TI*MES

```
100 DIM COLOUR(4). RSPEED(4). CSPEED(4)
110 CALL CLEAR
120 CALL CHAR(96. "81423C247E5A4242")
130 CALL CHAR(100. "42425A7E243C4281")
140 FOR S=1 TO 4
150 COLOUR(S)=S+1
160 RSPEED(S)=S-2:: CSPE
170 CALL SPRITE (#S
    RSPEED(S
180 NEX
190 R
200 F
210 C                                   6.128.
220 I
    C
230
240 C
250
                                     S.RSPEED(S).
                                  LACE(S)
500 SUB CHANGE(S. H. V)
510 CALL PATTERN(#S. 100)
520 H =-H::V=-V
530 CALL MOTION(#S. H. V)
540 FOR DELAY=1 TO 200:: NEXT DELAY
550 SUBEND
600 SUB REPLACE(S)
610 CALL PATTERN(#S.96)
620 SUBEND
```

ISSUE NO.41 SUMMER 93

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

EDITOR COMMENTS

As you can see this issue has a slightly different format to make it more graphically appealing. Please let us know what you think. The next issue will be compiled by Gary SMITH so send him your feelings about the layout etc.

DISCLAIMER

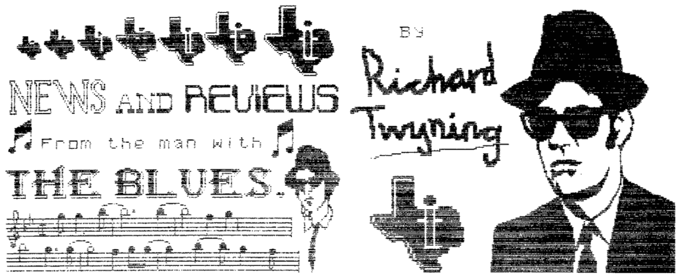
All views by contributors to this magazine are strictly their own, and do not represent those of the Commitee. Contrary opinions are very welcome and will if you request, be made available in print in the Consultation Zone. Any errors produced in this magazine will be corrected upon request.

NEXT COPY DATE

All copies by the 12th September 1993. Please ensure all copies as dark as possible please. All copies to fit on A4 paper with the following approximate maximum dimensions on the page.
Left and right margins 15mm. Top 15mm Bottom 20mm.
Thankyou all for yourhelp

CONTENTS OF THIS ISSUE

2022 note: The copy date for this issue was a week after my mothers death. I had the care of my blind insulin dependant diabetic father. As a result my output from now on was much reduced- not helped by the magazine appointing a new editor who did not wish to edit and admitted discarding what little I submitted without looking at it... S Shaw.



Dear CRAY1 owners everywhere,
The Bluesman is back with a big article again. The AGM was held in Derby on the 1st of May, and wasn't as well turned out as last year, but somehow we managed to achieve more. This was probably due to the attendance of Mr.Vice Chairman, Mark Wills. Mark brought his amazing RAMdisk utility program which will work on just a 4A with only floppies!!! The reason for this is that it loads up two disk directories **ENTIRELY** into memory. Mark differentiates between the two by calling one the RAM directory, and the other the floppy directory. You are asked to type the full path name though, so instead of typing DSK3 for Myarc RAMdisk, you could just enter DSK2. and use two floppies. It's directory organization coding is the best bit though. You can have a marked file on both logged directories, and then view both of the files by saying whether you want the file that is marked on the RAM directory, or the floppy directory!

He's also got little selection windows that come up over the top of everything else, and I thought he'd done it in machine code, but he just said he doesn't save the data, but just reprints the old screen again. It really appears as though he's saving the data and then restoring it, but he says he isn't, and it's all written in **XB**.

That's enough waffle for a bit. First thing's first, and here's the minutes of the **Annual General Meeting**:

There were two apologies that were read by **John Murphy** of **DORTIG** These were from **Trevor Taberner** and **Terence J. Leach** who were unable to attend the meeting.

ITEM 1: Officers reports:

Mr.Chairman (Trevor Stevens): Trevor has had to take over the job of editing TI*MES for two quarters, due to Alan Bailey's unfortunate illness. I'm sure that the thoughts of the entire T.I. community go out to him.

Mr.Secretary (Richard Twyning): I have had a couple of enquiries about the group, but cannot confirm that they have joined until I obtain a copy of the full group database.

We did have a new member at the AGM who we would like to welcome to the group.

The 80-Column card project from the designs obtained at last years AGM has not yet started, but Mark Wills has said he can produce the boards.

Mr.Vice Chairman (Mark Wills): Mark has all the PCB manufacturing materials needed to produce the 80-column cards, although we will need to exactly work out how many we need, so anyone who wants one, let Trevor, Gary, Mark, or myself know about it, because Mark has got to pay for the chemicals himself to etch the board.

What Mark can't do is have the boards drilled, but this raised a smile between myself and Trevor, because we spent until 1:30am building the **ROMOX E/A** Supercart.

As far as Mark's role of programming officer is concerned, he has had a couple of enquiries from users which he has been able to answer.

Mark is heavily into C programming, and says his articles on the subject will continue for some time.

Membership Secretary (Alasdair Bryce): Alasdair was unable to attend the AGM because he is a solicitor, but has unfortunately been made redundant.

We all wish you luck in finding a new job Alasdair.

Alasdair's report was given by Mr. (overworked) Chairman!!!!

There has been a decline in membership and there were around 70 members who were due for renewal, and Alasdair has received two thirds of those.

Our membership is stable at around 140 members.

Those members who have renewed their subscriptions have been making donations to the disk library, and this has raised about £25 which will help in the purchase of new software.

Mr.Treasurer (Alan Rutherford): The cost of room hire has fallen considerably this year, and the cost of producing **TI*MES** has risen only slightly, but there has been a drop off in membership. The groups' bank balance is sufficient to cover this however:

TI-99/4A User Group (UK)

Income and Expenditure Account
For the year ending **1st April 1993**

Income		**Expenditure**	
Subscriptions	£1465.80	TI*MES	£1065.16
Interest	£36.01	Room hire	£70.00
Sales	£136.60	Equipment	£150.00
		Expenses	£57.74
	£1638.41		£1342.90

The groups' current bank balance is £4267.40

Mr.Editor: Due to Alan Bailey's unfortunate illness, the editors report was given by Trevor.
There is no shortage of material for publication, and Stephen Shaw has been able to obtain quite alot of material because he is in contact with many people all over the world. We have enough article material (without repeating an item) for **FORTY** full issues!!!

This material is coming from Canada, Germany, and Italy. This doesn't mean, however that we don't want articles from you! Even the console only people could get writing articles. You will see in this newsletter, an article from Gary Smith which describes a printer port which uses the cassette port and just a couple of chips.

Disk Librarian (Stephen Shaw): Stephen Shaw sent his apologies for being absent from the AGM due to a family illness, and therefore, his report was given by Mr.Chairman.

The disk library is still very quiet, but has benefitted from the donations made during the last subscription.

Cassette Librarian (Nicky Goddard): Nicky gave his final report as Cassette Librarian before handing over to Mark Wills. He's only had about three orders during the past year!

Module Librarian (Phil Trotter): Phil was absent from the AGM and has to relinquish his post as both **Publicity Officer**, and **Module Librarian**.

Phil did send down the entire collection of modules, however, and these were collected by the new module librarian.

ITEM 2: Committee Elections

Most of the committee positions have remained the same with the main changes being to the respective library positions.
It was decided that Mike Goddard would no longer have a position as Hardware Officer. This position has been removed from the committee. He has no time to actually undertake actual hardware projects for new devices, but will continue his valuable service of supplying hardware such as disk drives, joysticks and power supplies.

After the elections here's the official committee list:

Mr.Chairman	- Trevor Stevens.
Vice Chairman	- Mark Wills.
General Secretary	- Richard Twyning
Publicity	- ?????????
Membership Secretary	- Alasdair Bryce
Treasurer	- Alan Rutherford
TI*MES Editor & Distribution	- Gary Smith

Librarians:	Disk	- Stephen Shaw
	Cassette	- Mark Wills (again)
	Modules	- Francesco L Lama

** Please Note **

Your magazine articles should now go to **Gary Smith**
Your Module orders should go to **Francesco Lama**
Your Cassette orders should go to **Mark Wills**

You will also note that the position of publicity officer is still open. If you are interested, then contact Trevor or I about it.

ITEM 3: Discuss Group Future/Expansion

It was thrown in as an idea in the last issue of TI*MES by Trevor Stevens, that it might be possible to include another machine in the group, and have TI*MES split between the two. This was thrown out during our discussion at the A.G.M. The group and magazine is going quite steadily by itself up to now, so it's decided that we should continue as we are. Dedicated to the no.1 machine.

What was decided though was that we should make contact with **East Anglia Region 99'ers**, and **West Midlands T.I. Users** and make up a **'Super Group'**. Mark Wills says that East Anglia became very disjointed as a group when Scott and Jo Ann Copeland went back to the U.S.A. I was a member myself until I met Gary and Trevor. The East Anglia newsletter increasingly contained pages that were produced using an IBM PC, and it became very ropey! I then subscribed to TI*MES.

It's quite an amazing story how I became a member. I was at West Notts College of Further Education, and was doing a two year National Diploma in Software Engineering. About six weeks into the course we were told that we were getting a new student from Newark! He had to travel from Newark everyday. His name was Simon Sloane. Because of his resemblance to Norman on the Sun Alliance insurance T.V. advert, he was nicknamed Norman!

I got chatting with old Norm, and he said he'd got this friend called Gary who'd got a Texas, and had heard of the GENEVE! I gave Norm a map, and he gave this to Gary, and it was arranged for Gary to come and see me one Friday night.

He was amazed at the amount of software I'd collected over the years, and what the GENEVE would do, and the next time he came he said he'd got a friend from Rainworth who would like to see the GENEVE!!! Both Gary and Trevor turned up and Gary brought a load of disks for me to copy. I had no spare floppies, so I made some directories on the hard disk and copied each floppy into a seperate directory to sort out later. I can remember they were both amazed at the speed. This was before Trevor had even got his expansion box. One thing I do remember from that night was getting a true disk based version of TI-Invaders that Gary had brought which included the famous *#* cheat screen that wasn't available on the cartridge version!

Oops! Back to the subject! (I always get carried away in every article!)

Mark Wills has agreed to make contact with the members of East Anglia Region 99'ers and Gordon Pitt of the West Midland TI Users and get their views on joining us as one big group. It would mean we would have a greater chance of survival. All of our funds would be combined, but East Anglia Region 99'ers or West Midland TI Users would not lose their own identity. They would still be able to hold their own seperate workshops or meetings such as **Bloxwich**, but these would be publicized in TI*MES, and would benefit from all of us being able to attend after reading it in TI*MES. They would also have the benefit of attending our shows, workshops and A.G.M. What a wonderful world it would be! We'll have to see what sort of response Mark has received. Let's hope that he's got something to report on the matter in his article.

ITEM 4: Discuss Membership Fees

This discussion lasted for a considerable time with much deliberation. It was pointed out by Derek Haywood that it would be sensible to put up the subscription fee by a small amount now, rather than forcing the members to pay alot extra in the future if the subscription had to jump by a large amount. We had a vote on the matter and it was decided to put up the membership to £13.00. Therefore, the next time you subscribe, the new fee will be:

£13.00

This means that overseas subscribers will now have to pay £15.50 and for subscriptions including Air Mail it will be £18.50.

ITEM 6: Open Forum

This was intended to allow anyone with something to say to have it discussed. Most of the important issues had already been covered in items 3 & 4, so there was not alot left to say.
I think that during this time we discussed the venue for the AGM, but we will be staying with Derby for that, although it was mentioned that we might possibly have three shows or workshops in a year. One for the north, one for the south, and the AGM in the middle. If you know of any place that might be suitable near you that charges a reasonable price, then let us know about it.

It was also mentioned that the magazine has no regular place where you can have your letters printed. This has been corrected in this issue with the ***Consoletation Zone***.

On the actual day of the AGM, we got up to some very interesting things. Gary Smith, Mark Wills, Francesco Lama and myself went for a traipse around Derby in search of nutrition and beverage
During our meal we discussed many things of great importance such as graphics cards, C99 programming, and waitresses!
At the AGM, Francesco had a copy of Micropendium in which there was an unexpected advert for a **Ron Walters SCSI** Hard and Floppy Disk Controller card. This costs $170. Yes, $170!!! That's £109 + shipping charges + whatever our current government wants to ripp off of us in the way of import duty and VAT!
It's still not bad. I'll try and find out more information on this.

The next part of my article is a sort of a review for my CAD program. The program's not ready yet though. In fact I've not even started it! That isn't strictly true though, because I have got a very slow routine running in XB which allows you to select an option from either an ICON, or command line. The full version is only a matter of converting from the XB version and then adding the extra bits to it.

The reason the XB routine is slow is because I've made it multitask three things together. It allows you to type a command straight in such as LINE, while at the same time running a cursor routine to allow you to select drawing functions from icons. That is just about manageable for XB though, but what slows it down even more is a grid routine.

