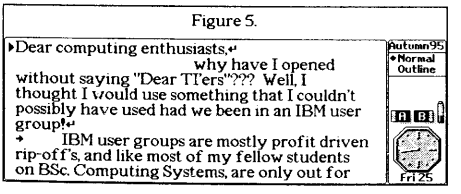
The organizer's word processor even gives 80-column text as shown in Figure 4.

Figure 4.



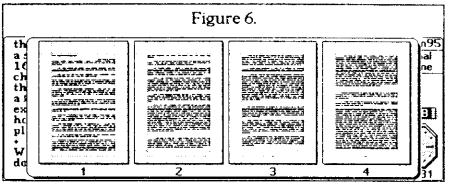
But if it's too difficult to read, then check out Figure 5!!!!

Figure 5.

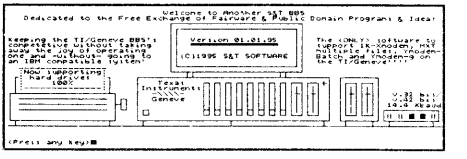


And when you've completed your document and want to see what it looks like, then check out Figure 6!!!

Figure 6.



And if that's not enough, then I've just used it to access the bulletin board!!!



I'd like to see Microsoft producing an operating system as advanced as that and still fit it into 2.5 Megs, complete with all the applications and data and everything!!!!

That's enough complaining about Bill Gates and the state of the world for this issue. Sorry no news on the SCSI card this quarter. I promise I will have a go at getting it in my box before the Winter issue.

Richard Twyning over and out

From The Chairman's Chair

I am back with you this quarter, after a long hair tearing time trying to sort out the BBS and its silly quirks. Life is not easy you know when it comes to Comms. As usual there is no real standard and you have to choose the best of the best, or rework it to make it do what you want. I will expand on that at a later date.

Issue 49

I wish to say to Richard SPEED thank you for taking up the post as Editor. The first of his issues No 49 was a lovely issue. I love the top and bottom reference to the magazine. What I would like to see is some more listings in the magazine for the lower end machine. I will be putting in some BASIC stuff this issue. So look around and find it.

I would also like to say thankyou to Mark Wills for taking on the job as Membership Secretary which can be a real pain at times. I believe he has had to sort out the data base onto his PC as he has problems with his TI at the moment.

The BBS

Back to the BBs.... The BBs has been running only a short while and is only on line at the weekends. (Fri 7pm - 10pm Sat & Sun 10pm -10pm). This so far has been a success, with a total number of listed members at the time of writing as 8. However there will be more as the board becomes used by outsiders. We will be making a small charge to non members of the group to help the BBs to run.

At this time I have two HARD DRIVES on line, a 20meg and a 57meg. On the board there are areas for TI, Amiga and IBM. You can get onto the board with any machine however the download and upload areas have to be accessed with the correct X MODEM CRC format to make it work properly.

I am going to write to the Tim Tesh, the man who wrote the software to see if he can put Z modem on the system. I will also be putting on a clock which will make the board look real good very soon. Apart from that everything is up and running with a :-))) Big smile!

I will now go back in time for a little while and do some basic programming which I know a lot of you out there enjoy.

Printing with BASIC

What I am first of all going to cover is that of Printing with Basic. This is at first sight very restricted to a scrolling type input such as :.

```
100 PRINT "hello"
110 GOTO 100
```

As we all know this goes up the screen and goes on forever. To be a little better you can put a line at **50 CALL CLEAR** and change the line **110 GOTO 100** to **110 GOTO 50**, and the 'hello' stays at the bottom.

That is ok if you want it at the bottom. What do you do if you want it some where else on the screen? Put it in Extended basic I hear a wag saying, OK - "But my Extended BASIC broke down" or "I just started with the machine".

Well you do it like this.

```
100 CALL CLEAR
110 ROW =10
120 COL =1
130 A$="HELLO HOW ARE YOU"
140 GOSUB 1000
150 GOTO 100
1000 FOR I=1 TO LEN(A$)
```

```
1010 CALL HCHAR(ROW,COL+1,ASC
(SEG$(A$,I,)))
1020 RETURN
```

This will now allow you to place your text anywhere on the screen with the **ROW COL** coordinates and then print out the screen text. When this performs it comes out like a typewriter.

The **PRINT** command also has other little tricks to it. There are things called operators that tell the computer what type of output to use. Try the following program.

```
100 CALL CLEAR
110 PRINT "HELLO ";
120 GOTO 110
```

WHAT HAPPENS? Well the **;** tells the computer to print along the screen until the screen is full with the complete word and then scrolls round to the next line. This can be very useful when you wish to print out a matrix of numbers like this.

```
100 CALL CLEAR
110 FOR A=10 TO 44
120 PRINT A;
130 NEXT A
```

If you put in single numbers you will find that they will drop out of line as the gaps are smaller. To cure this you add a space like this

```
100 CALL CLEAR
110 FOR A=1 TO 9
120 PRINT " ";A;
130 NEXT A
140 FOR A=10 TO 44
150 PRINT A;
160 NEXT A
```

You can play with this to your hearts content and work out all sorts of display ways. Next you can manipulate the print setting in mid stream. on the program above try **120** as **PRINT " ":A:** or **PRINT " ":A** See what happens.

The print command can also be used to place graphics on the screen in a very successful way. This can be done with redfined letters and printed onto the screen. The other way is to use data statements and redifined characters and display with the **CALL HCHAR** routine. In the following program both types of printing are shown. This program prints out a horse graphic.

```
100 REM HORSE
110 CALL CLEAR
120 FOR C=96 TO 126
130 READ C$
140 CALL CHAR(C,C$)
150 NEXT C
160 DATA 0000000001010103,422
27DFFFFFFFFBF,0060FEFFFFFFFF
F,0000000008080E0F,0300000103
070707
170 DATA 67E7EFFFFFFFEFD,FFF
FFFFFFFFFFF,F0E0F0F0F0F0F2F
E,000303071F1F3F3F,000080C0E0
F0F0F9
180 DATA 70000000000000E,8303
0307070F1F3F,7E7EBCB8B8B080C,
0003030100010703,7FFFFFFFEFE
FEFE
190 DATA FFFF9F3F7F7F7F7F,C0C
0C08080C6FFFF,BE3E3EFFFF3F3E3
E,3F3F1F1F1F1F3F7F,F7F7F7EFDF
BF8
200 DATA FFFFFFFFFEFCE,FFFFFC
F0F0F0F0F,CF8F1F3E7CF8F0E,010
30301,7EFCFCF8,7F3F0F07030301
01
210 DATA F8FEFFFF8E80C0C,F0F0
F0F0F0F0F078,030303010101,E0E
0E0F0F8F87,787C3C
220 PRINT "METHOD 1:";TAB(15)
;"METHOD 2:"
```

```
230 PRINT : : : "{3 SPACES} a
bc":"{3 SPACES}defg":" hijkfl
"
240 PRINT "mnofffp":" qrstuv"
250 PRINT "wxyz {":"  |} ~":
: : : :
260 FOR I=1 TO 35
270 READ X,Y,G
280 CALL HCHAR(X,Y,G)
290 NEXT I
300 DATA 13,21,96,13,22,97,13
,23,98,13,24,99,14,21,100,14,
22,101,14,23,102,14,24,103
310 DATA 15,19,104,15,20,105,
15,21,106,15,22,107,15,23,102
,15,24,108,16,18,109,16,19,11
0
320 DATA 16,20,111,16,21,102,
16,22,102,16,23,102,16,24,112
,17,19,113,17,20,114,17,21,11
5
330 DATA 17,22,116,17,23,117,
17,24,118,18,18,119,18,19,120
,18,20,121,18,21,122,18,23,12
3
340 DATA 19,20,124,19,21,125,
19,23,126
350 GOTO 350
360 STOP
```

When you type this in you will find that this will show you Method 1 which will fill in the picture in lines across screen scrolling upward. Method 2 uses the **HCHAR** statement to put the graphic onto the screen. With the print section you will note that the letters 96 through to 126 are placed on the screen. These being redefined by the **CHAR** statement allow you to print onto the screen.

There is just one more thing which is **TAB**. This acts like a TABULATOR on a typewriter and jumps along the line with no print for the spaces given ie **TAB(10)** jumps 10 spaces.

Look at the above program and you will see its use.

MEMORY FULL FCTN QUIT.....

The MOBB BBS Special Offer

How much would you like your own MODEM?
Be able to talk to other group users?
Have fun wth on-line games?
Download free software into the PD Domain?
Get useful info on different subjects?
Be part of something?

Well you can. The TIUG (UK) OFFERS YOU the chance to purchase at a very special price the following MODEMS which offer the following:

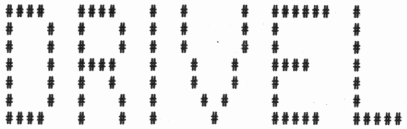
1. 14400bps and 9600 bps fax
2. 9600bps only

Both modems are fully BABT approved.

No.1 is £87 inc post, packing and VAT.
No.2 is £73 inc post, packing and VAT.

Contact Richard Twyning or Trev Stevens.

The small profit goes to Group Funds.



From The Vice Chairmans Very Own Clapped Out Keyboard...

Greetings once again. Been a long time since I wrote a darn thing forthe Mag, what with the birth of our our second child, moving house,working in foreign countries... it's amazing how the time goes. I really do think the older you get, the faster it goes! (It's a shame our TI's don't go faster as they get older!)

For those of you who need to contact me, you can either write to me (address on front cover) or mail me on the groups BBS. My user number is number 5. You are welcome to phone me at home as well on 01743 350588.

Membership News

Those of you that renewed your membership last quarter will know that I am now the membership secretary. Therefore, if you get the 'reminder' letter in with your mag, please send your re-newals to me.

Many thanks to all those that have renewed. I hope you are all happier with the new magazine layout. RICHARD SPEED has done an amazing job. Now that the group has had some internal re-organisation, and what with the groups BBS, things are really looking up for the group.

Thanks also to those that have given contributions in addition to their re-newal fee. The contributions will be sent to Stephen Shaw who will put them to good use in the procurement of new software for the groups disk library.

Renewals Last Quarter

'Welcome back to the fold' goes out to the following persons who have renewed their subscriptions: As of 26th August 1995 31 members have re-newed. A total of 16.00 pounds in voluntary donations has been given by the members.

John Bingham Roy Robinson Ian Hewitt Mr. Strong Sam Wardle John Murphy Graham Steward Walter Allum Mike Poskitt Victor Reinar Derek Hayward Bill Moran Stanley Moran David McCann Daryl Muncy Ross Bennett Francesco Lama Edwin George Edward Shaw Peter Jackson Thomas Southwell Ian Kileour Charles Skrzynski Alan Bray Roger Nicholl Alan Rutherford William Woermijer David Caine James Troy Terry Leach John Dunning James Murta Thomas Norman Douglas Moller

Nice to see Edward Shaw re-newing his membership. Edward did a sterling job a few years back as Module Librarian for the group. Welcome Edward.

A special welcome to Mr. Douglas Moller from Aitkenvale, Australia. Thanks for your kind comments about the group Mr. Moller. It's nice to know that our efforts are appreciated.