You can have a grid of pixels on the screen so you can exactly estimate where your cursor is. The trouble with the grid is that it takes hours to draw the grid on the screen. I did intend writing the CAD program in total XB, but it just isn't fast enough.

Now that I've obtained the full version of C99 V.4 from Mark Wills, that's the language I'll use. FORTRAN's no good because the 4A version won't let you have sprites and bit map at the same time because it puts the pattern descriptor table over the sprite table! Very sensible! I need a sprite to use as a cursor. I could use pixels as a cursor by saving the necessary pixels, displaying the cursor, and restoring the pixels afterwards. It's a good idea, but sadly unsupported. You can set a single pixel ON, of a certain colour, but you can't turn it OFF. If you plot a black pixel to the screen, you could overwrite it with a white pixel, but the pixel will still be set in the pattern descriptor table. If you plot a black pixel to the right of it, it will turn black again and you'll get blocking trouble.

What he should have done was include a pixel resetting routine which actually cleared the pixel completely by changing the particular 8-bit pattern of pixels in the pattern descriptor table.

With C99 you can use sprites in bit map mode, so I can use a sprite as a cursor, but I might not be able to delete individual pixels, and I definitely won't be able to delete lines or circles etc., so I'll have to put up with doing a complete screen clear and redrawing everything. It could take a bit of time if you've got a lot of objects on the screen.

The drawing area of the CAD program is 184 * 222 pixels. "Why that small?" you're all saying!

Well, I needed a bar at the top of the screen for displaying current cursor coordinates and image scale etc. and I've decided to use a 6 pixel high font which leaves one blank spacing pixel, and then the image border. That means that the maximum number of pixels the drawing area can have vertically is 192-8 which is 184. Why does it have an horizontal resolution of only 222 pixels though?

Well, that is due to the GENEVE. It gives the drawing area a true aspect ratio. The 9938 in the GENEVE and 80-Column card has a true aspect ratio display with true square pixels. To find the value for aspect ratio you simply divide the 9938 resolution, which is 512 * 424 pixels.

Therefore if you do 512 / 424 you get 1.2075471.

This means that the 9938 screen is 1.2075471 times wider than it is taller. Because monitors are wider than they are taller it means that all the pixels are perfectly square. The standard 4A however has 192 pixels down, which means that a true aspect ratio screen on the 4A should be 231.84904 pixels wide. The 4A has 256 pixels horizontally though, which means that it's vertical resolution should be 212 pixels, which is in fact the resolution of G7 screens on the 9938 which are 212 * 256 in 256 colours.

Therefore 1.2075471 * 184 = 222.18866 pixels wide.

So what use is a drawing program with only an area of 184 * 222 pixels when PICASSO has 384 * 480?

Well, if my idea works, and the program turns out as I hope, then it should be better than Picasso.

The program is based on units rather than pixels. If you want to draw a line 6,000,000 units long, then you could select the start point, and tell it to draw at whatever angle for whatever length you wanted.

What you'll realise though is that you can only see a maximum of 222 pixels of the line. But, what you can do is select VIEW EXTENTS, and the image will redraw immediately to display the maximum range of the picture. The 6,000,000 unit wide line will be scaled and will fit entirely into the viewing area. The same will go for any other object.

Therefore, if you want to start drawing something, then you can just keep piling lines or other shapes on the screen and not worry about how much space you've got left. With TI-Artist, GRAPHX, and even PICASSO, you have to plan your picture very carefully, or you'll have to move it across to fit another part in. With a full CAD program there's no problem at all. If your picture is already touching the right hand side of the screen and you want too add a bit more to the right, then just set the start point and tell it where to go.

Printing is another area where the program will improve over TI-Artist and PICASSO etc. Because the image is stored as objects, and not a bit map, each object can be scaled as it is printed. I'll design the program to work with 9-pin and 24-pin printers. In 9-pin mode you'll get a maximum of 1920 pixels across, but with 24-pin printers you'll get 2880 pixels across. That's 360 * 360 dots per inch. The program will look at the maximum extents of the currently loaded picture. This will then be divided into suitable sections that will fit onto the screen. The program will then plot each individual section of the image to the screen, and will dump it to the printer in sections. The more detailed parts of a picture that appear as a speck on the full screen view will be drawn in fine detail on the printer.

I'm also sick of TI-Artist, and other programs, chopping the edges off of images when they're printed. TI-Artist even chops the left edge of the left image and the right edge of the right image when you print two or more images across the page.

My program will avoid the non-printable areas and will totally scale the full image onto the page. Fonts are another problem on TI-Artist. Most PC software produces very smooth fonts on 24-pin printers, so I'm goingtto have a set of Super-Fonts! Each character will be drawn on a full TI screen at 256 * 192 pixels, but will be made of vectors. They will appear smooth, even if you zoom right down on them. When you include them in an image you'll be able to scale them to any size. If you use a very large font on TI-Artist, you will soon run out of screen area. Hopefully, my CAD program will be able to load a text file and give you the full text in the full Super-Font. There should be no rough edges at all when you print it out. It should come out O.K. on a 9-pin as well as 24. Just to show that there's no ill-feeling, it will do TI-Artist Fonts too (Hopefully, if I've got enough memory!) If I haven't got enough memory to support TI-Artist fonts I'll try and write a convertor which will convert them into "Super-fonts".

The easy part of developing the program will be a loader that works with Missing Link and will load an image produced with the CAD program and will scale it entirely to the TI screen. You can then save it as a TI-Artist image.

To save memory, the program will hopefully work similar to my Wordspace idea that keeps all of its "memory" on disk. It will use (hopefully if the file format is fully supported by C99) an INTERNAL/FIXED Relative file. Each record will have a starting symbol which will indicate what sort of object the record describes. This symbol will be full ASCII, which gives the possibility of 255 different CAD objects which will include LINE, ARC, POINT, ELIPSE. There will be no object for a circle. There's no point. Just draw a 360 degree ARC!

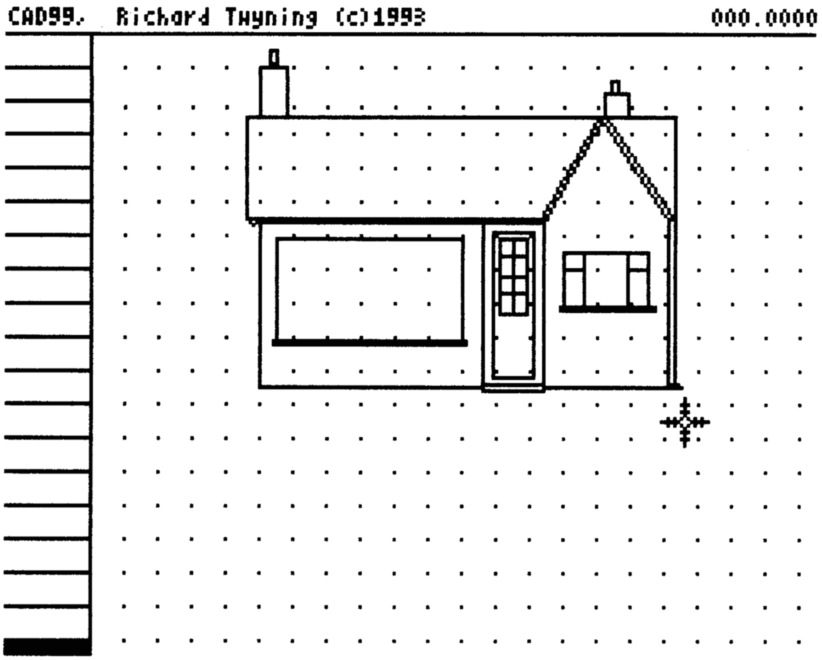
Somewhere in the magazine you should find a simulated screen dump of the program.

I've got some news from Trevor Stevens on the 80-Column card project. I thought I'd better mention something about it in case anyone else forgot.

Mark has found a RAM chip on the design which is now obsolete, but we will probably be able to find a replacement. For the address decoding of the circuit, he has selected some different chips and has reduced the chip count, which will make it even easier and cheaper to produce.

I was recently contacted by Bill Moran who has got an original Mechatronic 80-Column card. He has experienced problems with it though, and I have suggested that Richard Sierakowski might be able to fix it for him. I hope he has got it all sorted out, otherwise he can contact me again and Gary and I would be happy to look at it.

Another note on the V9990 board that Gary and I were designing. This project has been totally dropped, but we may be turning our attention to something else, such as a 34020 board, or an Hitachi GDP graphics board. I just thought it might be interesting to print the circuit diagram. You can see the layout of the circuit, but the text probably won't be readable since it started out as an A3 piece of paper!


CAD99. Richard Twyning (c)1993 000.0000

Those who attended the AGM may have seen Trevor's Editor Assembler 8K SuperCart which was made from a ROMOX cartridge. The design instructions for the SuperCart that were printed in Autumn 1992 TI*MES stated a 5565 RAM chip. You can now safely ignore this. I have found that the standard 6264 can be directly inserted in its place. 6264's are readily available from places such as RS.

However!

There is another possibility! We are still working on a way of decoding 32K into the cartridge port to give a full 32K SuperSpace, as opposed to an 8K SuperCart. I found out that a 32K chip will fit straight into the same socket as an 8K chip! Both chips have more or less the same pin out. The only difference is that on the 8K chip, pin 1 has no connection, and pin 26 is an extra chip select. You don't need this extra chip select. The ROMOX board doesn't use it. On the 32K chip, pin 26 is A13 and pin 1 is A14. This gives address bits from A0 to A14 which you will be able to calculate will give you an addressable 32K. If you intend undertaking the project of making the SuperCart, I recommend that you buy the 32K chip instead. You must bend up pins 1 and 26 before inserting it into the socket. These must then be grounded to a suitable ground somewhere. A ground can be found by tracing the ground from pin 14 of the chip. I think Trevor grounded his to a leg of a diode that went to ground. This has the advantage of providing a bit of extra smoothing to the signal!

The reason they must be grounded is that they will float around otherwise, and you will have access to different areas of the chip. The actual 8K area you are accessing could be one of four such areas on the chip, and this will drift around as the two additional address bits float.

The number of the 32K chip is μPD43256C-10L. The 10L bit means it's a 100 nano second chip, although 70 nano second chips are available that have faster access time. I suppose the μ sign means that it's a micro 32K because it fits into the same area as the 8K device. The 32K chip costs around £7 from RS.

While we're at it, for those who haven't got it, or have lost it, here's the pin outs of the Cray 1 GROM port:

Pin	Description	Pin	Description
1	RESET	2	GND (SYSTEM GROUND)
3	D7	4	CRUCLK
5	D6	6	CRUIN
7	D5	8	A15/CRU OUT
9	D4	10	A13
11	D3	12	A12
13	D2	14	A11
15	D1	16	A10
17	D0	18	A9
19	+5V	20	A8
21	$\overline{GS}$ (GROM SELECT)	22	A7
23	M0/A14	24	A3
25	M1 (DBIN)	26	A6
27	GROM CLOCK	28	A5

29	-5V	30	A4
31	GR (GROM READY)	32	$\overline{\text{WE}}$
33	GND (GROM GROUND)	34	$\overline{\text{ROM G}}$
35	GND (SYSTEM GROUND)	36	GND (SYSTEM GROUND)

Next is an item requested by John Murphy at the AGM. He wanted to know how to include a word that was created using my speech editor program as part of an Extended BASIC program.

The program creates a file called WORDFILE on DSK1. When you create a new word and wish to save it, it is appended onto the end of WORDFILE. When you want to use a word in your own program you must have it loaded. This can be a word that you have previously created and have loaded from WORDFILE, or it can be a word that you have just constructed from several words. If you select the **DATA STATMNTS** option you can view the data statements that you require to use the word in your own program.

You can have the data displayed to the screen, or the printer. If you display it to the screen, you can copy it onto a bit of paper and type it into your program later. To use the word in your program you will need the following code.

```
A$="" ::FOR LOOP=1 TO however many data items there are
READ A::A$=A$&CHR$(A)::NEXT LOOP
```

When you want to have the word spoken you can just use:

```
CALL SAY(,A$)
```

And that's all there is to it! And here's a challenge to console only people. The Speech Editor program will fit into console memory, so if you've got XB and cassette I want to see the code for a cassette based version. I must confess that cassette file handling is a bit of a mystery to me, so if anyone can write a few lines that convert it to cassette storage and send it to me, I can type it out if it's hand written and have it submitted to TI*MES.

You can find a copy of my Speech Editor to type in, somewhere at the end of my article.

During the Easter holiday I decided to upgrade my GENEVE's V9938 VDP RAM from 128K to 192K. In Alexander Hulpke's documents that come with YAPP he says that you should think twice about doing it if your RAM is not socketed!

Mine wasn't!

Something in my mind snapped, and I began one of my most dangerous projects! My entire set of GENEVE RAM's where 150 nano second access time devices, both CPU RAM, and VDP RAM! I removed it all! HAL 9995 was down for about five days until the replacement RAM arrived. I socketed the entire board and have replaced it with 80 nano second RAM. There is still some noise in the system on my buffer chips, and this still prevents it working correctly in GPL mode at 16MHz, which was the only reason I decided to upgrade. I'm 98% back to normal, as I've noticed a slight glitch in an area of memory that screws up a few sectors of my GENEVE MDOS RAMDISK. Having 192K of VDP RAM is amazing though. In YAPP you can do a full zoom on a 512 * 424 image, and there's also a colour clip art feature. One of the most amazing things though is the GIF loader that requires the extra RAM. You can load a GIF image directly into YAPP, and it even allows you to scale the picture horizontaly to allow it all to fit on the screen! Louigi played it cheap when he designed the GENEVE, and he omitted the extra bank of 64K video RAM.