Many thanks to Ross Bennett and his wife (also ex committee members) for their kind comments about the group.

Unfortunately, due to the change over of membership post from Alasdair Bryce to myself, there have been problems with members details going astray/not being kept up to date etc. A few members have contacted me to say that they have sent cheques to Alasdair but they have not been

cashed and they have not received any more magazines. In addition, we have had problems with the groups data base getting corrupted.

If your supply of magazines suddenly dries up without notice, or you get a reminder letter when you shouldn't then PLEASE CONTACT ME IMMEDIATELY so that I can change your details. We will do our best to get any issues that you have missed to you.

Drivel Drivel Drivel Drivel

Readers will have noticed that the subject of communications has dominated the last few issues, due mainly to the group getting it's phone and BBS installed.

I work for a company called Merlin Research writing communications software. It is my job to get two computers that are completely alien to one another (and use their own transmission / packet protocols) and make them talk. I do this using a third computer (phew!) which can understand the protocols of both the machines in question and translate them as the messages are sent and received. Literally a translator / interpreter that sits in between the machines and converts the data so that they can both understand them. Clever it is. Easy? Not blummin' always!

Anyway, I digress. I thought this issue, I would look at making your TI talk other computers. Namely a PC. Many of you have PC's at home in addition to your TI. Perhaps you have a laptop/notebook PC that you use at work, or perhaps the wife has one to hold the membership details of her monthly 'Blue Rinse, Cup Of Tea, Bit 'O Knitting and Good 'Ol Gossip Club'. More likely though, your five year old son or daughter has one do his / her homework on the investigation of Black Holes and their ability to bend light, matter, life, the universe and everything!

Any computer worth it's salt has a communications port. In the early days they were less common (certainly on home machines) because computer communications was "for the professionals man" and not something that the everyday space invader player was interested in.

TI, sensibly, gave you the option. Computers in the late seventies and early eighties were pretty expensive, so it was a good idea for TI to develop the Home Computer in the modular, expandable way that they did. You bought what you could afford, when you could afford it. What's the point in an RS232 when all you want to do is play Alpiner (my favourite TI game!)

Nowadays, the price of IC's is so low, just about any machine built this side of 1990 will have one or maybe two communications ports (The TI card has 2).

TI's and PC's have an RS232 port. Apple Macs have an RS422. RS232 is the world wide adopted standard for short distance communications (up to 50 metres), RS422 is for medium distance (200 metres) and RS485 (which I work with) works up to about a mile. RS485 and RS422 use a line differential system to communicate. They have two RX lines and two TX lines, whereas RS232 only has one. In RS422 and 485, as one line goes high, the other goes low. If the DIFFERENCE between them is more than (I think) 200 milli-volts then it is deemed a valid data pulse.

Using this system, you can see that data transmitted using this system is highly protected from noise, as any noise (say from mains cable) would have to induce a

voltage of over 200 milli-volts in order to corrupt the data travelling along the wires.

RS485 uses low impedance cable in order to minimize the volt dropalong long lengths of cable, helping to ensure that data can be carried long distances. In addition, RS485 transmitters tri-state themselves from the transmit lines when they are not talking, but always remain connected to the receive lines (the listen lines). This allows what is known in the industry as Multi-Drop systems to be used with RS485 (that is multiple machines connected to the same, common RS485 data lines).

Machines on a factory floor might be hooked up to an RS485 system with a 'data monitor terminal' collecting data from all of the machines and displaying it on a screen for the factory operators (my job!). RS422 does not have this capability. Any way, I digress again...

Connecting Your TI to a PC

For what we are going to do, only three wires are necessary. RX (receive) TX (transmit) and GROUND (0V).

RS232 wiring convention states that the TX line from one machine is connected to the RX line of the other, and the RX line is connected to the TX line of the other. In other words the RX and TX lines are crossed over, so that when one machine sends something on its TX line (transmit) the other machine receives it on it's RX line. The ground (0V) line is simply connected together so that the two machines are referenced to each other.

Assuming your PC to have a NINE pin COM port, this is how to connect your TI to your PC:

PC PIN No	TI PIN No
2	3
3	2
5	7

That's all you need. This should be good for TI to PC communications at up to 9600 baud, and good for PC to TI communications at 2400 baud to 4800 baud. (The TI is a slower machine, and cannot handle large amounts of data in a short time, thus slower baud rates are recommended - unless you connect the handshake cables of the RS232 leads - beyond the scope of this article. I recommend that you run at 2400 baud - you'll be surprised at how fast it is.)

PC Terminal Emulation Software

I will discuss talking to your TI via PROCOMM and Windows Terminal.

If you want a copy of the shareware version of PROCOMM for the PC, send me a PC disk (3&1/2" or 5&1/4") and a STAMPED ADDRESSED ENVELOPE and I will send you copy. TELCO is available from the group library at the normal rates.

First connect the two machines up via the cable. (Have both machines turned off.)

Setting the Baud Rate

This is the speed at which the data travels between the two computers. Both machines must be talking at the same speed or else they will not be able to communicate. Along with the baud rate, we shall set the number of stop bits and the Parity, and number of data bits. This forms part of the 'Communication Protocol', too involved for this article, but suffice it to say that if both machines are not set the same, you may be able to communicate but you will get false errors

being reported, and may notice 'rubbish' characters on the screens of the machines when you transfer text (especially if the text contains extended characters/control codes etc).

We are going to communicate at 2400 baud, 8 data bits (thus character codes 0 to 255 can be transmitted - essential for binary transfers, not essential for text transfers), No parity, and 1 stop bit. The techno speak for this communication protocol is 2400 8 N 1

With this protocol, you should have no problems communicating at all.

Procomm (PC)

Press ALT P Then press 9 and ENTER Select the appropriate number number for the COM port you are using and then select the number for save changes. Procomm will be set to these settings each time it loads.

PRESS ESC to return the Main Screen.

Windows Terminal

Click on the SETTINGS bar at the top of the windows, and then click on Communications. Click the appropriate buttons for Baud, Parity, Stop Bits, Serial port etc. *(Editor's note - Hyper Terminal, which is included with Wndows 95 works a treat and supports ANSI graphics too...)*

TELCO (TI)

When TELCO loads, select TERMINAL. When the Terminal loads, Press FCTN B for the terminal settings. Press Q for baud rate, then select 2400 baud.

Do the same for STOP BITS and PARITY, select 1 stop bit and no parity, and 8 DATA BITS.

Lets Talk

Lets have a look and see if we can 'Talk' from one machine to the other. We will also change a few settings and discuss them on the way such as line feed codes, carriage returns etc.

Type something on the TI keyboard. Does it appear on your PC screen? If not, then you've done something wrong, go back through the info above and check.

If it appears on the PC screen, but you don't see anything on your TI screen, then you need to change the ECHO settings. Hit FCTN B, and you will see the option on the menu, which you can toggle between on and off.

Now press a key on your PC. You should see it on your TI screen. If you see it on your TI but don't see it on your PC then :

Procomm:
Press ALT E to toggle Duplex
Windows Terminal:
Click on Settings then click Terminal Preferences. Click on the Local Echo button

Transferring Text Files

This is an aspect communications which always confuses new-comers.

Many times, when sending a text file from one machine to another via a terminal program, the text does not appear to arrive in the way that it was sent. Sometimes there is a blank line in-between each line of text, and sometimes all the text looks like one giant paragraph etc etc. This is all to do with the settings on the remote machine and your machine. When transferring TEXT, one will come across LINE FEED and carriage return options. A good rule of thumb is to set both machines to perform no translation on

either in-coming or outgoing text, that way you can guarantee the text is received in the way it was sent.

The other fool proof way of capturing text files is to download or upload them using a binary transfer method such as XMODEM. (Binary transfer protocols are usually used for sending and receiving programs and data but there is nothing to stop you sending text using the same protocols). XMODEM and it's counter parts perform no translation at all on any data, due to the type of data that they are sending/receiving. Ie. Programs. One hardly hardly change the contents of a program file and still expect it to work!

Transferring using XMODEM

The method is the same which ever way around you do it (TI to PC or PC to TI):

TI to PC

Select UPLOAD on TELCO, then select XMODEM Select DOWNLOAD on your PC program, then type a file name for the file that is going to be received. On your TI, type the file name of the file that you want to send and hit ENTER. The file will be sent from your TI to your PC.

PC to TI

Select upload on your PC, then select XMODEM. Select Download on TELCO, then XMODEM. Enter the filename to receive on your TI and hit enter. On your PC, give the file name of the file to send. The file will be sent from your PC to your TI.

Waffle Waffle Waffle

That's all for communications for this issue. If you have any difficulties then give me a call or write. Further, if you need a cable made to connect you machines together, send a stamped addressed envelope to me with a fiver in it to cover cost of parts and I'll make you one and send it back to you. Copies of PROCOMM (shareware) can also be supplied but only if you send me a disk (3 1/2" or 5 1/4")

Over and Out.

WANTED

Myarc Geneve 9640 card for TI Expansion Box with appropriate DOS software.

WILL PAY UP TO 150 POUNDS

Please contact Mark Wills on 01743 350588 if you have a Geneve that you wish to sell. Can collect if neccessary.

Overseas offers welcome. Address on front cover.

Cracking The Code

This article is in response to Walter Allums request last issue for information about interfacing Assembly Language programs to Extended Basic Programs. I'm sorry we seem to have lost touch Walter, will contact you soon. Got your letter about your membership details. Thanks!

It's Easy. If you've managed to do it with the editor assembler (oooh Matron!) and TI basic, or MINI MEMORY and TI Basic, then you're a good 95% of the way there.

It's A Bummer

There are only a few differences between the Editor Assembler and TI Basic environments, unfortunately, they are enough to totally stump the new-comer to Assembly Language and hence this article. The topic is covered in the Editor Assembler manual, but not directly. If memory serves me correctly, there is a section on the Extended Basic Object Code Loader (it's been eons since I looked at the Editor Assembler Book - (hangs head in shame))

Basically, the Extended Basic loader is naff. It cannot load Compressed Object Code (another subject!) and it does not support the REF directive (heavy bummer), additionally, it is SLOOOWWW.. you begin to get the picture!

It's important to realise that the way you write your code does not change as such. It's the way you assemble it that differs, although the addresses of all the Basic to Assembly Language Utilities change in addition to things such as the sprite attribute list etc etc, which can give rise to some fairly serious cranial agitation!

For example,the memory mapped ports are still in the same place (GROM READ/WRITE etc), but the address of the VSBW routine is totally different. I will give a list of the Extended Basic utility addresses later.

So What are you REFering to?

So where's this all going I hear you ask? Well guv, it's the REF directive innit. Yeah. Know wot I mean? It wanna work wiv du REF directive.

Extended Basic does know about it and does not care about it. If it sees a REF tag in the object code that it is trying to load, you get the old two fingered salute from your TI!

Let's have a look at what the REF directive is. It's rather clever, and I would suggest that when TI implemented it, it was years ahead of it's time.