On the 80-Column card for the 4A though, this RAM is standard. On the 4A though, to use the built-in hardcopy program you will need an 8K SuperCart, or a 32K SuperSpace.

I bet Mark thought I was going to forget to include something about CRU addressing! Well, I've just had to do an assignment on networking and operating systems. We've had to consider some sort of company and recommend a computer system and networking standard. Guess what I recommended? Yep! A Texas Instruments DX10 Mini Computer. We had to describe various parts of the operating system and I/O. Naturally, the I/O of a DX10 Mini Computer, is quite unique and very advanced. It is of course called **Communication Register Unit** addressing! I wrote a bit about this off of the top of my head, and it's the most simplified version of the theory I've ever written, so I thought I would just include it here!

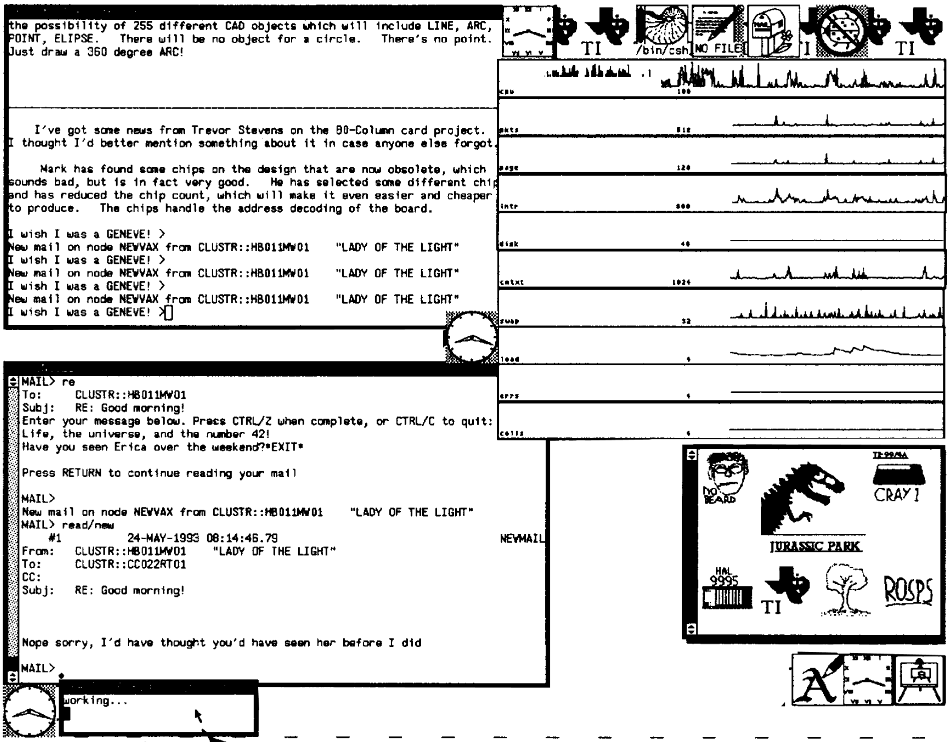
INPUT/OUTPUT

*The selected computer system, and operating system will also support a very advanced feature of I/O control known as a **Device Independant Operating System**. This is a feature of non-memory mapped devices which are addressed on a separate bus called the **Communications Register Unit**.*

*This is maintained by almost a complete sub-processor, called the **Programmable Systems Interface**. This processor receives I/O instructions from the CPU which already has the **CRU** Address of the individual device that has been selected for access placed in **R12** of its currently operational workspace. The I/O devices are actually addressed by the **Programmable Systems Interface**.*

*The **PSI** gets the required address from the CPU and sets up the CRU address bits to point to the selected device. The first bit of the particular CRU address space is used as an enable signal for a **Device Control ROM**.*

*This ROM contains software called a **Device Service Routine**, which is actually a collection of small, **Assembly-level** routines which control the actual hardware operation of the particular selected device. This ROM is physically mapped into the CPU's address space, but all ROMs are mapped to the same space. Therefore, only one such area of RAM*


```
the possibility of 255 different CAD objects which will include LINE, ARC,
POINT, ELIPSE.  There will be no object for a circle.  There's no point.
Just draw a 360 degree ARC!


   I've got some news from Trevor Stevens on the 80-Column card project.
I thought I'd better mention something about it in case anyone else forgot.

   Mark has found some chips on the design that are now obsolete, which
sounds bad, but is in fact very good.  He has selected some different chip
and has reduced the chip count, which will make it even easier and cheaper
to produce.  The chips handle the address decoding of the board.

I wish I was a GENEVE! >
New mail on node NEVVAX from CLUSTR::HB011MV01    "LADY OF THE LIGHT"
I wish I was a GENEVE! >
New mail on node NEVVAX from CLUSTR::HB011MV01    "LADY OF THE LIGHT"
I wish I was a GENEVE! >
New mail on node NEVVAX from CLUSTR::HB011MV01    "LADY OF THE LIGHT"
I wish I was a GENEVE! >
```

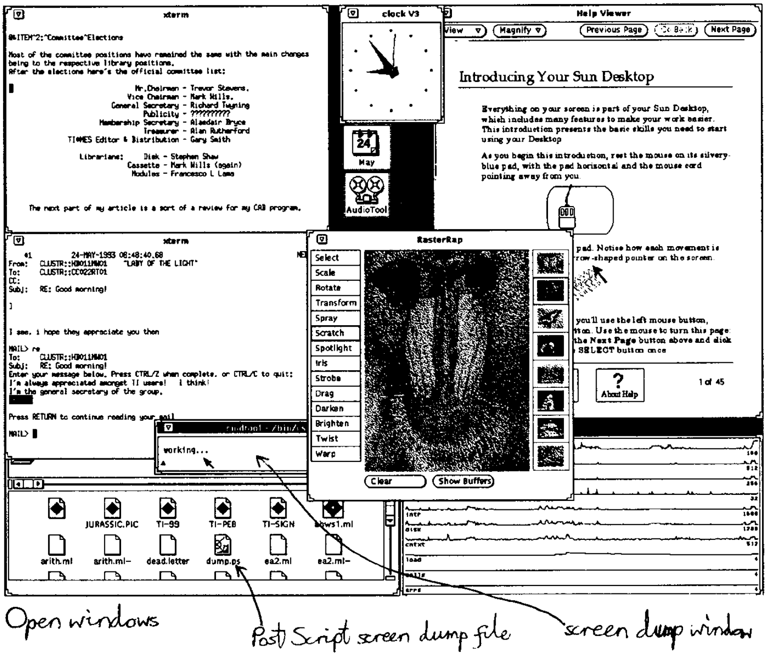
```
MAIL> re
To:     CLUSTR::HB011MV01
Subj:   RE: Good morning!
Enter your message below. Press CTRL/Z when complete, or CTRL/C to quit:
Life, the universe, and the number 42!
Have you seen Erica over the weekend?*EXIT*

Press RETURN to continue reading your mail

MAIL>
New mail on node NEVVAX from CLUSTR::HB011MV01    "LADY OF THE LIGHT"
MAIL> read/new
    #1          24-MAY-1993 08:14:46.79                                NEWMAIL
From:   CLUSTR::HB011MV01    "LADY OF THE LIGHT"
To:     CLUSTR::CC022RT01
CC:
Subj:   RE: Good morning!



Nope sorry, I'd have thought you'd have seen her before I did

MAIL>
```

```
working...
```

SUN Tools

This is the window that's actually running the screen dump program

After running the screen dump program to dump my SUN screen, I have to log onto my SUN account from a PC and file transfer the post script screen dumps onto the PC hard disk. Then I have to log onto the PC network, assign the drivers to access a post script laser printer and then PRINT the files from Messy DOS. Then you have to watch that no one uses the machine because it uses the hard disk for spooling!! Two hours later the screen dumps are ready for collection from bimbo services!!!



```
@%ITEM^2;^Committee^Elections

Most of the committee positions have remained the same with the main changes
being to the respective library positions.
After the elections here's the official committee list:

                              Mr.Chairman - Trevor Stevens.
                             Vice Chairman - Mark Wills.
                         General Secretary - Richard Twyning
                                 Publicity - ?????????
                      Membership Secretary - Alasdair Bryce
                                 Treasurer - Alan Rutherford
               TI*MES Editor & Distribution - Gary Smith

                    Librarians:      Disk - Stephen Shaw
                                 Cassette - Mark Wills (again)
                                  Modules - Francesco L Lama


     The next part of my article is a sort of a review for my CAB program.
```

```
    #1          24-MAY-1993 08:48:40.68
From:   CLUSTR::HB011MW01     "LADY OF THE LIGHT"
To:     CLUSTR::CC022RT01
CC:
Subj:   RE: Good morning!

]


I see, i hope they appreciate you then

MAIL> re
To:     CLUSTR::HB011MW01
Subj:   RE: Good morning!
Enter your message below. Press CTRL/Z when complete. or CTRL/C to quit:
I'm always appreciated amongst TI users!   i think!
I'm the general secretary of the group.

Press RETURN to continue reading your mail

MAIL>
```

```
working...
```

Introducing Your Sun Desktop

Everything on your screen is part of your Sun Desktop, which includes many features to make your work easier. This introduction presents the basic skills you need to start using your Desktop

As you begin this introduction, rest the mouse on its silvery-blue pad, with the pad horizontal and the mouse cord pointing away from you.

pad. Notice how each movement is
row-shaped pointer on the screen.

you'll use the left mouse button,
tton. Use the mouse to turn this page:
the Next Page button above and click
SELECT button once

need be allocated for Device Routines, because each ROM is mapped into this area individually, depending upon which base CRU address is selected. Each ***Programmable Systems Interface*** *can have 4096 such addresses, and Control ROM Selectors.*

Therefore, the computer system can have up to 4096 devices connected at the same time!

You'll be glad to hear that my article is now coming to a close. I just thought I'd mention that the section about my CAD program was typed using a *Sun* SPARC *Workstation!* When I had spare moments at university I thought, I could be typing my article now! The software I actually used to type it was a VAX editor using the Universities VAX! You can connect to the VAX through the SUN's, so I used the SUN as a VAX terminal. You can have as many VAX windows open as you like, so I have one at the top of the screen for VAX editor, and one at the bottom for VAX MAIL

When I'd finished my text, I accessed the VAX from home using TELCO, and opened a log to disk. I did a **TYPE TIMES.TXT** and the entire file was logged to my disk as a DV80 file which I merged into the rest of my article using Funnelweb editor! Why don't PC owners ever think of doing anything like that?!?!?!?

There might be another hardware product coming soon. A 32K card! The console only people have immediately turned to the next item now that I've mentioned new hardware, and a card. Let's wake 'em up shall we!

Console Only

YES! A 32K card for the console that plugs in on the right hand side, before you plug your speech synthesizer in. The reason it must be plugged in before the speech, is that the speech synth puts too much resistance on the data lines. The reason that the box isn't affected by this is because the box has an interface card that contains data buffers that amplify the signal after its strength has been reduced by the speech synth.
It must be very expensive. Console only people need some excuse for not buying one. Well, I'm afraid it will be very expensive. You will not have much change left out of £12.00.

YES £12.00!!

The advantage of it will be that if you've got XB and 32K you can use **CALL LOAD's**, which means you can use driver software for Gary Smith's very expensive console-only printer port!

I've also had an idea for a 256K Editor/Assembler Mega Cartridge! By having a single byte decoded has a memory mapping byte, you can poke the mapping byte using machine code, or **CALL LOAD** from TI-BASIC. This will switch in different 8K blocks as required. This would be useful for all sorts of things such as multiple menu systems. You can write title screen routines that reside at >6000 and appear as option 3 on the TI Master Title screen. Imagine if you'd got 32 different sets of these all in memory! You could even chain the menu's together and have the previous menu alter the mapping byte to load another menu in. Do that on an IBM!

As I said, it's just an idea yet! I'll have to consult on the matter with Gary and see what he thinks.

Coming Soon

Auto Route 4A! YES! On the Amiga is a program called GB Route Plus. This includes two data files which will both fit onto one TI disk. With the package are routines that load the data and produce routes from it. It should be possible to work out what this is doing and convert it into TI-Extended BASIC, or C99. I can then upload the map data from the Amiga onto a TI disk, and we'll have a route planning program of our own!

I've also not forgot about Lemmings 4A. I've just not had time to do anything with it lately.

Well, that's all I can think of for this article.

I hope you like my **JURASSIC PARK** picture. The dinosaur was actually drawn with Trevor's console using GRAPHX and Trevors joystick. Imagine how good it might have turned out if I'd actually used a TI Joystick!

I've dragged on for long enough!

That's it!!!