Here's a piece of Assembly Language that will work in the editor assembler environment:

```
        REF VSBW
        DEF START
MYWS    BSS 32
START   LWPI MYWS
        LI R0,767
        LI R1,>2000
LOOP    BLWP @VSBW
        DEC R0
        JNE LOOP
LOCK    B @LOCK
```

Those of you who know machine code will know that the above routine will clear the screen, starting at the bottom right and finishing at the top left ('cause it's quicker that way - look at the code and think about it. Answers on a postcard please!)

You would load it from TI BASIC with CALL LOAD and then do a CALL LINK("START") to call it. It would clear the screen (fast) and then lock up.

This would NOT work in extended basic however. Not because the code is wrong, but because the XB loader does not recognize the REF directive.

The REF directive

The REF directive allows the programmer to REFer to certain system utilities by name rather than by address. Which line of code would be more understandable twelve months after you wrote it:

```
BLWP @VMBW
```

or

```
BLWP @>20C0
```

(**20C0** is just an example address)

There are many utilities tucked into the TI's ROM that the programmer can refer to by name GPLLNK, DSRLNK, VSBR, VWTR etc etc.

Why Not Use the EQU Directive?

This is a good question. You would be quite right to say that you could accomplish the same using the following:

```
VMBW    EQU >20C0

        ....
        ....
```

(later in program)

```
        BLWP @VMBW
```

Portability Mate. It's Portability Innit!

The idea is as clever as it is sophisticated. The three main modules (Editor Assembler, Mini Memory, and Extended Basic) all have pretty much the same number of utilities tucked into them (I think the XB module has some of them missing... may be wrong though). The problem is, that the vectors for the utilities are all at different addresses depending which module you have plugged into your rather old and somewhat dodgy and decrepit module port (if it's anything like mine).

All this spells headaches for the Assembly Language Programmer. It means that if he wants his machine code program to run in any of the modules, he must have a version for each that has the correct address of all the utilities he is using.

TI saw this problem and set about fixing it. Hence the REF directive.

When you include a REF in your source code, the assembler makes a note of it in the resulting object file (have a look at the end of a DF80 file with the funnelweb program. Notice anything?)

When you load that object file into BASIC with the CALL LOAD command, those REF's say "Oy. I'm gonna be using VSBR etc etc"

The object code loader (which is built into the module that is plugged in) know's which address corresponds to which utility. For example, if the ED/Ass module is in use and it comes across a KSCAN ref, each time it sees KSCAN in the object file, it will swap it for the correct address of the KSCAN utility - ON THE FLY as your object code loads!! Really NEAT!

This means that the very same object file will work on both the Mini Mem and the Ed/Assembler module! No re-programming and having to remember loads of different addresses! Cool or what!

The Bad Apple

(Music from the film JAWS plays while a scene is shown depicting a poor Assembly programmer trying to get a program that is fine under Ed/Assembler but just wont have it under XB to work...

Our programmer looks around in sheer disgust and exasperation. Why won't XB even LOAD the object file? He looks at the error message again, and the looks at his source code. Oh No! It's those REF statements... (change music to the music from Physco)... Fade to Black.

A new scene. The programmer sits happily in his padded cell playing a game of Tetris on his Game Boy...Aaah, that's better!

In an effort to squeeze all the functionality of XB in, they had to kick a few bells and whistles out. One of them was all the funky things in the object code loader.

The object code loader DOES NOT SUPPORT THE REF DIRECTIVE so don't use it. See below for the solution. The DEF directive is allowed however.

Also please note that the BSCSUP file found on the editor assembler disk IS NOT REQUIRED under Extended Basic, so again, don't use it. It won't work anyway due to the fact that the object code contains REF directives which, as I have been at pains to point out, is not supported!

Machine Code in the XB Environment (spit!) Here is the above program, written to work in XB.

```
        DEF START

VSBW    EQU >2020
MYSW    BSS 32
START   LWPI MYWS
        LI R0,767
        LI R1,>2000
LOOP    BLWP @VSBW
        DEC R0
        JNE LOOP
LOCK    B @LOCK
```

You'll notice that all I've done is declare VSBW myself using an EQU directive. This is all you have to do. Of course this requires you to know the correct addresses of each of the utilities, so in the best Blue Peter tradition, here's a little list that I prepared earlier...

The numbers in brackets are the Editor Assembler equivalents. You will find a similar table in the Editor Assembler manual, but please note that the value for CFI (convert floating point to integer) in the book is WRONG!! Just about the most important one too as that's the one you use when you pass numbers from your BASIC program to your machine code program! You can trust TI to cock it up!

The Mark Wills Magic Table of Extended Basic Equates:

-

```
VSBW    EQU >2020   (6024)
VSBR    EQU >2028   (602C)
VMBW    EQU >2024   (6028)
VMBR    EQU >202C   (6030)
NUMASG  EQU >2008   (6040)
NUMREF  EQU >200C   (6044)
STRASG  EQU >2010   (6048)
STRREF  EQU >2014   (604C)
KSCAN   EQU >201C   (6020)
CFI     EQU >12B8   (????)
XMLLNK  EQU >2018   (601C)
GPLWS   EQU >83E0   (83E0)
```

```
STATUS EQU >837C   (837C)
CIF    EQU >2000   (????)
VWTR   EQU >2030   (????)
```

Okay. So now you know the addresses of the most used Extended basic utilities. Lets have a look at writing some assembly language programs and passing data between them and an Extended Basic program.

Passing numbers from XB to Assembly

Not hard. First the NUMREF routine is used to actually get the number from wherever XB keeps it into our machine code program.

You have to tell NUMREF which parameter you want from the CALL LINK statement. For example : CALL LINK("BLOG",X,Y) X is parameter 1, Y is parameter 2. You tell it by loading the parameter number in R1.

If you are passing whole arrays (not covered in this article) then you give the element number in R0. (We will be setting R0 to 0)

XB handles numbers in floating point (radix 100) format, so, before we can store the number in a register, we need to convert the radix 100 number to an integer.

Now the assembly language:

```
        DEF TRY

NUMREF  EQU >200C
XMLLNK  EQU >2018
CFI     EQU >2000
MYWKSP  BSS 32

TRY     LWPI MYWKSP
```

The radix 100 number, after it has been fetched from XB is stored in a place called the Floating Point Accumulator (FAC) and it's STARTING address is >834A.

After we have converted it to an integer, the integer will be stored at >834A, ready for us to load into a register.

The above addresses are constant and fixed. They cannot be changed by you, and you can be sure that the system will not change them at any time.

The Convert Floating Point to Integer routine (CFI) is accessed via the XMLLNK utility. This utility takes a parameter in the form of a number after the branch to XMLLNK in your code. Instead of a number however we will use the symbol CFI which we will have previously declared using an EQU directive.

Here is a program in assembly language which will take a number from XB and add 1 to it, leaving that number in R8:

First an example XB program to call our assembly program:

```
10 CALL INIT
20 CALL LOAD("DSK1.TEST/O")
30 A=145
40 CALL LINK("TRY",A)
50 END
```

```
        CLR R0          not an array element-normal variable
        LI R1,1         first parameter from call link
        BLWP @NUMREF    get the number in radix 100 from XB
                        (number starts at >834A)
        BLWP @XMLLNK    call XMLLNK routine
        DATA CFI         tell XMLLNK we are using CFI routine
                        (the integer number is now at >834A)
        MOV @>834A,R8   get integer in R8
        INC R8          add 1 to the number
LOCK    B @LOCK
```

Okay, so it's a boring program. But it demonstrates just how easy it is to do it. If there was another parameter in the CALL LINK statement, just add 1 to R1 (the parameter counter in the CALL LINK statement) and do the same again. Make sure that the parameters type matches though. Don't use NUMREF when you pass a string to your assembly language program!! You can see that you have to define the order of your parameters and stick to them. It is possible to determine the type of parameters passed, and the number of parameters passed from within your assembly language program, but it is beyond the scope of this article. Request it and I will write about in a future issue if anyone wants to know.

Returning Numbers from Assembly Language to XB

Building on from the first example, we will pass a number to assembly language, the assembly language program will add 1 to it, and then return it back to XB. this program also illustrates the correct way to return back to the XB environment.

To assign a number to a basic variable we use the NUMASG utility.

```
        DEF TRY

NUMREF  EQU >200C
NUMASG  EQU >2008
XMLLNK  EQU >2018
```

Again, R0 contains the element number, R1 the parameter, and the NUMASG utility expects the value to assign in the floating point accumulator (FAC) already in floating point format. We therefore have
to convert the number twice - once from floating point to integer so that we can add 1 to it, and then back to floating point so that we can assign it.

To convert from integer to floating point is very simple. We move the integer to the first word in the FAC and then call the CIF routine in XMLLNK.

First, the XB:

```
10 CALL INIT
20 CALL LOAD("DSK1.TRY/O")
30 A=145
40 PRINT "BEFORE ASSEMBLY LAN
GUAGE=";A
50 CALL LINK("TRY",A,B)
60 PRINT "AFTER ASSEMBLY LANG
UAGE=";B
70 END
```

Now the assembly:

```
CFI     EQU >2000
GPLWS   EQU >83E0
MYWKSP  BSS 32

TRY     LWPI MYWKSP
        CLR R0          not an array element-normal variable
        LI R1,1         first parameter from call link
        BLWP @NUMREF    get the number in radix 100 from XB (number starts
                        at >834A)
        BLWP @XMLLNK    call XMLLNK routine
        DATA CFI        tell XMLLNK we are using CFI routine
                        (the integer number is now at >834A)
        MOV @>834A,R8   get integer in R8
        INC R8          add 1 to the number

            (we now get ready to assign the number back to XB)
              (notice R0 is still zero-not an array)

        MOV R8,@>834A   move R8 to FAC
        BLWP @XMLLNK    call XMLLNK utility
        DATA CIF        tell XMLLNK we are using CIF

            (the value in R8 is now in floating point format in.FAC)

        INC R1          point to second parameter in CALL LINK
        BLWP @NUMASG    call numeric assign routine. XB now has the variable.
        LWPI GPLWS      get ready to return to XB. Load GPL.registers
        CLR @>837C      clear GPL status byte (YOU MUST DO.THIS)
        RT              return via GPL register R11
```

If you assemble this and run it in conjunction with the XB program, you should find that A contains 145 and B contains 146.

Next Issue

Next issue I'll show you how to send TEXT strings from XB to machine code. Just to wet your appetite however, here's a program in machine code for the XB environment that I wrote yonks ago which is basically a DISPLAY AT command ala XB. however, it uses the 32 column screen, not the 28 column screen like DISPLAY at.

Also, the third parameter is a direction parameter. Using this you can print text backwards etc. Here are some examples:

```
CALL LINK("PRINT",10,10,1,"HE
LLO")
```

RESULT **HELLO**

```
CALL LINK("PRINT",10,10, -
1,"HELLO")
```

RESULT **OLLEH**

```
CALL LINK("PRINT",10,10,32,"H
ELLO")
```

```
RESULT    H
          E
          L
          L
          O

CALL LINK("PRINT",10,10,-
32,"HELLO")

RESULT    O
          L
          L
          E
          H

CALL LINK("PRINT",10,10,31,"H
ELLO")

RESULT      H
           E
          L
         L
        O

CALL LINK("PRINT",10,10,-
31,"HELLO")

RESULT    O
           L
            L
             E
              H
```

You get the idea. It also illustrates the ASCII offset required when printing text on the screen from machine code. We'll discuss the program next issue. Ta Ta for now.

```
* PRINT ROUTINE:
* PRINTS A STRING AT A
* SPECIFIED Y&X LOCATION,
* WITH A SPECIFIED "GAP"
* BETWEEN CHARCTERS ALLOWING
* VERTICAL PRINTING (UP AND
*.DOWN) BACKWARDS PRINTING,
* ANGLED ETC.

NUMREF  EQU >200C
XMLLNK  EQU >2018
STRREF  EQU >2014
CFI     EQU >2000
GPLWS   EQU >83E0
MYWKSP  BSS 32
BUFFER  BSS 256

PRINT   LWPI MYWKSP
        CLR R0
        CLR R3
        LI R1,1
        LI R2,BUFFER
        BLWP @NUMREF
        BLWP @XMLLNK
        DATA CFI
        MOV @>834A,R5
        LI R4,32
        MPY R4,R5
        INC R1
        BLWP @NUMREF
        BLWP @XMLLNK
        DATA CFI
        MOV @>834A,R5
        A R5,R6
        INC R1
        BLWP @NUMREF
        BLWP @XMLLNK
        DATA CFI
        MOV @>834A,R7
        INC R1
        LI R5,>FF00
        MOVB R5,@BUFFER
        BLWP @STRREF
        MOVB *R2+,R3
        SWPB R3
        MOV R6,R0
LOOP    MOVB *R2+,R1
        AI R1,>6000
        BLWP @VSBW
        A R7,R0
        DEC R3
        CI R3,0
        JNE P1
        LWPI GPLWS
        CLR @837C
        RT
```

After my staggering success of having an article appear in TI*MES many issues ago,I feel committed to try to repeat the feat. I know that there's someone else out there with a 99/4A! This article was originally submitted for publication many moons ago, but for some reason never appeared in print. So, dusted of, and re-arranged, try again. For the confused, let me explain. Previous to sitting at the keyboard to write the article in issue 22, I had never submitted anything for publication. To have my meanderings and ravings printed was wonderfully rewarding. But, hello, what is this in the letter box? Not a gas bill,inland revenue summons, or even junk mail, but a genuine reply from an enthusiast in West Sussex. Leo Hughes, you are a diamond. His response to my primitive efforts with TI-Writer are reward aplenty. I hope that he is still a subscriber. Thanks also to the commitee members for correcting, reprinting and publishing my reminiscences.

Now I know how copy should be sent, this item should be correct, apart from my usual bad spelling. A rose by any other name and so on. For this article I am using left margin 5, right margin 75, page length 58, with fill adjust turned on. The extra space after full stops (and question and exclamation marks) is avoided by using a required space carat. *[Which Microsoft Word 6 has decided to completely ignore! - ed]*

Tempting Fate

All this elation was shortlived however, as after my assertion that computer equipment was reliable the last months have been the worst ever for faults developing. Perhaps I was tempting fate but everything has now been fixed and at risk of it happening again I will describe the problems. To those readers who find this of no interest I apologise; to stop my ravings you will just have to write an article yourselves. No doubt if there is a sudden glut of material for TI*MES, my articles will be the first in the editor's bin again!

Disk Repairing

The nightmare started soon after the last article was written. Attentive readers may remember a comment about my second disc drive not formatting discs. Whilst investigating this I realised how litte I knew about how our computer talks to the discs, so I researched and discovered that the drives are designated as 1,2,or 3 by a shorting plug on each drive. These were correctly set but I had a terminating network on both drives! Wrong! So remove network from drive 1 and now 2 would perform correctly. But now a problem which I had experienced on odd occasions became much worse. Drive 1 head would often not "load" into contact with the disc, giving drive error codes. Studying the circuit diagrams showed what to me seemed a complicated arrangement of an AND gate supplying the head load solenoid with a 20 msec one shot multivibrator switching a transistor on to overcome the inertia of the solenoid. Presumably this is to reduce the normal current load when the drive is reading or writing. The one shot was half of a 9602 chip and my comment about state of the art electronics backfired when I discovered that this is no longer available! However by some butchering, piggybacking a 74LS121 onto this chip, and working out the logic required, this was fixed. Who called it "logic" in the first place?

During all this testing and disturbance of the console a persistant crashing of the

system became apparent, 3 hours of searching the processor pcb with a magnifying glass, and twisting the board to find a slight squeak, produced a dry joint. Why this had not shown up years ago is a mystery. So, great, now it must work. Wrong, now the drive searched endlessly for the directory on the disc! At that I gave up for three weeks and struggled with one drive working, resigned to buying one or maybe two new half height drives.

New drives? Bah, Humbug!

The Scrooge instinct prevailed eventually and a close examination of the drive pcb revealed a FET with two broken leads, possibly due to handling of the drive whilst finding the first fault. Looking through my collection of discarded pcb's only produced P type FET's, so with nothing to lose I tried to solder two leads back on. This proved to be dificult as there was no wire sticking out of the device. Eventually I managed to attach one strand of flex to each lead but the FET was very hot by this time. I did warn you that I butchered! Even though I had to melt the plastic encapsulation and FET's are notoriously delicate to heat and static, it worked. When everything was reassembled, both drives worked fine, much to my relief.

Assuming that they would stay working for some time, another 100 discs were purchased, but double sided just in case I had to replace one or other of the drives. Fortunately there are few programs which demand double sided drives so the cash could have been spent on further expansion, maybe a ram disc which I am tempted to try to build. Anyone know how it is done?

The discs were a bargain, 22p each and a free 100 locking box too! They seem to be good quality, working fine at 80 track double density on the office Amstrads. On the subject of components, my best source of free bits is the local office equipment suppliers as they scrap large numbers of keyboards, printers, typewriters etc. because the ironmongery is shot but the electronics are usually in good order and of the right age for our use. I even use reclaimed chips, unsoldering them with a suck tool and metal heat sink. Disc drives from this source however are normally worn out and beyond repair due to head wear and/or misalignment.

PEB Power Supply

These recent repairs had brought to light the fact that the 12v supply in the PEB is only just sufficient for two older type drives. I seem to remember reading that the regulator was a large T03 can type but my PEB and the manual both agree that it is a 7812 plastic reg. Is this yet another production change by TI? The PEB transformer should be capable of more than 1 amp max so I will double up on this regulator next time the PEB is opened. The 12v rail sinks to 11.66v when both drives are enabled as when disc copying. The 5v rail maintains 4.87v which is plenty, no need to wear out chips too soon! This is no doubt why at the AGM Trevor had found powering up the 65Meg hard drive impossible with the PEB power supply. However we are fortunate that the design of the PEB is so arranged that all cards have their own regulators for both 5v and 12v. Any power failures on cards are isolated to that card only, more of this later!

We have a Silver Reed electric typewriter at work and as the interfaces to use it as a printer are rapidly dissapearing I decided to buy one. Unfortunatly only the parallel version was available so if anyone will

swap for a serial RS232 version they can have my brand new one. Otherwise a parallel interface for the 99/4A will have to be built. Does anyone know what signals are used? The Silver Reed instructions refer to "data strobe", "prime", "ack", "busy", and "select".

Presumably the DSR is already in my RS232 eprom so once I can decode the correct addresses and signals it should work. The dense black print out should be better and cheaper than using this Brother EP44 in its overstike mode as it overstikes no less than four times with TI-Writer and uses up film ribbons at an alarming rate.

Trevor's phone call at the end of July has resolved this particular problem; he is sending me a proper RS232 card, what a nice person he is!

In the months following the disc drive problems, my feelings towards these old BASF items became more and more distrustful. They are noisy, large, old and I would suspect, heavy on the discs. The local electronics shop came to the rescue just at the time when I had some cash. Alan Bray in Reddish had a chip problem with his disc controller card which we managed to fix, and whilst setting up his drives again, mentioned that this shop had some new double sided 40 track drives. When I called he had 3 new Panasonics left, I bought the lot on impulse for 40 pounds. This was probably the best purchase I have made for some time, they are quiet, smaller and use practically no power from the PEB compaired with the old ones.

Whilst in the spending mood, I also bought a spare Western Digital chip for the disc controller card as they are no longer in production and have no equivalent. Should any one want one I can get them for 4 pounds 50p which I considered to be a cheap insurance against losing the use of my system. The rest of the chips on this card are common with the exeption of the PAL12 but I believe from Trevor that Gary Smith can program new PALs.

Realising that now I had twice the disc capacity the decision of whether to change the existing format to all double sided or to leave the software on singe sided had to be made. The decision was made for me by finding a source of good quality double sided discs at a brilliant price. 500 were purchased, more than I will ever need. So, if anyone wants any discs, Fuji MD2D, at £2.00 for a box of 10, I will gladly post them off and donate 25p a time to TI*MES funds. Fuller details are on the MOBB BBS.

Thoughts on the BBS

And so to the BBS. I tried this out the last couple of weekends (22 29 July) and found it great to use, logging on for 18 mins and printing out the log after to find my way around the menus. Sysop Trevor was great, breaking in when I was having problems uploading and as he put it 'leading by the hand' without being at all intrusive, many thank Trevor for the help and also to all the members who have spent so much time and effort to get the board up and running. To those who have yet to try it, all I can say is go for it. This could be the best way of bringing all the Group together. For those without the necessary hardware the Group are arranging a loan system. It will be interresting to see how many new members or just other users that the BBS will attract. Shame that the hollidays forced a shut down, but I suppose Trevor couldn't be expected to carry all that gear away! Bet he misses it whilst away though!

The Curse of MOBB

After using the BBS, next day my PEB died. That lump in the throat feeling again! The console worked OK both alone and connected but SIZE showed no PEB expansion present. Trying not to panic and be logical, (THAT word AGAIN!) I first checked that the unregulated supplies from PEB power supply were OK, yes, so apart came the firehose plug. In there is only one data buffer chip and 5v regulator. Only 1v on 5v line and no shorts meant the 78M05 reg was duff. Quick replacement with available (second hand of course,) 7805 and up and running again.

This regulator now runs much cooler than before, convincing me that all new uprated regs should be fitted throughout, 78S series will give cooler running being rated at 2 Amps. BBS tip, at present you must type in messages in the For Sale and Wanted Sub Boards, so upload your message from disc into the TI*MES area of the Message Base and leave just a note refering in the Sub Board areas. Or dump your message into the Sub Board area using pre-written Macros chained together from Telco. It all saves on line time.

I seem to be having difficulty using ANSI terminal option in Telco, perhaps my file is duff as it locks my keyboard out without locking up the computer, so I have yet to see all the graphics and menu boxes.

Anyone for an Adventure?

Who is playing adventures? There must be someone who knows where that cursed alarm clock is in Scott Adams' Return To Pirates' Isle. Please tell me, as for ten years this has driven me mad. I have tried hacking into the code but my assembler experience is too limited. Most of the other common adventures are logical but the frustration of this one is discouraging me from trying any of the later difficult ones.