P.S. Not quite!! Late news from Mark!
He's found some serious errors on the 80-Column card. It will take an afternoon's work to re-design from scratch! It will use a single 256K RAM chip! The original design used six RAM chips! Mark's also thinking about reviving our V9990 project! I'm also sworn to secrecy on a very big new project. Watch this space, the final frontier may have been broken....

SPEECH EDITOR by Richard Twyning

```
1 FOR D=0 TO 14 :: CALL COLO
R(D,2,8):: NEXT D :: CALL SC
REEN(2):: PRINT "":"":""
 :: GOSUB 19000 :: ON WARNIN
G NEXT
2 DIM WORD$(9,1),SC(24,32)
3 CALL CHAR(124,RPT$("18",8)
,126,"000000FFFF",95,"000000
FF000000"):: ON ERROR 11
000
4 CALL CHAR(40,"000000030F0C
1818",41,"000000C0F0301818",
123,"18180C0F03000000",1
25,"181830F0C0000000"):: CAL
L TITLE
5 CALL OUTPUT(1,1,"~~~~)¦"):
: CALL OUTPUT(2,1,"GET ¦¦"):
: CALL OUTPUT(3,1,"NEW ¦
¦"):: CALL OUTPUT(4,1,"WORD¦
¦"):: CALL OUTPUT(5,1,"~~~~}
¦")
6 CALL OUTPUT(6,1,"~~~~)¦"):
: CALL OUTPUT(7,1,"SAVE¦¦"):
: CALL OUTPUT(8,1,"MAIN¦
¦"):: CALL OUTPUT(9,1,"WORD¦
¦"):: CALL OUTPUT(10,1,"~~~~
}¦")
7 CALL OUTPUT(11,1,"~~~~)¦")
:: CALL OUTPUT(12,1,"SAY ¦¦"
):: CALL OUTPUT(13,1,"MA
IN¦¦"):: CALL OUTPUT(14,1,"W
ORD¦¦")
8 CALL OUTPUT(15,1,"~~~~}¦")
:: CALL OUTPUT(16,1,"~~~~)¦"
):: CALL OUTPUT(17,1,"TA
KE¦¦")
9 CALL OUTPUT(18,1,"WORD¦¦")
:: CALL OUTPUT(19,1,"OFF ¦¦"
):: CALL OUTPUT(20,1,"~~
~~}{~~~~~~~~~~~~~~~~~~~~~~~~~~
~~")
10 CALL OUTPUT(21,1,"~~~~)(~
~~~~~~~)(~~~~~~~~)(~~~~)"):
: CALL OUTPUT(22,1,"SEE
¦¦TRUNCATE ¦¦DATA    ¦¦SEE ¦
")
11 CALL OUTPUT(23,1,"WRDS¦¦C
OMPONENT¦¦STATMNTS¦¦ TI ¦"):
: CALL OUTPUT(24,1,"~~~~
}{~~~~~~~~~}{~~~~~~~~}¦WRDS¦
")
12 CALL SPRITE(#1,140,2,10,1
0,#2,132,7,10,10):: CALL MAG
NIFY(3)
13 CALL CHAR(140,"0000000106
0C0811202140808043241 80E3146
982040C0ACB2C40830C00000
00")
14 CALL OUTPUT(14,31,"(~"):: 
 CALL OUTPUT(15,31,"¦Q"):: C
ALL OUTPUT(16,31,"¦U"):: 
 CALL OUTPUT(17,31,"¦I"):: C
ALL OUTPUT(18,31,"¦T"):: CAL
L OUTPUT(19,31,"{~")
15 CALL CLS :: GOSUB 15000
16 CALL CHAR(132,"0000000001
03070E1F1E3F7F7F3C1800000E38
60C08000404C38F0C0000000
00FF818181818181FF")
25 CALL SAY("ENTER+YOUR+COMM
AND"):: GOSUB 16000 :: IF CC
>40 THEN 30
26 ON INT(CR/40)+1 GOTO 1000
,2000,3000,4000,5000
30 IF CR>96 AND CR<144 AND C
C>240 THEN CALL SAY("FINISHE
D. GOODBYE"):: CALL CLEA
R :: END
31 IF CR>152 AND CC>208 THEN
 GOTO 8000
32 IF CR>152 AND CC>128 THEN
 GOTO 7000
33 IF CR>152 THEN GOTO 6000
34 IF WORD$(INT(CR/16),1)<>"
" THEN CALL SAY(,WORD$(INT(C
R/16),1))
35 GOTO 25
1000 CALL SAY("GET+WORD"):: 
CALL CLS :: DISPLAY AT(3,6):
"PRESS" :: DISPLAY AT(5,
6):"1. FOR TI RESIDENT WORD"
1001 DISPLAY AT(6,6):"2. FOR
 WORD FROM FILE" :: CALL SAY
("ENTER+YOUR+CHOICE")
1002 CALL KEY(0,K,S):: IF S=
0 THEN 1002
1003 IF K=49 THEN 1006
1004 IF K=50 THEN 1500
1005 GOTO 1002
1006 CALL CLS :: CALL SAY("E
NTER+WORD")
1007 ACCEPT AT(6,6)VALIDATE(
UALPHA,DIGIT,"#"):A$ :: CALL
 SAY(A$):: CALL SPGET(A$
,B$)
1008 GOTO 1600
1500 CALL CLS :: DISPLAY AT(
1,6):"GET WORD FROM FILE " :
: CALL SAY("ENTER+WORD")
1501 ACCEPT AT(6,6)VALIDATE(
UALPHA,DIGIT):A$
1502 OPEN #1:"DSK1.WORDFILE"
,DISPLAY ,VARIABLE 254
1503 LINPUT #1:I$ :: LINPUT
#1:I2$ :: IF A$=SEG$(I$,1,LE
N(A$))THEN B$=I2$ :: CAL
L SAY(,B$):: CLOSE #1 :: GOT
O 1600
1504 IF EOF(1)THEN CLOSE #1
:: DISPLAY AT(16,6):"WARNING
: WORD NOT FOUND" :: CAL
L SAY(,WARN$):: CALL SAY(,WA
RN$):: CALL SAY(,WARN$):: CA
LL SAY(,WARN$):: GOTO 15
00
1505 GOTO 1503
1600 GOSUB 15000 :: CALL SAY
("IN+WHICH+POSITION+DO+YOU+W
ANT+THE1+WORD")
1601 GOSUB 14000 :: WORD$(LO
C,0)=A$ :: WORD$(LOC,1)=B$ :
: GOSUB 15000 :: GOTO 25
2000 CALL SAY("SAVE+WORD"):: 
 GOSUB 17000
2001 CALL CLS :: DISPLAY AT(
2,5):" PRESS " :: DISPLAY AT
(4,5):" 1. TO VIEW WORDS
"

2002 DISPLAY AT(5,5):" 2. TO
 APPEND WORD" :: CALL SAY("E
NTER+YOUR+CHOICE")
2003 CALL KEY(0,K,S):: IF S=
0 THEN 2003
2004 IF K=49 THEN GOSUB 1200
0 :: GOTO 2001
2005 IF K=50 THEN 2010
2006 CALL SAY(,WARN$):: GOTO
 2003
2010 OPEN #1:"DSK1.WORDFILE"
,DISPLAY ,APPEND,VARIABLE 25
4
2500 CALL CLS :: DISPLAY AT(
```

RequiresSpeech Synthesiser, Extended Basic, Disk system

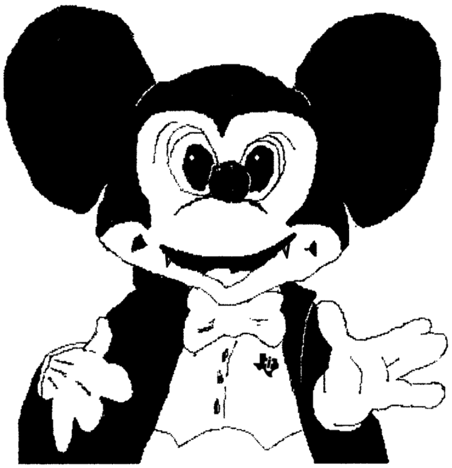
```
2,6):"HOW SHALL I IDENTIFY"
:: DISPLAY AT(4,6):"THIS
 WORD ON DISK?" :: CALL SAY(
"ENTER+WORD+NAME")
2501 ACCEPT AT(7,6)SIZE(16):
N$ :: GOSUB 17000 :: PRINT #
1:N$ :: PRINT #1:COMP$ :
: CLOSE #1 :: GOSUB 15000 ::
 GOTO 25
3000 CALL SAY("SAY+WORD")::
GOSUB 17000 :: CALL SAY(,COM
P$):: GOTO 25
4000 CALL SAY("TAKE+WORD+OFF
")
4001 CALL SAY("WHICH+WORD"):
: GOSUB 14000 :: WORD$(LOC,0
),WORD$(LOC,1)="" :: GOS
UB 15000 :: GOTO 25
5000 CALL SAY("SEE+WORDS")::
 GOSUB 12000 :: GOSUB 15000
:: GOTO 25
6000 CALL SAY("MAKE+WORD+SHO
RTER")
6001 CALL SAY("POINT+TO+WORD
"):: GOSUB 14000
6002 A$=SEG$(WORD$(LOC,1),4,
300):: L=LEN(A$):: CALL CLS
:: DISPLAY AT(16,5):"TRU
NCATE HOW MANY BYTES?"
6003 ACCEPT AT(18,5)SIZE(2)V
ALIDATE(DIGIT):BYTE :: L=L-B
YTE :: WORD$(LOC,1)=CHR$
(96)&CHR$(0)&CHR$(L)&SEG$(A$
,1,L):: GOSUB 15000 :: GOTO
25
7000 CALL SAY("OUT+PUT DATA"
):: GOSUB 17000
7001 CALL CLS :: DISPLAY AT(
1,7):"Display to Screen" ::
DISPLAY AT(2,7):"or prin
ter " :: DISPLAY AT(4,9):"1.
 Screen" :: DISPLAY AT(5,9):
"2. Printer"
7002 CALL SAY("PRESS+ONE+FOR
+SCREEN+OR+TWO+FOR+PRINTER")
7003 CALL KEY(0,K,S):: IF K<
49 OR K>50 THEN 7003
7004 IF K=50 THEN OPEN #1:"P
IO" ELSE GOSUB 24000
7005 EL=0 :: PRINT #K-49:"DA
TA ";:: FOR D=1 TO LEN(COMP$
```

```
)
7006 PRINT #K-49:ASC(SEG$(CO
MP$,D,1));",";:: EL=EL+1 ::
IF EL=8 THEN GOSUB 24020
 :: PRINT #K-49:"":"DATA ";:
: EL=0
7007 NEXT D :: PRINT #K-49:"
":""
7500 IF K=49 THEN GOSUB 2401
0 ELSE CLOSE #1
7501 GOSUB 15000 :: GOTO 25
8000 CALL SAY("SEE+#TEXAS IN
STRUMENTS#+WORDS"):: RESTORE
 20000 :: ROW=1
8001 CALL SAY("PRESS+ANY+KEY
+TO+STOP")
8002 READ A$ :: IF A$="*" TH
EN GOSUB 15000 :: GOTO 25
8003 DISPLAY AT(ROW,5):A$ ::
 CALL SAY(A$):: ROW=ROW+1 ::
 IF ROW=20 THEN ROW=1
8004 CALL KEY(0,K,S):: IF S=
0 THEN 8002 ELSE GOSUB 15000
 :: GOTO 25
11000 ON ERROR 11000 :: CLOS
E #1 :: CALL SAY("ERROR")::
GOTO 25
12000 CALL SAY("PRESS+ANY+KE
Y+TO+STOP"):: ROW=1
12001 OPEN #1:"DSK1.WORDFILE
",DISPLAY ,VARIABLE 254
12002 LINPUT #1:I$ :: LINPUT
 #1:I2$ :: DISPLAY AT(ROW,5)
SIZE(24):I$ :: CALL SAY(
,I2$)
12003 IF EOF(1)THEN CLOSE #1
 :: CALL KEYPRESS :: RETURN
12004 ROW=ROW+1 :: IF ROW=20
 THEN ROW=1
12005 CALL KEY(0,K,S):: IF S
=0 THEN 12002 ELSE CLOSE #1
:: RETURN
14000 GOSUB 16000 :: LOC=INT
(CR/16):: IF LOC>9 THEN CALL
 SAY("#THAT IS INCORRECT
#"):: GOTO 14000
14001 RETURN
15000 CALL CLS :: FOR D=0 TO
 9 :: DISPLAY AT((D*2)+1,5):
WORD$(D,0):: DISPLAY AT(
(D*2)+1,25)SIZE(4):LEN(WORD$
```