TI Writer Problems

As I use TI-Writer more I find the window system of text display less difficult but it is a poor substitute for the 90 column screen I am used to. Also, when inserting characters at the end of words, pressing reformat wipes out the space at the end of the word. Is there a way of avoiding this infuriating bug or is it supposed to? The manual seems to make no mention of this but there again it fails to mention a lot of other things too. And no-matter what I try, headers (.HE text) refuse to work. Perhaps one day someone who uses TI-Writer frequently will tell us occasional users of all the quirks so that we can correct our versions.

As this article will already take about 20 minutes to print at four overstrikes perhaps this is enough for this time, must leave space in TI*MES for everyone else! Better idea, i'll send the disc to Richard instead and then blame him for the wrong format and typos! *[gee, thanks - ed]* Only joking. So happy soldering till next issue.

The Art Of Assembly Part 7

This month's installment is not for the faint of heart. It will be heavy seas, high winds, rough waters. The subject is loaders. A loader is a program whose primary job is to load another program. Much of this article will be difficult to understand, and you may feel a little like Chico Marx in the movie The Cocoanuts, when he keeps asking Groucho "Why a duck?". Groucho was of course speaking of a Viaduct.

Putting first things first, we should answer the question "Why a Loader?" There are two answers to that question. One is speed, the other is memory allocation. One can, for example, write a loader that performs certain "once only" chores, then is mostly replaced by the program it loads, thus freeing up memory space for that program to use. The example we'll use today is taken from our Word Processor. In the sidebar is the source code for a loader we use so that the program may be used with the E/A module or the TI Writer module. The file it creates is an Option 5 type program file called UTIL1. That is the default filename that both E/A Option 5 and TIW Option 3 will look for on DSK1.

We said the going would get rough, and here's the opening blast of the hurricane. This source code actually makes two programs in one. The object file containing both programs is loaded into memory under E/A Option 3, along with TI's SAVE utility. We enter the program by typing GETUT as a Program Name, thus entering the first of the two programs at that label. This little program gets a memory image file containing the Extended Basic utilities and stashes that code within the memory space occupied by the second program, then exits to TI's SAVE utility, so we can save the second program to disk as UTIL1. "Why a Duck?", you ask.

Well, over here is the Viaduct. Just kidding! The reason is simple. The Word Processor was first developed as an Option 3 E/A program, mainly for our own use. When we decided to market it as a commercial product, we adapted it to run under Extended Basic with a custom loader submerged under an XB LOAD program. This meant that all the utilities such as VMBW, KSCAN, and so on were handled as Equates to the locations of these vectors when XB is in place. Much later, we decided, as part of a general upgrade, to add the ability to load from either E/A or TIW. Still, all the bulk of the program relied on finding utilities where XB places them. Thus we had to give our UTIL1 loader the capability of putting all those utilities in the right place before turning the computer over to the Word Processing program.

We then wrote a small Assembly program which would run under XB and capture the XB utilites for us in a memory image file called XBUT. Having that gave us the means to use the source code shown in the sidebar along with TI's SAVE utility to create our own custom Option 5 loader.

And that is part of the reason for a custom loader. Back when we decided to make the program operate from Extended Basic, we needed a loader so that we would not have to load the object file with XB's CALL LOAD. The Word Processor proper fills nearly all of both the Low and High portions of the 32K memory. The object file (uncompressed) fills 394 sectors of disk space, and cannot be loaded by the E/A Option 3 loader because it AORGs into space used by that loader itself. Loading that object code under Extended Basic takes all of seven

minutes. That's a long time to look at a screen which says "LOADING MAIN PROGRAM" and "PLEASE STAND BY". Thus we included in the WP code a custom "save" utility, so that the WP program could save itself as five separate memory image files. Then we wrote a loader which would be embedded under an XB LOAD program, and would load in these five memory image files from the disk. That way, from selecting XB to the Main Menu of WP on screen takes about 25 seconds from a floppy drive, or about four seconds from RAMDISK.

Much of the code in today's source is derived from that original Assembly loader we wrote for submerging under the XB LOAD program. It's not the prettiest code we've shown. In fact it's probably the ugliest we'll ever show you. We have violated many of our own rules in throwing this together. For example, there are data sections mixed in among the code sections. Also, there are places where we could have used our own methods to save memory space, but haven't done so. Instead, we followed the maxim "First, get it to work".

When we looked at the source code to prepare it for this article, we found "dead code" sections, subroutines that were never called, and unused data lines in it. Our only excuse is that this was mostly borrowed from the loader written for the XB version, hastily thrown together for a show deadline, and we quit looking at it once we got it to work. The version shown here will still work, but has been cleaned up considerably, and of course has been annotated so you can follow what it's doing line by line.

Before we look at the source code in detail, let's look at the Memory mapping for the WP program. (See Figure 1) In Low memory we have the XB utilities, then the section of code which starts up the program and puts the menu on the screen. Starting at >2A66 we have the code used for printing documents, plus the part that finds and reads a user's configuration file if he's configured his copy. That all ends at >39FE.

In High memory we have three arbitrarily divided sections of the code, which ends at >F1C0.

These five sections of code are stored on the disk in five Memory Image files, named as shown in Figure 1. The part called PRINTCODE could have been combined with the MENUCODE in one file, except that the section PRINTCODE is on occasion overwritten by either utility programs loaded as overlays, or by text from a Move or Copy text operation. Thus there are times during the program's operation that it must re-load PRINTCODE to print a document.

As you can see, the Loader sits at the higher addresses in High memory. At present, there's a gap between the end of the main program's code and the loader's beginning. If we wanted to, we could have the last section of code in the main program overlap all but a small portion of the loader without any harm. As it is, we have no plans presently for using up that remaining space.

Now into the source code. The REFs at the beginning refer to the E/A utilities which are available to us when the object code is loaded under Option 3. The code between label GETUT and the line reading B SAVE is all that gets executed after the Option 3 loading. This code establishes a Peripheral Access Block using the data at label SAVDT, then uses DSRLNK to bring the memory image file

XBUT into a VDP Ram buffer area. It then performs a VMBR operation to place that file's contents at label DATALD, within the part of the code that will be saved as UTIL1.

The TI SAVE utility takes everything between label SFIRST and SLAST and stores that as an Option 5 program file, which we name UTIL1. Thus the section at GETUT allows us to embed the Extended Basic utilities within that Option 5 program file before it's saved to disk.

We hope that's all clear, because if it isn't, then what follows will be very muddy indeed.

We now plunge into the murky waters of how the actual loader, saved as UTIL1, works. The very first thing it does is stash R11, which probably could be dispensed with, but it's there. Next it loads a temporary workspace, rather than our usual >20BA. The reason for this is simple. We will be writing to the area in low memory that includes >20BA, using registers as pointers, and we can't overwrite the workspace we're using. Thus we have a temporary workspace at label WS16 within the UTIL1 program's space.

The program now moves the 1262 bytes containing the XB utilities from label DATALD to their proper place in low memory, using the loop at label PUTUT. From this point on, the program will use the XB equates for its utility vectors such as VMBW. We've given these equates different names so they won't conflict with the REFs used in the GETUT section.

At label OPEN, we perform what's called "Boot Tracking". This section of code finds out what disk drive the UTIL1 program was loaded from, and passesthat information into the PAB data that it will use to load the main program's five memory image files. Thus if one has the Word Processor disk in drive 2, or any other drive including Ramdisk, the UTIL1 program will go to that same drive to find its files and load them. This program, incidentally, has not been made compatible with hard disk systems, but will load and run from any floppy or Ramdisk, regardless of what its drive number or letter is.

After that, we branch to label MENU, where we load the final workspace at >20BA. Next we perform a little operation that's only needed when we've entered with the TI-Writer module. We simply capture the character definition for the space character, then use that to define the zero character [CHR$(0)] to look like a space. The main WP program will load its own character set beginning at character one [CHR$(1)], and extending through character 144, but the TIW module would leave the zero character defined, and we want it to look like a space.

Now we do some VWTR operations to set the screen to text mode, set up the colors for text mode, and to insure that VDP will look for character definitions at >800, then clear the screen. Now we give the user two messages on the screen, and get on with the real work of loading the five memory image files.

This would get repetitious, so we'll just go through the process of loading the first such file, called MENUCODE. First, we set R0 to point to the PAB area in VDP, R1 to point to the data, R2 to the length of that data, then write the PAB into VDP Ram. The PAB contains the opcode 05, which is used to load a memory image file. Next in the PAB is the VDP buffer

address into which the bytes from that file will be dumped by the DSR. The third word in the PAB is always 0 for this kind of file, and the fourth indicates the maximum number of bytes to be taken from the file. This number must be equal to or greater than the actual file content. In our case, we've made it exactly equal to the number of bytes found in MENUCODE. Finally, there's a byte set to zero, then the length of the file descriptor, followed by the file descriptor text 'DSKx.MENUCODE'. The x is there to indicate that the 1 of DSK1 will have been replaced by the boot tracking process.

Once we've moved the PAB address plus nine into >8356 and cleared the STATUS, we proceed to get the file into the VDP Ram buffer by a BLWP DSRLNJ. DSRLNJ? Why not DSRLNK, you ask? The utility vector DSRLNK has been overwritten when we moved the XB utilities into Low memory, so we can't use it. Instead, we've included a DSR Linkage (thanks to Doug Warren/Craig Miller) which has been renamed DSRLNJ so it won't upset the Assembler. This is the same DSRLNK that was included in Barry Traver's column some time ago. That link vector and its associated code is part of our program UTIL1, which is kept in the highest available addresses in High memory.

When our main program is running, it too uses this linkage vector to perform file accesses. It also uses the GPLLNK included in our UTIL1 program, just before DSRLNJ.

Okay, so now we've got the section called MENUCODE in VDP Ram. The next step is to put that into Low memory at the correct address. That address is >24F4+128, or >2574. The 128 bytes between >24F4 and >2574 are used when we enter from XB to stash some system data which XB will need when we exit the program, hence the actual content of our WP program starts at >2574. Moving of the code into its proper place is done by a BLWP VMBRA, which is of course one of those XB utilities we put in place earlier.

Ths process continues until we've loaded all five memory image files. These files are true memory image, with no file headers attached to them. Many authors will go to some trouble about making file headers, but our reasoning was that, since the only possible use these files have is to be loaded by our own loader, there was no point in providing them with headers. There are days when we regret that decision, especially when we make changes in the main program which require changing the addresses equated in our loader, which then must be re-assembled just because one of those addresses changed by two bytes. The very next time we write a WP program, we'll put a file header of some kind in, giving us some information about the length of the file's content, and then we won't need to change and assemble the loader each time we change the main program.

The last couple of things this program does is to place the last section of code (WORDCODE3) into high memory, send the drive designator byte to location >FD0E+13, then branch to address >2840, the entry point of the main progam. Location >FD0E+13 is meaningless in this program per se, but it happens to be the location where the XB loader has the drive designator byte, so we park that byte at that location, where the main program can get the information about what drive the program disk resides in.

This loader is fairly messy, and we really should get around to cleaning it up, but the idea in today's article was to give you some ideas to play with, and to show some essential things a loader must do. We understand that there are some "General Purpose" loaders available in the TI community, but our experience has been that, since our own style of doing things is unique, we're stuck with writing our own loaders. After doing a couple of them, one can always take an old one, make some changes to the source code, and generate a new custom loader with very little effort. We hope you now know "Why a Duck".

Our next article will go into the topic of file accesses in more depth, with emphasis on trapping errors in file operations.

```
* Source code for util1 loader
* to load harrison's wp under editor/assembler option 5
* this is actually two programs in one
* the first gets and stows the xb utilities within the second,
* then branches to ti's save utility
*
        AORG  >F690             set memory location for this code
        DEF   GETUT             define entry point for first program
        REF   SAVE              refernce the label save in the ti save utility
        REF   VMBW,VMBR,VSBW,VSBR,DSRLNK
GETUT   MOV   R11,>8300         stash register 11 at >8300
        LWPI  WS16              load a workspace within our own code
        LI    R0,PAB1           set pointer for peripheral access block
        LI    R1,SAVDT          point to the data for that pab
        LI    R2,19             nineteen bytes in the pab data
        BLWP  VMBW              write 19 bytes to vdp ram
        AI    R0,9              add 9 to r0
        MOV   R0,PABPNT         place that number at >8356
        CLR   STATUS            clear the gpl status byte
        BLWP  DSRLNK            get the xb utilities into vdp buffer from disk file
        DATA  8                 data for dsr linkage
        LI    R0,>1020          point to buffer space in vdp ram
        LI    R1,DATALD         point r1 to location within program to be saved
        MOV   SAVDT+6,R2        get length of file into r2
        BLWP  VMBR              read the xb utilities into memory
        B     SAVE              branch to the ti save utility
SAVDT   DATA  >0500,>1020,0,>24F6->2008,>0009
        TEXT  'DSK4.XBUT'

* End of first program

* start of second program
* the part from here to the end is saved by the ti save utility as file util1
* this part is what loads the five memory image files comprising the wp program
```

```
        DEF   SFIRST,SLAST,SLOAD    defined labels required by ti's save utility
        MOV   R11,>8300             stash r11
        LWPI  WS16                  load temporary workspace
        LI    R9,DATALD             point at data from saved xb utilities
        LI    R10,>2008             point at start of xb utility vector area
        LI    R4,>24F6->2008        set r4 for number of bytes in xbut
PUTUT   MOV   *R9+,*R10+            move a word into low memory area, increment
                                    pointers
        DECT  R4                    decrement count by two, since we're moving a
                                    word
        JNE   PUTUT                 if not zero, repeat
        B     OPEN                  branch to next section of code
WS16    BSS   32                    temporary workspace
DATALD  BSS   >24F6->2008           storage area for xb utilities
FSTEND  EQU   >2A66                 end of first section of main program
ENDCNF  EQU   >39FE                 end of configuration setting code
DEFPRN  EQU   >F0F8+200             end of high memory part of wp
FAC     EQU   >834A                 floating point accumulator
WS      EQU   >20BA                 real workspace
VSBRA   EQU   >2028                 the xb vsbr vector's address
OPEN

* The section here at label open performs "boot tracking"
* that is, it tells our program which drive it was loaded from

        MOV   >83D0,R12             get the cru base in r12
        MOV   >83D2,R9              get the rom address for device
        LDCR  ONES,0                enable the rom
        AI    R9,4                  adding four puts us at the length byte
        MOVB  *R9+,R4               place that in r4 and increment r9
        SRL   R4,8                  right justify length in r4
        LI    R10,SAVDT3+10         point to text buffer
MOVIT   MOVB  *R9+,*R10+            mov one byte from rom to text buffer
        DEC   R4                    finished?
        JNE   MOVIT                 no, do another byte
        LDCR  R4,0                  disable the rom (r4 is zero at this point)
        MOVB  SAVDT3+13,R1          move drive number (or letter) into r1
        MOVB  R1,MENUDT+13          then move into the pab data lines
        B     MENU                  branch to next section of code
ONES    DATA  >0101                 word to turn on rom in cru

*

VMBWA   EQU   >2024                 xb's vmbw vector location
VMBRA   EQU   >202C                 xb's vmbr location
```

```
VSBWA   EQU  >2020              xb's vsbwa
VWTR    EQU  >2030              xb's vwtr location
KEYADR  EQU  >8374              key-unit address
STATUS  EQU  >837C              gpl status byte
SCRMO   EQU  >83D4              storage location for screen mode byte
PABPNT  EQU  >8356              pointer location for dsr linkage
PAB1    EQU  >400               first pab address
TEXMO   BYTE >F0                text mode byte
BLNKLN  TEXT '                                                '
OPMSG   TEXT 'LOADING IN MAIN PROGRAM' text message
PSBMSG  TEXT 'PLEASE STAND BY'
CRITE   BSS  8
```

** Following data section contains the pab data for the sections of main program*

```
MENUDT  DATA >0500,>1000,0,FSTEND->2574,>000D
        TEXT 'DSK1.MENUCODE'
SAVDT3  DATA >0500,>1000,0,ENDCNF-FSTEND,>000E
        TEXT 'DSK1.PRINTCODE'
WRD1DT  DATA >0500,>0D00,0,9983,>000E
        TEXT 'DSK1.WORDCODE1'
WRD2DT  DATA >0500,>0D00,0,9983,>000E
        TEXT 'DSK1.WORDCODE2'
WRD3DT  DATA >0500,>0D00,0,DEFPRN+2->A000-9983-9983,>000E
        TEXT 'DSK1.WORDCODE3'
```

** main part of loader begins here*

```
MENU    LWPI WS                 sets up workspace
        MOVB TEXMO,SCRMO        move byte >f0 into >83d4
        CLR  KEYADR             clear word at >8374
```

** The next six lines are here to clear out the definition of character zero.*
** that character is defined when we enter from ti-writer module, so we set it*
** up to look like a space character.*
** this is necessary since our wp makes use of character zero, and we want it to*
** look like a space on the screen*

```
        LI   R0,32*8+>800       set r0 to space character definition
        LI   R1,CRITE           use a storage space
        LI   R2,8               eight bytes to get
        BLWP VMBRA              read eight bytes
        LI   R0,>800            point to character zero definition
        BLWP VMBWA              write eight bytes
TEXT    LI   R0,>01F0           prepares for text mode
        BLWP VWTR               sets screen in text mode
        LI   R0,>074E           sets colors
```

```
       BLWP  VWTR          for text mode
       LI    R0,>0401      prep to set charcter table at >800
       BLWP  VWTR          set it there
CLS    CLR   R0            point r0 to screen origin
       LI    R4,24         24 rows to clear
       LI    R1,BLNKLN     point to 40 spaces text
       LI    R2,40         40 characters per row to write
LOOP   BLWP  VMBWA         write 40 spaces
       A     R2,R0         move write address 1 line (add 40 to r0)
       DEC   R4            decrease count of rows
       JNE   LOOP          if not zero,loop back and do another
       LI    R0,9+9        set r0 for row 10, column 10
       LI    R2,23         23 characters in message
       LI    R1,OPMSG      point r1 at message
       BLWP  VMBWA         write "loading in main program" to screen
       LI    R0,11+12      set r0 for row 12, column 13
       LI    R2,15         15 bytes in message
       LI    R1,PSBMSG     write "please stand by"
       BLWP  VMBWA         to the screen
       LI    R0,PAB1       point r0 to peripheral access block vdp address
       LI    R1,MENUDT     point to first pab data block
       LI    R2,23         23 characters in block
       BLWP  VMBWA         write pab to vdp ram
       AI    R0,9          add nine to address
       MOV   R0,PABPNT     move that value to >8356
       CLR   STATUS        clear status byte
       BLWP  DSRLNJ        use dsr linkage vector
       DATA  8             data for dsr link
       LI    R0,>1000      point to buffer area
       MOV   MENUDT+6,R2   get file length into r2
       LI    R1,>24F4+128  point at low memory location for first code section
       BLWP  VMBRA         read the section menucode into low memory
       LI    R0,PAB1       point to pab location
       LI    R1,SAVDT3     savdt3 is second pab data portion
       LI    R2,24         24 bytes to write
       BLWP  VMBWA         write pab into vdp
       AI    R0,9          add nine to address
       MOV   R0,PABPNT     move that to >8356
       CLR   STATUS        clear gpl status byte
       BLWP  DSRLNJ        use dsr linkage vector
       DATA  8             required data
       LI    R0,>1000      point to buffer
       LI    R1,FSTEND     point to end of first code section
```

```
MOV   SAVDT3+6,R2        length of code section in r2
BLWP  VMBRA              move the file printcode into low memory
LI    R1,WRD1DT          point to next pab data
LI    R0,PAB1            set r0 to pab
LI    R2,24              24 bytes to write
BLWP  VMBWA              write data to pab
AI    R0,9               add NINE
MOV   R0,PABPNT          move to >8356
CLR   STATUS             clr status
BLWP  DSRLNJ             dsr link
DATA  8                  req'd data
LI    R0,>0D00           set to buffer location
LI    R1,>A000           point r1 to start of high memory
LI    R2,9983            9983 bytes in file
BLWP  VMBRA              read this section into high memory
LI    R0,PAB1            reset to pab
LI    R1,WRD2DT          second high memory part
LI    R2,24              24 bytes
BLWP  VMBWA              write pab
AI    R0,9               add 9
MOV   R0,PABPNT          to >8356
CLR   STATUS             clr status
BLWP  DSRLNJ             dsr link
DATA  8                  data
LI    R0,>0D00           point to buffer
LI    R1,>A000+9983      address for second section of high memory code
LI    R2,9983            9983 bytes to read
BLWP  VMBRA              read into high memory
LI    R0,PAB1            set to pab
LI    R1,WRD3DT          third high memory part
LI    R2,24              24 bytes pab data
BLWP  VMBWA              write to vdp
AI    R0,9               add 9
MOV   R0,PABPNT          to >8356
CLR   STATUS             clr status
BLWP  DSRLNJ             dsr link
DATA  8                  data
LI    R0,>0D00           set to buffer
LI    R1,>A000+9983+9983 address for last section of code
MOV   WRD3DT+6,R2        length of code section
BLWP  VMBRA              read code into high memory
MOVB  SAVDT3+13,>FD0E+13 "mailbox" the drive designator for main
                           pgm
B     >2840              branch to main program entry point
```

```
* General purpose gpl and dsr links
* for use under extended basic
* this code by doug warren/craig miller of miller's graphics

GPLWS   EQU  >83E0
GR4     EQU  GPLWS+8
GR6     EQU  GPLWS+12
STKPNT  EQU  >8373
LDGADD  EQU  >60
XTAB27  EQU  >200E
GETSTK  EQU  >166C
        AORG >FF2C              set memory location for linkage routines

* This aorg is set so that utilities wind up at the same location as with the
* extended basic loader

GPLLNK  DATA GLNKWS
        DATA GLINK1
RTNAD   DATA XMLRTN
GXMLAD  DATA >176C
        DATA >50
GLNKWS  EQU  $->18
        BSS  >08
GLINK1  MOV  *R11,GR4
        MOV  *R14+,GR6
        MOV  XTAB27,R12
        MOV  R9,XTAB27
        LWPI GPLWS
        BL   *R4
        MOV  GXMLAD,>8302(R4)
        INCT STKPNT
        B    LDGADD
XMLRTN  MOV  GETSTK,R4
        BL   *R4
        LWPI GLNKWS
        MOV  R12,XTAB27
        RTWP
PUTSTK  EQU  >50
TYPE    EQU  >836D
NAMLEN  EQU  >8356
VWA     EQU  >8C02
VRD     EQU  >8800
GR4LB   EQU  >83E9
GSTAT   EQU  >837C
DSRLNJ  DATA DSRWS,DLINK1
DSRWS   EQU  $
DR3LB   EQU  $+7
DLINK1  MOV  R12,R12
        JNE  DLINK3
```

```
        LWPI GPLWS
        MOV  PUTSTK,R4
        BL   *R4
        LI   R4,>11
        MOV  R4,>402(R13)
        JMP  DLINK2
        DATA 0
        DATA 0,0,0
DLINK2  MOVB GR4LB,>402(R13)
        MOV  GETSTK,R5
        MOVB *R13,DSRAD1
        INCT DSRADD
        BL   *R5
        LWPI DSRWS
        LI   R12,>2000
DLINK3  INC  R14
        MOVB *R14,TYPE
        MOV  NAMLEN,R3
        AI   R3,-8
        BLWP GPLLNK
DSRADD  BYTE >03
DSRAD1  BYTE 00
        MOVB DR3LB,VWA
        MOVB R3,VWA
        SZCB R12,R15
        MOVB VRD,R3
        SRL  R3,5
        MOVB R3,*R13
        JNE  SETEQ
        COC  GSTAT,R12
        JNE  DSREND
SETEQ   SOCB R12,R15
DSREND  RTWP
SLAST   END
```