```
(D,1)):: NEXT D
15001 FOR D=2 TO 18 STEP 2 :
: DISPLAY AT(D,5):"________
____________" :: NEXT
 D :: RETURN
16000 CALL KEY(0,K,S):: IF S
=0 THEN CALL MOTION(#1,0,0,#
2,0,0):: GOTO 16000
16001 IF K=ASC("E")THEN CALL
 MOTION(#1,-10,0,#2,-10,0)::
 GOTO 16000
16002 IF K=ASC("X")THEN CALL
 MOTION(#1,10,0,#2,10,0):: G
OTO 16000
16003 IF K=ASC("S")THEN CALL
 MOTION(#1,0,-10,#2,0,-10)::
 GOTO 16000
16004 IF K=ASC("D")THEN CALL
 MOTION(#1,0,10,#2,0,10):: G
OTO 16000
16005 IF K=ASC("A")THEN CALL
 MOTION(#1,0,-20,#2,0,-20)::
 GOTO 16000
16006 IF K=ASC("F")THEN CALL
 MOTION(#1,0,20,#2,0,20):: G
OTO 16000
16007 IF K=ASC("3")OR K=ASC(
"4")THEN CALL MOTION(#1,-20,
0,#2,-20,0):: GOTO 16000
16008 IF K=32 THEN CALL MOTI
ON(#1,20,0,#2,20,0):: GOTO 1
6000
16009 IF K=ASC("Q")THEN CALL
 POSITION(#1,CR,CC):: CALL L
OCATE(#2,CR,CC):: CR=CR+
1 :: CC=CC+15 :: RETURN
17000 COMP$=WORD$(0,1)
17001 IF WORD$(1,1)="" THEN
17002 ELSE COMP$=COMP$&WORD$
(1,1)
17002 IF WORD$(2,1)="" THEN
17003 ELSE COMP$=COMP$&WORD$
(2,1)
17003 IF WORD$(3,1)="" THEN
17004 ELSE COMP$=COMP$&WORD$
(3,1)
17004 IF WORD$(4,1)="" THEN
17005 ELSE COMP$=COMP$&WORD$
(4,1)
17005 IF WORD$(5,1)="" THEN
17006 ELSE COMP$=COMP$&WORD$
```

```
(5,1)
17006 IF WORD$(6,1)="" THEN
17007 ELSE COMP$=COMP$&WORD$
(6,1)
17007 IF WORD$(7,1)="" THEN
17008 ELSE COMP$=COMP$&WORD$
(7,1)
17008 IF WORD$(8,1)="" THEN
17009 ELSE COMP$=COMP$&WORD$
(8,1)
17009 IF WORD$(9,1)="" THEN
RETURN ELSE COMP$=COMP$&WORD
$(9,1):: RETURN
19000 RESTORE 19004 ! [ or s
tarting                    line
 of
program ]
19001 READ A,B,C :: A$=CHR$(
A)&CHR$(B)&CHR$(C)
19002 FOR I=1 TO C :: READ Z
 :: A$=A$&CHR$(Z):: NEXT I
19003 WARN$=A$ :: RETURN
19004 DATA 96,0,100
19005 DATA 73,227,179,67,229
,226
19006 DATA 22,85,142,118,225
,157
19007 DATA 59,84,189,58,4,59
19008 DATA 222,80,236,244,84
,89
19009 DATA 187,75,241,221,34
,120
19010 DATA 172,44,37,84,241,
212
19011 DATA 178,211,84,111,37
,189
19012 DATA 204,138,211,120,2
00,81
19013 DATA 30,155,65,215,60,
167
19014 DATA 212,149,36,195,10
7,203
19015 DATA 44,71,146,204,164
,44
19016 DATA 43,12,73,48,163,1
50
19017 DATA 44,87,180,206,202
,134
19018 DATA 170,76,184,24,203
,51
19019 DATA 207,20,85,172,109
,109
19020 DATA 107,75,154,0,0,0
19021 DATA 0,0,240,0
20000 DATA "0","1","2","3","
4","5","6","7","8","9"
20001 DATA A,A1,ABOUT,AFTER,
AGAIN,ALL,AM,AN,AND,ANSWER,A
NY,ARE,AS,ASSUME,AT,B,BA
CK,BASE
20002 DATA BE,BETWEEN,BLACK,
BLUE,BOTH,BOTTOM,BUT,BUY,BY,
BYE,C,CAN,CASSETTE
20003 DATA CENTER,CHECK,CHOI
CE,CLEAR,COLOR,COME,COMES,CO
MMA,COMMAND,COMPLETE
20004 DATA COMPLETED,COMPUTE
R,CONNECTED,CONSOLE,CORRECT,
COURSE,CYAN,D,DATA
20005 DATA DECIDE,DEVICE,DID
,DIFFERENT,DISKETTE,DO,DOES,
DOING,DONE,DOUBLE,DOWN,D
RAW,DRAWING,E,EACH
20006 DATA EIGHT,EIGHTY,ELEV
EN,ELSE,END,ENDS,ENTER,ERROR
,EXACTLY,EYE,F,FIFTEEN,F
IFTY,FIGURE,FIND,FINE
20007 DATA FINISH,FINISHED,F
IRST,FIT,FIVE,FOR,FORTY,FOUR
,FOURTEEN,FOURTH,FROM,FR
ONT,G
20008 DATA GAMES,GET,GETTING
,GIVE,GIVES,GO,GOES,GOING,GO
OD,"#GOOD WORK#",GOODBYE
,GOT,GRAY,GREEN,GUESS
20009 DATA H,HAD,HAND,"#HAND
HELD UNIT#",HAS,HAVE,HEAD,HE
AR,HELLO,HELP,HERE,HIGHE
R,HIT,HOME,HOW
20010 DATA HUNDRED,HURRY,I,"
#I WIN#",IF,IN,INCH,INCHES,I
NSTRUCTION,INSTRUCTIONS,
IS,IT,J,JOYSTICK,JUST
20011 DATA K,KEY,KEYBOARD,KN
OW,L,LARGE,LARGER,LARGEST,LA
ST,LEARN,LEFT,LESS,LET,L
IKE
20012 DATA LIKES,LINE,LOAD,L
ONG,LOOK,LOOKS,LOWER,M,MADE,
MAGENTA,MAKE,ME,MEAN,MEM
ORY,MESSAGE,MESSAGES
20013 DATA MIDDLE,MIGHT,MODU
LE,MORE,MOST,MOVE,MUST,N,NAM
E,NEAR,NEED,NEGATIVE,NEX
T,"#NICE TRY#",NINE,NINETY
20014 DATA NO,NOT,NOW,NUMBER
,O,OF,OFF,OH,ON,ONE,ONLY,OR,
ORDER,OTHER,OUT,OVER,P,P
ART,PARTNER,PARTS
20015 DATA PERIOD,PLAY,PLAYS
,PLEASE,POINT,POSITION,POSIT
IVE,PRESS,PRINT,PRINTER,
PROBLEM,PROBLEMS,PROGRAM,PUT
20016 DATA PUTTING,Q,R,RANDO
MLY,READ,READ1,"#READY TO ST
ART#",RECORDER,RED,REFER
,REMEMBER
20017 DATA RETURN,REWIND,RIG
HT,ROUND,S,SAID,SAVE,SAY,SAY
S,SCREEN,SECOND,SEE,SEES
,SET,SEVEN,SEVENTY,SHAPE
20018 DATA SHAPES,SHIFT,SHOR
T,SHORTER,SHOULD,SIDE,SIDES,
SIX,SIXTY,SMALL,SMALLER,
SMALLEST,SO,SOME,SORRY,SPACE
,SPACES
20019 DATA SPELL,SQUARE,STAR
T,STEP,STOP,SUM,SUPPOSED,"#S
UPPOSED TO#",SURE,T,TAKE
,TEEN,TELL,TEN,"#TEXAS INSTR
UMENTS#"
20020 DATA THAN,THAT,"#THAT
IS INCORRECT#","#THAT IS RIG
HT#",THE,THE1,THEIR,THEN
,THERE,THESE,THEY,THING,THIN
GS
20021 DATA THINK,THIRD,THIRT
EEN,THIRTY,THIS,THREE,THREW,
THROUGH,TIME,TO,TOGETHER
,TONE,TOO,TOP,TRY,"#TRY AGAI
N#"
20022 DATA TURN,TWELVE,TWENT
Y,TWO,TYPE,U,UHOH,UNDER,UNDE
RSTAND,UNTIL,UP,UPPER,US
E,V,VARY,VERY,W,WAIT,WANT,WA
NTS,WAY,WE
20023 DATA WEIGH,WEIGHT,WELL
,WERE,WHAT,"#WHAT WAS THAT#"
,WHEN,WHERE,WHICH,WHITE,
WHO,WHY,WILL,WITH,WON,WORD,W
ORDS,WORK,WORKING
20024 DATA WRITE,X,Y,YELLOW,
YES,YET,YOU,"#YOU WIN#",YOUR
```