** Slast marks the end of what's saved in the file util1*

Kiwi Korner

Firstly thankyou Stephen for your comments (p17 ish 49) it is certainly hard to belieive that it is only 12 years since I eagerly sent my first order to you (care of Stainless Software) for some cassette based Exented BASIC games software. Great stuff!

Yes, I am currently residing in New Zealand but my trusty TI99/4a is still somewhere on the high seas along with most of our worldly belongings! I do however plan to remain a member of the UK Ti User Group despite living (for the time being at least) thousands of miles away from the UK and anyway, I doubt there is a New Zealand TI user group - please write in if you know otherwise.

It would be sad to see the group go under through dwindling support so in an effort to boost membership I sent information to Computer Shopper earlier this year for inclusion in their user group page (yes - confession time - it was me!) Unfortaunately, it took several months to be included and in the meantime Alasdair Bryce resigned his post as membership secreatary - so apologies to Alasdair if he was innundated with mail.

I wonder if it actually increased membership? Perhaps the new membership secretary should submit a more detailed overview of the group to Computer Shopper?

Apologies also to anyone who tried register with Edward Schwartz for TI Emulator for the PC after reading my article in issue 48. Stephen Shaw's update (page 17 ish 49) suggests that Edward has had to withdraw support. I hope this is only temporary and I am currently trying to find out further details from Edward about TI Emulator's long-term prospects.

I am pleased to see the enthusiasm with which our new editor seems to have taken on his new post.

The magazine has been given a new lease of life (and that's no criticism of the previous editor, who did an excellent job) and I particularly hpoe the inclusion of an editorial and a 'back page' will remain a regular feature - it kind of gets the whole thing together.

Well, I'd better sing off for now - must tend to the sheep...

FCTN - QUIT

Why Does It Work?

A mathematical friend of mine sent me an iterative procedure for calculating a well known number. He had from a mutual friend who, I understand, came across a description, but not the original reference, in a recent book. The thing is: none of us can see why it works. A remarkable feature is that, if you can upgrade the precision of the machine's arithmetic to match, each iteration roughly doubles the number of significant figures in the estimate. Here is the procedure using standard TI arithmetic and print out.

```
100 A=1
110 X=1
120 B=1/SQR(2)
130 C=1/4
140 V=0
150 VR=0
160 Y=A
170 A=(A+B)/2
180 B=SQR(B*Y)
190 C=C-X*(A-Y)^2
200 X=2*X
210 V=(A+B)^2/(4*C)
220 IF V=VR THEN 270
230 VR=V
240 PRINT
250 PRINT "A,B,C,X,Y,V";A:B:C
:X:Y:V
260 GOTO 160
270 END
```

The TI actually holds the digits 3 5 9 3 for the ninth to twelfth decimal places but prints only to the ninth after rounding up. Only the tenth place is correct.

The method was used by some Japanese to develop our number to sixteen million decimal places, requires some tens of iterations. A fast conventional series could require about seven hundred thousand terms. In either case, there must be massive practical problems in controlling the work. While I do not intend to experience too much of them myself, I am sure that some of us will be interested in an article on general tactics from a member wise in the matter of high precision calculations.

As to the underlying theory, the word is that it has something to do with Gauss' work on arithmetic and geometric means of two numbers. Now, we all know that Karl Friedrich G had a finger in a great many pies but I don't of this work and have been unable to find a reference. Can anyone help?

When I read about going to sixteen million places my reaction was 'whatever for?' Since 1882, the number has been known to be transcendental, which fact rules out it having a recurring decimal part of however long a period. So, finding one can't be the object of the exercise. On the other hand, there are some curious patterns even as early as the stretch from the sixth to the thirty first decimal places:

265358979 32 38 49264338 32 795

Is someone on the track of a little surprice the Nature has concealed from us so far?

Music SDA

Another Official TI Never Released Module

described by Charles Good

Lima Ohio User Group

This is the MUSIC MAKER module with an extra Grom. This extra grom turns MUSIC SDA into something really useful, not just the toy that MUSIC MAKER is. With MUSIC SDA you can create music disk files directly usable in BASIC, GPL, and ASSEMBLY programs.

The title screen says "copyright 1980 Texas Instruments" and looks identical to the MUSIC MAKER title screen except that it says "MUSIC SDA" instead of MUSIC MAKER. I have no idea what the "SDA" means. First you load some music into the module from disk or tape, or you create some music from within the module in the same way as is done with MUSIC MAKER. You are then presented with the following list of options, two of which are not found in MUSIC MAKER:

EDIT

PLAY

SAVE (creates a cassette file or a 59 sector disk file irrespective of how many measures long the music actually is)

PRINT (only works with the TP, prints music on the staff)

DUMP (not in MUSIC MAKER)

EXECUTE (not in MUSIC MAKER).

I am not really sure about the purpose of EXECUTE. You are given options to change speed, start and stop measure, number of voices, etc., just as when you select PLAY. The computer then grinds away internally for a while and then plays the music. You can speed up the pace of the music that is played tremendously with EXECUTE, much more so than with PLAY. I think you are supposed to EXECUTE if you make any changes before you DUMP the music.

DUMP is the really neat feature of MUSIC SDA. It saves music to disk in really useful formats, not the "can only be read into MUSIC MAKER" 59 sector format you get with SAVE. When DUMP is selected you get these options:

1. GPL

2. BASIC (DISPLAY FORMAT)

3. BASIC (MERGE FORMAT)

4. ASSEMBLER

DUMP GPL creates a DV80 file that can be used as source code for programming in GPL.

DUMP BASIC (DISPLAY) creates a DV80 file that looks like a TI BASIC program listing. You can type this list into TI BASIC or XB and play the music at the speed you designated from EXECUTE.

DUMP BASIC (MERGE) creates a DV163 file that you can MERGE into any XB program.

DUMP ASSEMBLER makes a DV80 file of the music that can be used in assembly language programming.

Thanks to Mike Wright for calling this little gem to my attention. I asked Mike what the "SDA" of MUSIC SDA stands for. He is obviously as knowledgable as I

am. "Hell if I know," he said.

This would have been a REALLY USEFUL command module if it had ever been released to the public. Its title screen date date suggests that it was in existence in the early stage. of the 99/4(A)'s history. Why was it never released? My guess is that TI didn't want the public to know what GPL source code looked like.

The following are samples of the exact same two measures saved to disk with DUMP. Each DUMP automatically generates comments indicating the title of the music, the start of each new measure, and the total length of the dumped music in bytes.

```
---  DUMP GPL ---
*       FUGHETTA
*       MEASURE:  0001
        DATA >03,#>8E0F,>90,>
0C
        DATA >02,#>8A0C,>04
        DATA >02,#>830E,>04
        DATA >02,#>8315,>04
        DATA >02,#>8709,>04
        DATA >02,#>8E0B,>04
*       MEASURE:  0002
        DATA >02,#>8A0C,>04
        DATA >02,#>8E0F,>04
        DATA >02,#>8F07,>10
        DATA >02,#>8708,>08
ENDSND  DATA >04,>9F,>BF,>DF,
>FF,>00
*       00047 BYTES

--- DUMP BASIC DISPLAY ---
20000 REM FUGHETTA
20010 REM MEASURE: 0001
20020 CALL SOUND(0200,00440,0
0)
20030 CALL SOUND(0067,00554,0
0)
20040 CALL SOUND(0067,00494,0
0)
20050 CALL SOUND(0067,00330,0
0)
20060 CALL SOUND(0067,00740,0
0)
20070 CALL SOUND(0067,00657,0
0)
20080 REM MEASURE: 0002
20090 CALL SOUND(0067,00554,0
0)
20100 CALL SOUND(0067,00440,0
0)
20110 CALL SOUND(0267,00880,0
0)
20120 CALL SOUND(0133,00831,0
0)
20130 REM 00427 BYTES

--- DUMP ASSEMBLER ---
*       FUGHETTA
*       MEASURE:  0001
        BYTE >03,>8E,>0F,>90,
>0C
        BYTE >02,>8A,>0C,>04
        BYTE >02,>83,>0E,>04
        BYTE >02,>83,>15,>04
        BYTE >02,>87,>09,>04
        BYTE >02,>8E,>0B,>04
*       MEASURE:  0002
        BYTE >02,>8A,>0C,>04
        BYTE >02,>8E,>0F,>04
        BYTE >02,>8F,>07,>10
        BYTE >02,>87,>08,>08
ENDSND  BYTE >04,>9F,>BF,>DF,
>FF,>00
*       00047 BYTES
```

Never Released TI Modules

Never Released Official TI Modules:

by Charles Good

Lima Ohio User Group

Wing War

The title screen says "Texas Instruments presents Wing War, copyright IMAGIC 1983" You see an elaborately detailed underground cavern with stalagtites, stalagmites, etc. A pair of birds descend from the ceiling, and the speech synthesizer says in a very realistic voice, "Adventures await dragon master." A dragon appears in the lower part of the screen that you control with the joystick. You make the dragon fly by flapping its wings, accompanied by very realistic flapping wing sounds. The dragon can spit out fireballs which will melt the rock walls of the cavern and allow access to goodies that you can see imbedded in the rock. There are holes in the top of the cavern, and the dragon can fly out these holes to the open sky. Various treasures and things can be found floating around in the sky.

On screen instructions give you the following information:

Collect crystals for power.
Obtain treasures from a cave.
Blast rocks in sky for gifts.

Wash crystals and treasures in the magic fountains (you find lots of these in your travals around the caverns) and then take them to your lair.