```
,Z,ZERO
20025 DATA "END OF DATA",*
24000 CALL SAY("PLEASE+WAIT"
):: FOR R=1 TO 24 :: FOR C=1
 TO 32 :: CALL GCHAR(R,C
,SC(R,C)):: NEXT C :: NEXT R
 :: RETURN
24010 FOR R=1 TO 24 :: FOR C
=1 TO 32 :: CALL HCHAR(R,C,S
C(R,C)):: NEXT C :: NEXT
 R :: RETURN
24020 IF K=50 THEN RETURN
24021 CALL KEY(0,J,F):: IF F
=0 THEN 24021
24022 RETURN
31000 SUB TITLE :: CALL CHAR
(132,RPT$("F",16))
31001 PRINT "            (~~~
~)"
31002 PRINT "          (~}
 {~)"
31003 PRINT "         ()
   ()"
31004 PRINT "        ()
    ()"
31005 PRINT "        ¦
     ¦"
31006 PRINT "       ()
     ()"
31007 PRINT "       ¦
      ¦"
31008 PRINT "       ¦
      ¦"
31009 PRINT "       ¦
      ¦"
31010 PRINT "       ¦
      ¦"
31011 PRINT "       ()
     ()"
31012 PRINT "        ¦
     ¦"
31013 PRINT "        ()
    ()"
31014 PRINT "         ()
   ()"
31015 PRINT "          {~)
 (~}"
31016 PRINT "            {~~~
~}":"":"     TWYNING ELECTRO
NICS":""
31017 CALL HCHAR(8,13,132,5)
:: CALL VCHAR(9,15,132,6)::
CALL HCHAR(10,17,132,4):
: CALL HCHAR(13,17,132,3)
31018 CALL HCHAR(16,17,132,4
):: CALL VCHAR(10,17,132,7)
31020 PRINT "        SPEECH
EDITOR":"" :: FOR D=1 TO 200
0
31021 NEXT D :: SUBEND
32000 SUB KEYPRESS :: CALL S
AY("PRESS+ANY+KEY")
32001 CALL KEY(0,K,S):: IF S
=0 THEN 32001 ELSE SUBEXIT
32002 SUBEND
32765 SUB CLS :: FOR D=1 TO
19 :: DISPLAY AT(D,5):" " ::
 NEXT D :: CALL VCHAR(1,
31,32,13):: CALL VCHAR(1,32,
32,13):: SUBEND
32766 SUB OUTPUT(R,C,P$):: F
OR D=1 TO LEN(P$):: CALL HCH
AR(R,C+D-1,30):: CALL HC
HAR(R,C+D-1,ASC(SEG$(P$,D,1)
)):: NEXT D :: SUBEND
```

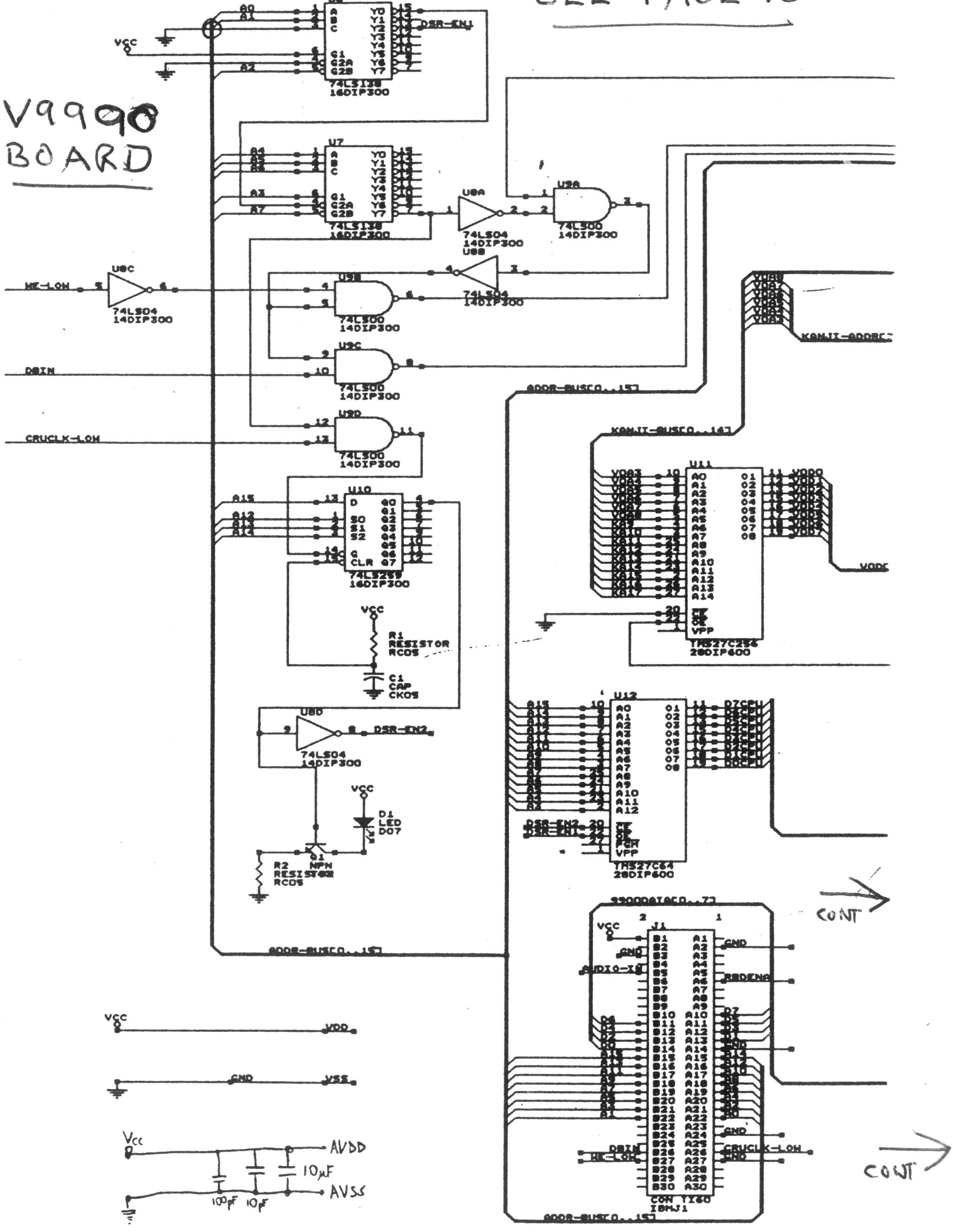


From a mine in South America came a piece of amber, containing the fossilized remains of a prehistoric mosquito. One of many that had fed upon the blood of dinosaurs.

From the DNA of that blood, science was able to recreate those giants. And, for the first time, man and dinosaur shared the Earth.

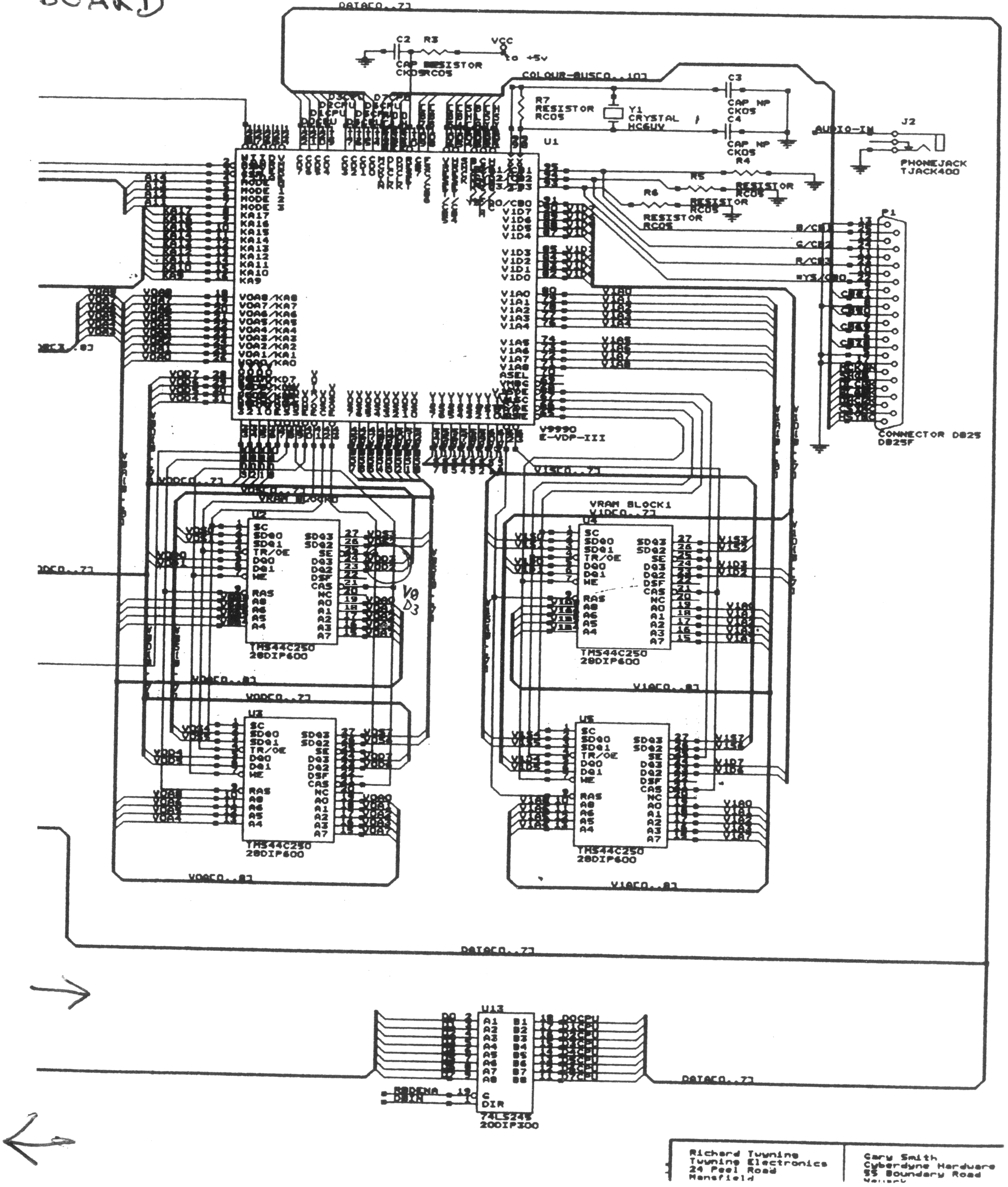
SEE PAGE 10

V9990 BOARD



REDUCED TO 25% OF ORIGINAL SIZE

V9990
BOARD



See Page 10

FROM THE CHAIRMANS CHAIR

T.Stevens c 1993

Hello there again. Well spring has sprung and summer races towards us. Just a thought, it's only 30 odd weeks till Christmas.. However I suspect that some of you will be reading this from some sunny place on this planet watching the world go by as you lounge on a pleasant shore. Inside this bumper issue are some real treats. So if you are out there on your holidays you will most likely look forward to trying out some of the routines on your return.

I have again been busy arranging things. You will see that I have again produced this issue as our Editor, Alan BAILEY has now had to give up the job through ill health. I on behalf of all the committee and members thankyou for what you have done for the club in the past years. It is very much appriciated. We now have a new Editor for the next issue. He is Gary SMITH. So if you look at the committee list you will see his details. Can you now send all your submissions to him.

There has also been other changes in the committee, so if you see the report on the AGM you will see who's who, and what's what.

There will also be some great news for all you out there in TI land, World Wide, next issue. This news if it gets off the blocks will send a few people in the computer world back to the drawing board. I will say no more than that, other than Watch this space!!!!!!!!

Now onto the continued saga of Sprites. Up till now we have discussed quite a few new programming tricks. Here are a few more for you to get your computer round.

You have all heard of multi tasking. This is where the computer is able to carry on with two programs at once. If you remember last issue we looked at how once a sprite is set into motion it goes off on its own without further program control. This means with sprites you can infact Multi Task by getting the computer to do more than one thing without loss of any speed. The following program shows how you can put a moving pattern on the screen with sprites and also display print statements. You can if you like use 'display at', instead of 'print'. Try and experiment with the routine and see what else you can do with it. Try putting music to the sprites, change the sprite colours or what ever. If you make up something really spectacular send it into me.

```
 50 RANDOMIZE :: CALL CLEAR
100 FOR N=1 TO 28 :: CALL SPRITE(#N,42,2,N*6,N*2,0,N*4):: NEXT N
110 PRINT "HELLO"
120 CALL DELAY(1000)
130 CALL CLEAR
140 PRINT "HELLO THERE!"
150 CALL DELAY(1000)
160 CALL CLEAR :: GOTO 110
170 SUB DELAY(D)
180 FOR A=1 TO D
190 NEXT A
200 SUBEND
```

You will see from the example above how good your computer really is. It is infact running, or multitasking 28 sprites and a display program all at once, with no loss of sprite positions which is important in this program for the effect.

We now move onto the problem of the Pacman syndrome. Have you ever tried to program a sprite to pick up something and put it down? What we are going to do is bring together some of the programing tips that we have discussed into pratice to show you what can be done. We will look at two bits of code. The first this issue is a simple version, the second which will be in next issue is a bit more complicated. So lets walk before we run.

In the first example your sprite will be set into motion according to an input from joystick 1. The program will check its position and place a block underneath it.
you can if you wish put call key statements into the routine to replace that of the joysticks.

```
100 CALL CLEAR :: CALL SCREE
N(5):: CALL CHAR(35,"FOFOFOF
O"):: CALL SPRITE(#1,35,2,89
,121):: CALL COLOR(1,1,11,2,
7,7)
110 CALL JOYST(1,X,Y):: CALL
 MOTION(#1,-Y*2,X*2):: CALL
POSITION(#1,R,X):: IF R>188
THEN R=1-(Y>0)*187 :: CALL L
OCATE(#1,R,X)
120 CALL GCHAR(INT((R+7)/8),
INT((X+7)/8),Y):: IF Y=32 TH
EN CALL SOUND(-90,660,9):: C
ALL HCHAR(INT((R+7)/8),INT((
X+7)/8),40)
130 CALL KEY(1,X,Y):: IF Y T
HEN CALL CLEAR :: GOTO 110 E
LSE 110
```

NB If you have a Version 100 Extended Basic Cartridge change line 110 to CALL MOTION(#1,-Y,X)

To find your version of cartridge Type NEW then CALL VERSION(V):: PRINT V

If we go through the program. Line 100 clears the screen and then sets it to screen colour dark blue (5). The character 35 is then defined as a small block. Then the sprite #1 is defined as character 35 with the colour black. It will be placed on the screen at dot row 89, dot column 121. The color command then turns sets 1 and 2 so that set 1 is transparent on darkyellow and set 2 dark red foreground and the same on the background. You will note that the space character is ascii 32. So when the set 1 is changed the space color turns dark yellow. So we have a yellow screen with a dark blue boarder. After the set up in 100 we now go to 110. This commences a loop through 110 120 and 130. The line 110 sets the joystick routine in motion. (See previous artical for joysticks). Then the call position is called. This places in the variable R(Row) and X(col). We can reuse the variable X but we must retain Y. The if statement works like this If R IS GREATER THAN 188 THEN IF Y IS GREATER THAN 0 (or stick pushed up), THEN R=188. (The call locate relocates your sprite at the variables R and X, which now will be at the bottom of the screen.) ELSE IF Y IS NOT GREATER THAN 0 (or stick pushed down), THEN R=1. (The Relocate puts the sprite at the top of the screen). ELSE IF R IS NOT GREATER THAN 188 THEN LEAVE R ALONE. (Then do not relocate the sprite.

That is this issues programs. Next time we will start the harder and more complex code, which will allow you to move items set on the screen and place them into boxes. This you will see lead onto greater things.

I hope you enjoy these articals. If you have any query at all drop me a line with a SAE inside so I can return you query as soon as possible.

SILLY SOUNDS FROM YOUR TI (T.S)

Some time back I put into one of my articals some short sound programs. These will run on any consol. I had some comments from a few members on how they liked them. So here are a few more. Also for those who have the TE2 cartridge some silly sounds for them.

BASIC PROGRAMS

```
100 REM RADIO
110 N=1
120 F=RND*15000+110
130 A=RND*30
140 CALL SOUND(-99,111,30,)
    111,30,F,A,-8,30,-A)
150 N=N+1
160 IF N=100 THEN 170 ELSE 120
170 END
```

```
100 REM TELEPRINTER
110 N=1
120 CALL SOUND(22,2975,0)
130 FOR D=1 TO 5
140 S=850*INT(RND*2)
150 CALL SOUND(22,2125+S,0)
160 NEXT D
170 CALL SOUND(31,2125,0)
180 N=N+1
190 IF N=30 THEN 210 ELSE
120
200 END
```

TE11 PROGRAMS

```
 50 REM SNAKE
100 OPEN #1:"SPEECH",OUTPUT
120 PRINT #1:"//0 0"
130 PRINT #1:"GHGHGHGHGHGHGHG
    GHGHGHGHGHGHGHGHGHGHGHGHG
    GHGHHGHGHGHGHGHGHGHGHGHGH
```

```
 50 REM HELICOPTER
100 OPEN #1:"SPEECH",OUTPUT
110 PRINT #1:"//0 0"
120 PRINT #1:"MNMNMNMNMNMNMNMNM
    NMNMNMNMNMNMNMNMNMNMNMNMN
    MNMNMNMNMNMNMNMNMNMNMNMNM
```

In data section at 130 try PKPKPK,ZKZKZK,JJJJJ,1A1A1A1A,GNGNGN. There are many more to find so get to it. The line 110 alters the pitch rate. To talk properly you can reset with //43 128.

MARK WILLS