JS or keyboard.
To move press left or right.
To fly down press up.
To fire fireballs press down.
To fly up press firebutton.

Talk about confusing instructions! "To fly down press up." I never did really get the hang of controlling the dragon, mainly because the consfusing use of joystick movements. Why not, for example, press the firebutton to fire fireballs. Movement of the dragon responds realistically to the earth's gravity. If you press the fire button a couple of times (for up), the dragon will shoot up even after you let go of the fire button.

Its rate of upward movement will gradually slow and then it will start to sink unless you give it a few more jolts of "up" with the firebutton. When set in motion left/right, the dragon will continue to move after the joystick is released, but will gradually sink to the bottom of the cave unless some "up" force is also applied. These sorts of movements are just what one would expect from analysis of vector physics on a free moving body influenced by gravity.

According to the on screen instructions, scoring is as follows:

Purified crystals	10
Super crystals	100
Super super crystals	500
Eggs (game over)	512
Lives (game over)	1024
Mate	1000
Treasures	x256
Fireballs (game over)	1

Presumably the word "Mate" above is a noun rather than a verb. I have never found a mate in my bumbling around with this game. When one of the creepy things in the cave kills you and you lose a life, the computer says "Alas." When you are down to your last dragon (life), the computer says "No eggs left in lair,

master." When your last dragon bites the dust, the game ends with the computer saying "Our glory is now only a passing memory."

Mouse Attack:

The title screen reads "by Don Fitchhorn, copyright 1983 Sierra On-Line". This game is another PacMan look alike for one or two players. The players are "plumbers" who have to traverse every spot in the maze before moving on to the next screen. There are three mice that float around the maze. At specific times the plumber can catch mice for extra points. At most times the mice chase and try to kill the plumber.

At the beginning of the game, you are given the opportunity to change defaults, as follows:

1 or 2 players

Keyboard or Joystick. If you chose the keyboard, you are given the option of using the ESDX or the ESDF keys for movement.

Music YN

Sound effects YN
Character speed 1= slow 9= fast

The ability to set the speed of each of the two possible plumbers and each of the three mice individually is a nice feature. You can set up this game so that any klutz can get lots and lots of points. Just set your plumber speed for maximum and the mice speed for minimum. It is really hard to get killed this way, and the game can go on almost forever. I have seen my 9 year old spend several hours just piling up the points with Mouse Attack set up this way.

Sub Oceanic:

The title screen says "by Dominic J. Melfi, copyright 1982 by Texas Instruments Incorporated. Press enter or joystick fire to use keyboard or joystick."

You command a submarine and can move rapidly left/right and up/down under water, or surface. You fire vertical torpedos at an endless host of ships, planes of various types, and helicopters that are out to get you. Your torpedos pop right out of the water and up into the sky to hit the planes. The opposition drops depth charge clusters and rapidly falling vertical torpedos which you must dodge. After you have been hit a few times you can't dive down as far into the ocean, and eventually you must stay on the surface. This brings the game to a rapid end, because on or near the surface your reaction time to dodge the incoming torpedos is greatly reduced.

There is alot of fast action and eye/hand coordination in Sub Oceanic. It is your typical "shoot up the never ending hords of bad guys" kind of game. The 1982 copyright suprises me. Apparently this is not a module that was abandoned by TI when they left the home computer market. The 1982 date suggests that TI had previously decided not to market this module.

Paddle Ball:

"Copyright 1983 TI." Do you remember when TV games first became popular? Just before the original Atari game system was marketed the most popular TV game was PONG, which resembled ping pong. This is TI's version of PONG, with lots of possible variations. What I have is an EA5 file that will load and run out of a GramKracker in the usual way and which can also be loaded and run using an ordinary E/A module without a gram

device or supercart. After booting, if you just leave this module on the screen without pressing keys the module will eventually shift into a self demonstration mode and illustrate automatically all of the following options.

1. Single ball
2. With central fog area
3. Center field blockades
4. 2 balls
5. Double blockades
6. 3 balls
7. Bomb with blockades
8. 2 bombs
9. Hole paddle on screen 3
10. Hole paddle on screen 4

A center field blockade is a moving bar with a hole in it near the center of the screen. The bouncing ball may go through the blockade if it passes through the hole. Otherwise the ball hits the blockade and rapidly bounces back toward the paddle. A bomb is a ball that randomly turns from green to red. When it is green, you are supposed to hit it with the paddle as is normal. When the ball is red you must AVOID touching the ball with the paddle or you loose the turn. Bombs change color rapidly and randomly, making game play very interesting.

This is my personal favorite of this bunch of never released modules. The action is fast and there is lots of variety. I think this would have been a commercial success for TI.

Simon Says!

That's what the title screen says, complete with explamation mark. There is no copyright notice. This is your typical SIMON game in which you are asked to exactly repeat an ever increasing sequence of ESDX keypresses. Each keypress generates a different tone. With each successful try the computer adds one keypress to the end of the sequence. There are four levels of difficulty, but I can't tell what the difference in difficulty is. Each level seems to play the tones at the same speed. If you make a mistake you are given the opportunity to review the keypress/tone sequence before continuing the game or starting another game.

There is nothing fancy about SIMON SAYS!, but I like it anyway. It is as fast, as colorful, and as pleasing to the ear as any of the stand alone SIMON electronic games that were sold a few years ago.

Acknowledgements:

In ending I want to acknowledge the assistance of Mike Wright (TI Sig, Boston Computer Society), and Gary Cox (Mid South 99ers). These individuals provided me with much of the software described in these articles.

More on the pestiferous asterisk bug in TI-Writer. Dr. Guy-Stefan Romano has confirmed and explained it. If you are printing out of the Formatter mode and your text contains an asterisk followed by two or more numeric digits - the asterisk and two digits will disappear! For instance, A*256 becomes A6, and I've noticed that A6 in programs published in several newsletters recently.

The TI-Writer program misinterprets the asterisk and two digits as an instruction to input data from a "value file" (see Alternate Input on p. 111 of the manual).

The solution to this bug is to type two asterisks foll- owed by two dummy digits, then the actual digits.

For instance, instead of A*256 type A**25256.

Trouble is, the bug usually shows up in a program which has been LISTed to disk and then MERGEd into TI-Writer, and is usually not noticed. The solution? Run the pro- gram through my 28-Column Converter (see Tips #18!).

I would suggest that you also avoid the use of the & and @. The & will only underline a single word, unless you tie words together with the ^ sign. If you tie words together, the Fill and Adjust will leave gaping blanks in your lines and if you tie too many together the line will extend beyond the right margin!

Also, the underlining is a broken line. It is better to use the escape codes CTRL U, FCTN R, CTRL U, SHIFT -,CTRL U, SHIFT A, CTRL U, which will give a solid underline until you turn it off with CTRL U, FCTN R, CTRL U, SHIFT -, CTRL U, SHIFT @, CTRL U.

The @ is handy to emphasize a single word, but if you want to double-strike a whole sentence or paragraph it is better to use the escape code CTRL U, FCTN R, CTRL U, SHIFT G, and turn it off again with CTRL U, FCTN R, CTRL U, SHIFT H.

The period bug is another killer - the Formatter thinks that any line which begins with a period is a formatter command, and deletes the whole line! If your text contains a decimal value such as .11 and the wraparound puts it at the beginning of a line, the line disappears! There are two ways around this - put a 0 in front of all your decimals, as 0.11, or transliterate all your periods.

In all, the TI-Writer formatter is a temper- amental piece of software, prone to unwanted line feeds and unexpected paper- wasting form feeds. I like to use it to right-justify text back to the disk, but from then on I prefer to print it out of the editor mode, or out of my own program.

[Art Green has addressed these problems in his TI Writer Version 5.0 available from the disk library- sjs]

Micropendium ran a contest to improve on a brief ingenious organ program. The winner was Michael Christianson, who wrote a superb program. I didn't enter the contest, of course, and my version is not nearly as good, but have fun -

```
90 CALL CLEAR
95 PRINT TAB(5);"MICROPENDIU
M ORGAN": : : : : : : : :"Pl
ay bass with left hand": :"o
n left side of keyboard,": :
"melody on the right": : :
100 REM - MICROPENDIUM ORGAN
```

```
 modified by Jim Peterson
110 OPTION BASE 0
120 DIM NOTE(20)
130 FOR A=0 TO 20
140 READ NOTE(A)
150 NEXT A
160 DATA 40000,220,247,262,2
94,330,349,392,440,494,523,5
87,659,698,784,880,988,1047,
1175,1319,1397
170 CALL KEY(1,K1,S)
180 CALL KEY(2,K2,S)
190 CALL SOUND(-1000,NOTE(K2
+1),0,NOTE(K2+1)*1.01,5,NOTE
(K1+1)*3.75-ABS(K1+1=0)*1100
00,30,-4,0+ABS(K1+1=0)*30)
200 GOTO 170
```

A sprite routine that doesn't do anything but look pretty. I call it Patches.

```
50 CALL CLEAR :: CALL SCREEN
(5)
100 A$=RPT$("AA55",16):: B$=
RPT$("F",64):: CALL MAGNIFY(
4):: RANDOMIZE
110 FOR CH=40 TO 136 STEP 8
:: CALL CHAR(CH,A$,CH+4,B$):
: NEXT CH
120 C=2 :: S=40 :: R=1 :: FO
R T=1 TO 24 STEP 2 :: COL=15
0*RND+50 :: CALL SPRITE(#T,S
,C,R,COL,#T+1,S+4,C+1,R,COL)
:: S=S+8 :: C=C+1 :: R=R+15
:: NEXT T
140 FOR T=1 TO 50 :: CALL CO
LOR(#INT(24*RND+1),INT(16*RN
D+1)):: NEXT T :: GOTO 120
```

This is one that I fancied up, based on a sprite routine written by a youngster named Andrew Sorenson, published in the Sydney Newsdigest from Australia.

```
100 ! WILL O' WISP
      by Jim Peterson
       based on
       Andrew Sorensen's
       sprite routine
110 CALL CLEAR :: CALL SCREE
N(2):: CR=48
120 FOR CH=48 TO 63 :: FOR L
=1 TO 4 :: RANDOMIZE :: X=IN
T(16*RND+1)*2-1 :: X$=SEG$("
0018243C425A667E8199A5BDC3DB
E7FF",X,2):: B$=B$&X$ :: C$=
X$&C$ :: NEXT L :: CALL CHAR
(CH,B$&C$):: B$,C$="" :: NEX
T CH
130 FOR N=1 TO 28 :: CALL SP
RITE(#N,CR,INT(14*RND+3),8*N
+20,120,5,0):: NEXT N :: IF
CR=64 THEN CR=48 :: T=T+1+(T
=2)*2 :: CALL MAGNIFY(T)
140 X=(INT(3*RND)-1)*4 :: Y=
(INT(3*RND)-1)*4
150 IF INT(10*RND+10)<>10 TH
EN 170
160 CR=CR+1 :: GOTO 130
170 FOR N=1 TO 28 :: CALL MO
TION(#N,-Y*20,X*20):: NEXT N
 :: GOTO 140
```

Memory Full!

Jim Peterson

This page is blank