```
%%%%  %%%%  % %    % %%%%% %      %
%   % %   % % %    % %     %      %
%   % %   % % %    % %     %      %
%   % %%%%  % %    % %%%   %      %
%   % %   % % %    % %     %      %
%   % %   % %  %  %  %     %
%%%%  %   % %   %%   %%%%% %%%%%  %  NO. 4
```

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

Greetings once again. As a writer for the magazine, I am always surprised at how fast three months comes around and it is time for me to submit my material for the mag. There I was sitting in front of the telly watching the Eurovision Song Contest (well someone has to watch it) when it suddenly struck me that the editors deadline was fast approaching and I hadn't even begun to write any material for Drivel.

So whats new? Quite a lot actually. The AGM has been and gone, and very interesting it was too. I believe my mate Richard is reporting on it so I won't duplicate any efforts here, except to say that I thought it was very succesful, with a good turnout.

Unfortunately, owing to the intense pressures of being a teenager (and a lot of us have fond memories of that i'm sure!) Nicky Goddard is no longer able to continue in his role of cassette librarian, so I have taken over. If any person(s) wish to order from the cassette library then please forward your orders to me. At the time of writing I have not had the chance to go through them (theres forty disks full of software) but I hope to get the time to develop a database using TI-BASE which will catalog every program in the library including a short review, so I should be able to give more details of what is actually available for the next issue (head on the block here!). Potential users of the library should note that having the library contents on a database such as TI-BASE will give the potential user more flexibility when ordering software. All titles will be filed under their respective category in the database and will contain details about the hardware neccessary to run the programs etc. This will allow a potential user to place an order for a software title without knowing its name or code number. For example "I am looking for a word processor type program for my TI. I have Extended basic and 32k plus a tape player. Can you suggest anything?". Using TI-BASE it will be a simple matter to interrogate the database and pull up the most appropriate peice of software in no time. Again, I hope to be able to offer this service from the next issue. (Gulp!)

All members should now be aware that I have moved house again (not permanantly - just while I'm working here!) and have moved to London.

My new address is : 207a Field End Road, Eastcote, Middx. HA5 1QZ.
My phone number is 081 [illegible], but please do not phone between the hours of 9.00am to 5.30pm as it is also my works number during office hours, but is a private number after those hours.
I would be interested in hearing from any TI'ers in the London area with a view to meeting up and talking about TI's. John Stocks, what are you doing these days?

Any road, this issue sees more on c99 as promised last issue. This issue we will look at actually controlling program flow/decision making etc.

c99 for the slightly more initiated!

c99

The IF command

Users of BASIC will be aware of the IF command which allows the computer to make a decision and execute a series of instructions according to the result of the decision. (Always either true or false). In basic, the IF command takes the form :

```
      IF expression THEN instruction(s)
```

ie

```
      IF A=4 THEN P=0
```

As you can see, P will equal zero when the expression evaluates to true. ie when A=4

c is very similar but its syntax is different:

```
      if(expression) intruction;

                  or

      if(expression) {
       multiple instructions;
      }
```

So taking the first c example above and replacing it with some actual c code:

```
      if(a==4) p=0;
```

is the equivalent c way of saying the same as the BASIC statement above.

The second c example is as follows:

```
      if(a==4) {
       p=0;
       t=44;
       z=(p+2)*t;
       etc...;
       etc...;
       etc...;
      }
```

The above example is called a "block if" because there is a "block" of code (the code between the curly brackets - or braces to give them their proper name) which is executed every time the condition inside the brackets evaluates to true.

There are a few rules and regulations we need to make clear when using if's relating to thier syntax, just to clear up any confusion:

A single if:

The expression to test must be enclosed inside brackets.
The statement to execute in the event of the expression evaluating to true must terminate with a semicolon. (;)

Block if's:

Again, the expression to test must be enclosed within brackets.
The beginning of the code block must be indicated with an open brace ({). The end of the clode block must be indicated with a closed brace (}). All statements inside the code block must be terminated with a semicolon (;) - as indeed all c statements must do.

Relational operators:

This is probably a good time to discuss the difference between c's relational operators and BASIC's, as they are different. A simple list will clarify the differences better than I can so...

Operator	Description	BASIC version
==	Is equal to ?	=
!=	Is not equal to ?	<>
>	Is greater than ?	>
<	Is less than ?	<
>=	Is greater than or equal	>=
<=	Is less than or equal to	<=
!>	Is not greater than ?	<x*
!<	Is not less than ?	>x*
&&	Comparative AND	Not supported
||	Comparative OR	Not supported

*Is not greater than, and is not less than is not really directly supported in TI-BASIC, so in order to express an equivalent expression in TI-BASIC you would say "is LESS than x" or "is GREATER than x" where x is the value you are testing, respectively.

Comparative AND and OR, which again is not DIRECTLY supported, allows you to mix multiple expressions in the same test. Example:

```
     if(a==4 && b==2) p=41;
```

p will be set to 41 when a==4 AND b==2

```
     if(a==21 || z==1) x=2;
```

x will be set to 2 when a==21 OR z==1.

I guess thats if's more or, less covered, except to say that you can call another part of your program (a function) with an if like this:

```
     if(a==4) game_over();
```

Will call the game_over routine when a==4.
This of course can be done with code blocks:

```
      if(a==4) {
       score=score+1000;
       game_over();
      }
```

If's and if code blocks can also be nested:

```
      if(a=2) {
       if(b=2) game_over();
      }
```

Also, you can use the ELSE clause, like in basic:

```
      BASIC
      -------------------------+----------------------
      IF A=1 THEN Z=2 ELSE Z=3 ¦ if(a=1) z=2; else z=3;
```

Note also, as stated in the first tutorial, c is case SENSITIVE. So a in this program:

```
      main()
      {
       int a;
       int A;

       a=a+1;
       A=A+1;
      }
```

a and A are treated, unlike BASIC, as completely seperate variables in thier own right.

A quick aside here, the lines a=a+1 and A=A+1 can be re-written as:

```
      a++;
      A++;
```

This in c says increment by one, and generates smaller code (and faster) than doing it the long way. You can also decrement by one:

```
      a--;
      A--;
```

Will decrement the two SEPERATE varaibles a and A by one each. Those of you who know machine code will spot that the compiler can use the INC and DEC instructions rather than A and S instructions which require operands and thus is larger and slower. Whenever you need to add or subtract one, you should use this method.

The while Loop:

Another powerful and easy to use method of controlling program flow is the while loop. It looks like this:

```
      while(expression) do something;

                    or

      while(expression) {
       do loads of things...;
      }
```

In the while loop, a command or block of code can be executed for as long as a particular condition is true. Or, put another way, code can be made to execute WHILE the test expression is true - lets look at an example:

```
#include dsk1.conio

#asm
 REF PRINTF
#endasm

main()
{
 int a;
 a=0;
 while(a<100) printf("a is %d",a++);
}
```

Lets go through this program line by line and work out what the program does.

```
#include dsk1.conio
```

This line, tells the compiler to include the file conio when compiling. It contains certain defined variables that you may refer to in your programs - we're not using any here but its good practice to get into the habit of including it all the time.

```
      #asm
```

This line tells the compiler that all lines up until #endasm are "raw machine code" and not to alter them in any way, just to pass them straight on to the source file for assembling. In this case all we are passing is a REF to the assembler because we will be loading the PRINTF file at runtime because we are using the PRINTF command to display the contents of our variable a.

```
       REF PRINTF
```

As just mentioned, this line is passed un-modified through to the assembler source file in. Note that there is a space before the REF. It is IMPORTANT!

```
      #endasm
```

Tells the compiler that we have finished passing machine code through to the assembler source file.

```
      main()
```

This indicates the beginning of the function called main. Remeber that all c programs start and end in main().

```
      {
```

The open brace means that all code is intended for the function called main.

```
      int a;
```

This line tells the compiler to reserve space for an integer variable called a.

```
      a=0;
```

Sets the variable a to zero.

```
      while(a<100) printf("a is %d",a++);
```

This line looks at the value of a. If a is less than 100 (ie the expression is true) then the value of a is printed on the screen, and then increased by one for the next time round (the ++). The %d in the printf command means that the operand folloowing the comma is to be treated as an integer, which we defined before (int a;)

Eventually, a will become 100 and the expression in the brackets (a<100) will evaluate to false. At this point, control is passed to the next line of the code, which, in our case, is the end of the main() function, so the program simply stops.

```
      }
```

Tells the compiler that we have finished our code for the main function, and, in this case, finished altogether!

The other technique worth a mention, and the final one for this issue is the do/while loop:

The do/while Loop:

Another very powerful way of controlling program flow/iteration, it looks like this:

```
      do {
       various things;
      } while(expression);
```

You may wonder what the difference is, well, the difference is that the code inside the curly braces will ALWAYS be executed at least once, because the expression is tested at the END of the code block. This is a very handy method for implementing things like menus, where you put a number of options to select from, one of them being quit. If the user presses quit, the program flow will fall out of the bottom of the do/while loop because the test expression is testing for the quit key being pressed. Handy and efficient.

I will leave it here for this month, next month we'll actually start doing things like putting things on the screen, reading the keyboard etc. It is at this point that you will see just how much faster c is than BASIC.

To give you an idea of how fast it is, I present a program here for you to type in and try. It is called a bubble sort and its function is to sort an array of numbers into the correct order. The array is filled backwards in this program, which happens to be the worst case as far as a bubble sort routine is concerned, so it has to work very hard. I also present an Extended BASIC version that does exactly the same for you to try for speed comparison purposes.
Have fun with your TI.

P.S Does anyone know how the Quick Sort sorting algorithm works, and if so, would you kindly publish the details, or write to me direct. Thanks.

Mark Wills.

Vice Chairman.
Programming Official.
Cassette Librarian.
(Busy!)

The programs...

Note: Type in the c99 version using editor assembler. Save as SORT;C
run c compiler:

Choose option 5 from ed/as menu, give DSKx.C99C as the filename - where x is the drive number.
When asked for the input filename type DSKx.SORT;C as before.
Specify the output filename as DSKx.SORT;S as before.
Press n when asked for a re-run.
(if you get any errors, you've made a typing mistake!)

Assemble using the TI assembler:
Load the assembler. Type DSKx.SORT;S for the source filename.
Type DSKx.SORT;O for the object name.

When finished, select option 3 from the menu and load the following programs.

SORT;O
PRINTF
CSUP

PRINTF and CSUP are on the c99 disks.

When you have done this, press FCTN 3 then enter.
Type START for the start name, the program will then run.

You should find that the c99 version run approx. 8.18 times faster than the XB version!

```
/* Bubble sort 1.0 (c) Mark Wills
   Written May 1993 in c99. Compiler by Clint Pulley (clever dude!)
   Total sort time : 27.00 seconds for 50 elements
*/

#include dsk1.conio
#define EL 50 /* change this number for more/less elements */

#asm
 REF PRINTF
#endasm

main()
{
 int array[EL]; /* reserve space for our pre-sorted list */
 int i,a,s,p; /* integer variables that we'll be using */

 putchar(FF); /* clear screen */
 puts("Filling the array... \n");

 a = EL;
 for(i=0; i<EL; i++) {
  array[i] = a--;
 }

 i=0; s=1; p=0;

 puts("Sorting array...");

 while(s!=0) {
  s = 0;
  for(i=0; i<(EL-1); i++) {
   if(array[i] > array[i+1]) { /* if array(i+1) > array(i) */
    a = array[i+1];           /* then swap them over      */
    array[i+1] = array[i];    /* in this                  */
    array[i] = a;             /* "if" code block          */
    s++;                      /* another swap done so add 1 to swap variable*/
    locate(4,1);
    printf("Swaps this pass = %d    ",s);
   }
  }
  locate(5,1);
  printf("Passes so far = %d\n",++p);
 }
 putchar(FF);
 puts("Array sorted:");

 for(i=0; i<EL; i++) printf("%d ",array[i]);
}
```

```
1 ! Bubble sort 1.0 XB version (c) M.Wills.
2 ! Total sort time : 3 Mins 42 seconds for 50 elements
10 OPTION BASE 0
20 EL=50 ! change this for more or less elements, and change the number in brack
ets in line 30 to the same value
30 DIM ARRAY(50)
40 CALL CLEAR
50 DISPLAY AT(1,1):"Filling array..."
60 A=EL
70 FOR I=0 TO EL
80 ARRAY(I)=A :: A=A-1
90 NEXT I
100 DISPLAY AT(2,1):"Sorting array..."
110 S=0
120 FOR I=0 TO EL-1
130 IF ARRAY(I)>ARRAY(I+1)THEN A=ARRAY(I+1):: ARRAY(I+1)=ARRAY(I):: ARRAY(I)=A :
: S=S+1 :: DISPLAY AT(4,1):"swaps this pass =";S;"   "
140 NEXT I
150 P=P+1
160 DISPLAY AT(5,1):"Passes so far =";P
170 IF S<>0 THEN GOTO 110
180 DISPLAY AT(7,1):"Print array sorted:"
190 FOR I=1 TO EL
200 PRINT ARRAY(I);
210 NEXT I
```

TI-99/4A USERS GROUP (UK)

MEMBERSHIP NEWS

by Alasdair Bryce

There are several new recruits to announce since issue 40. A warm welcome goes to David Jenkins, John McCartney and Scott Whitley as well as a welcome back to Brian Wickham who has rejoined after a short absence.

After the mass renewal following the last issue and the A.G.M. things should be a little quieter on the membership front this month but there are still a fair number of you due to renew with this issue. I very much hope that as many of you as possible will sign up again for at least another 12 months. With your input TI*MES continues to go from strength to strength and the group offers access to some of the best software and advice available for the TI. If anyone is thinking about moving on to bigger and arguably better things then why not keep the TI running? There are a good number of PC users out there who wouldn't be without their old TI.

Lastly for now, I read Walter Allum's letter in issue 40 and as a result I spent some time checking the comments which members had made on their subscription renewal forms. Unfortunately I can't really give any helpful answers to Walter's query as very few of you seem to use the comments section of the form. Of the few who do most simply offer welcome encouragement and praise for TI*MES although comments on the print quality of TI*MES were duly noted and a couple of others made known their displeasure at Stephen Shaw's "Rambles" being muzzled. If any of you do have comments that you want to make about TI*MES or the group generally then why not jot it down on the back of the form? - I do read them you know.

PROGRAMMING MUSIC THE EASY WAY

PART 4

by Jim Peterson

The first three parts of this series were written and published some time ago, so I had better review.
In Part 1, I showed you this one-line routine to set up a musical scale.

```
100 DIM N(36):: F=110 :: FOR
 J=1 TO 36 :: N(J)=INT(F*1.0
59463094^(J-1)+.5):: NEXT J
:: N(0)=40000 ::GOTO 110
101 D,T,A,B,C,V1,V2,V3,J,X,V
102 CALL SOUND
103 !@P-
```

That sets up a scale of three octaves beginning with A. If you decide to change the music to a higher key, just change the 110 to 117, 123, 131, 139, 147,156,165, 175, 185, 196, 208 or 220. In fact, for some music you will have to change it, if the program crashes with a BAD VALUE error message.

If you have programmed the music with high notes, you can lower the key by changing 110 to 104, 98, 92, 87, 82, 78, 73, 69 or 65. Again, if you try to go too low you will get that BAD VALUE message.

I have given N(0) a value of 40000, which creates a tone too high to be heard. This can be used to silence a note, but it can also cause a crash when used with some of the following routines. If you are programming three voices and want to play a single note, the easiest way is to give all three notes the same number, such as A,B,C=10. If you need a silent rest, play all the notes at an inaudible volume by V1,V2,V3=30 and then, after the GOSUB, restore their original volume by V1= (whatever is in line 110) and the same for V2 and V3.

Lines 101-103 are a pre-scan routine to start the music playing sooner. There will still be a few seconds delay while that array is set up in line 100. You can perhaps shorten that delay slightly by changing the 36 to the highest note number you have used in programming your piece.

However, Bruce Harrison wrote for me an assembly link which eliminates any delay; this also makes it possible to change key while the music is playing. I won't list the source code here, because everyone is afraid to key in source code anyway, but it is available on my TI-PD disk #1143 and will also be on a tutorial disk on this type of music programming.

Part 2 of this series contained a listing of a program to easily give you the numbers you would need in order to key in a particular piece of sheet music. If you don't have that, you can just take a piece a paper and list the scale A Bf B C C# D Ef etc., on through as many as you will need, and then number them consecutively. For the length of the notes, give the shortest note a value of 1 unless it also appears as a dotted note, in which case it must be 2, and then number the others according to their relative length - for example, 2 for a quarter note, 3 for a dotted quarter, 4 for a half note, 8 for a whole note.

Part 2 showed you how to key in single-note music, and Part 3 showed how to do 3-part harmony. To recap briefly -

First, save yourself a lot of work by identifying any groups of notes in the sheet music that are repeated two or more times. Mark them off wherever they appear. Key them in first, starting with line number 500; at the end, put RETURN. If you find another such series, label it 600 and do the same; you may find several such series. Just stay below line number 1000, which is reserved for mergeable routines. Then, while you are programming the music and come to such a series of notes, just put in GOSUB 500 or whatever.

Start keying in your music in line 120; line 110 is reserved for a line to be merged in. To key in the music, just give T the number for the length of the first note, and give A, B and C the numbers for the melody and first and second part harmony. Then GOSUB 1000. For instance, T=1 :: A=23 :: B=18 :: C=12 :: GOSUB 1000 .

And for each succeeding note, give a new value to whatever changes; if T is still 1 and B and C are still the same, all you need is, for instance, A=19 :: GOSUB 1000.

Merge in one of the following routines, put in a line 999 STOP, and after every several notes enter RUN and listen to what you have done so far, to catch any errors while it is still easier to find them.

You can merge in any of the following routines to create many different musical effects. The D in line 110 controls the tempo of the music; change it as you wish. V1, V2 and V3 are the volume (loudness) of the three voices; adjust them as you like.

Key this in and save it by SAVE DSK1.PLAY1,MERGE

```
110 D=500 :: V1=1 :: V2=5 :: V3=7
1000 CALL SOUND(D*T,N(A),V,N(B),V,N(C),V):: RETURN
```

That plays simple 3-part music, all at the same volume, which may sound rather harsh to your years. Try changing the second V to V2 and the 3rd one to V3. Save that as PLAY2.

For a bass accompaniment in the 3rd voice, change that to

```
CALL SOUND(D*T,N(A),V1,N(B),
V2,N(C)*3.75,40,-4,V3)
```

For a bass melody with accompaniment, change the A to C, V1 to V3, C to A and V3 to V1.

For the melody in two voices two octaves apart, change the C back to A and the V3 back to V1. Are you beginning to see how many different effects can be created by making changes in just this one line? Save any ones you like in merge format with a different name for each.

Perhaps those bass notes sound too deep. Try changing the 3.75 in any of those routines to 7.5 . Better yet, change it to X and add :: X=3.75 to line 110. Then you can switch back and forth in your music by simply X=7.5 or X=3.75. Getting interesting, no?

Music played in that way has a strong throbbing beat, so try this method -

```
110 D=4 :: V1=1 :: V2=5 :: V
3=7
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),V1,N(B),V2,
N(C),V3):: NEXT J :: RETURN
```

I'll be referring back to this one as the negative duration method. Again, you can change the tempo by changing the value of D, but sometimes not as exactly as I would like. With this method, you will find that a series of the same note runs together into a single long note. To avoid this, use different harmony notes each time, or different volumes for V2 and V3.

There's no law that says the harmony has to be lower than the melody, so try changing N(B) to N(B)*2 or even N(B)*4 or do the same with N(C), or both. Or, use *X, add X=1 to line 110, and then in the middle of your music program you can switch by X=2 or X=4 (don't try 3!)

For a vibrato effect, we alternate a note with the same note multiplied by 1.01 -

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),V1,N(B),V2,
N(C),V3):: CALL SOUND(-4250,
N(A)*1.01,V1,N(B),V2,N(C),V3
):: NEXT J :: RETURN
```

For vibrato in the harmony rather than the melody, multiply N(C) or N(B), or both, by 1.01 instead - or multiply all three.

For a stronger vibrato, change the 1.01 to 1.02 or even 1.03. Of course, you can also multiply the harmony notes in both CALL SOUNDs by 2 or 4, as above. Or for a "chop" effect, multiply them in one CALL SOUND but not the other. The possibilities are almost endless!

For a tremolo, we alternate the volume rather than the frequency. Add X=3 to line 110 and use this routine -

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),V1,N(B),V2,
N(C),V3):: CALL SOUND(-4250,
N(A),V1+X,N(B),V2,N(C),V3)::
 NEXT J :: RETURN
```

You can vary the value of X as much as you want (V3+X can't total more than 30) for any amount of tremolo from a flutter to a wobble or a stutter, and you can put the +X after V1 or V2 or all three. You can even change it in the middle of your music, by X= whatever you want.

And you can multiply any or all by 1.01 for different combinations of vibrato and tremolo.

To enhance a note, play it twice in the CALL SOUND but multiply one of its voices by 1.01 -

```
110 D=4 :: V1=1 :: V2=5 :: V
3=7
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),V1,N(A)*1.0
1,V1,N(B),V2):: NEXT J :: RE
TURN
```

Of course, with this trick you can only have 2-part harmony, but you can choose to enhance the harmony rather than the melody.

Now, try combining the enhanced note with the vibrato and/or tremolo, for many more effects. For enriched vibrato, use N(A),V1,N(A)*1.01,V1 in the first CALL SOUND and N(A)*1.01,V1,N(A)*1.02, V1 in the second.

The bass notes do not go well with this method because interrupting them through a loop introduces a rattle, but the baritone works well and gives a unique reedy sound. To do this, place the note you want in the 3rd position, multiply it by 7.5, give it a volume of 30, and add the -4 noise at whatever volume you want. You can also combine this with other effects, for instance with vibrato

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),V1,N(B),V2,
N(C)*7.5,30,-4,V3)
1010 CALL SOUND(-4250,N(A)*1
.01,V1,N(B),V2,N(C)*7.5,30,-
4,V3):: NEXT J :: RETURN
```

Now for the real fun - the "piano" effects that we get by decreasing the volume gradually. This is the basic routine -

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),J+V1,N(B),J
+V2,N(C),J+V3):: NEXT J :: R
ETURN
```

Of course, with all of these you must also have that line 110 to define the duration and volume.

If you want a little more percussion in your piano, try this -

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),J*1.5,N(B),
J*1.5,N(C),J*1.5):: NEXT J :
: CALL SOUND(-4250,N(A),15,N
(B),15,N(C),15):: RETURN
```

And, of course, all those tricks we learned above - vibrato, tremolo, baritone, enhanced, high harmony, chop - can also be used with piano. This will give you the vibrato -

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),J+V1,N(B),J
+V2,N(C),J+V3):: CALL SOUND(
-4250,N(A)*1.01,J+V1,N(B),J+
V2,N(C),J+V3):: NEXT J :: RE
TURN
```

And an increasing tremolo -

```
1000 FOR J=1 TO T*D :: V=J*2
 :: CALL SOUND(-4250,N(A),J,
N(B),J,N(C),J):: CALL SOUND(
-4250,N(A),V,N(B),V,N(C),V):
: NEXT J :: RETURN
```

And just one more, the "reverse piano" with an increasing volume -

```
1000 FOR J=T*D TO 1 STEP -1
:: CALL SOUND(-4250,N(A),J+V
1,N(B),J+V2,N(C),J+V3):: CAL
L SOUND(-4250,N(A),J+V1,N(B)
,J+V2,N(C),J+V3):: NEXT J ::
 RETURN
```

By the time you get through exploring all the possible combinations of those, you should have a hundred ways of making music. Save each one you like, complete with line 110, in merge format, so you can try them all with each piece of music you create.

I had intended this to be the last part of this series, but I still haven't told you about autochording, so there will have to be one more.

TIUG

CONSOLETATION ZONE

This is the letters and help page. Lets hear from you. We want to hear what you say. So please write to your editor.

ARTICLE by Stephen Shaw

Hi everyone. Not much written by me this issue due to family illness*, but a brief bit for you...

We will start with another graphics program, this time a RECURSIVE one. Our versions of Basic do not support recursion, but as long as there is enough memory we can stack up our subroutines, which is how this program operates.

Every time you GOSUB, the computer remembers where you left from UNTIL it returns, so lots of GOSUBS without any returns can end up with a MEMORY FULL error, but with a small program like this, we can stack up the GOSUBS far enough to produce a perfectly adequate recursive program:

```
1 ! KOCH CURVE
2 ! recursive pattern
3 ! written for TI ExBas plus The Missing Link
4 ! by stephen shaw march 1993
5 ! from "fractals for the classroom" by peitgen,jurgens, and saupe.
Springer-Verlag.
6 !
7 ! after Helge von Koch, Arkiv for Matematik, 1904
8 !
100 CALL LINK("CLEAR"):: RANDOMIZE
110 ! CALL LINK("PRINT",41,12,"Input peak offset
(suggest 0.29)"):: CALL LINK("INPUT",71,81,R)
120 R=RND*.52 ! 0<R<1 -rem out if using line 110.
130 CALL LINK("CLEAR"):: R$="R="&STR$(R):: CALL LINK("P
RINT",140,100,R$)
140 L=5 :: XL(L)=10 :: XR(L)=220+10 :: YL(L),YR(L)=120
150 GOSUB 190
160 CALL LINK("PRINT",170,20,"DONE-press a key")
170 CALL KEY(5,A,B):: IF B<1 THEN 170
180 END
190 IF L>1 THEN 220
200 CALL LINK("LINE",YL(1),XL(1),YR(1),XR(1))
210 GOTO 390
220 L=L-1
230 XL(L)=XL(L+1):: YL(L)=YL(L+1)
240 XR(L)=.333*XR(L+1)+.667*XL(L+1)
250 YR(L)=.333*YR(L+1)+.667*YL(L+1)
260 GOSUB 190
270 XL(L)=XR(L):: YL(L)=YR(L)
280 XR(L)=.5*XR(L+1)+.5*XL(L+1)-R*(YL(L+1)-YR(L+1))
290 YR(L)=.5*YR(L+1)+.5*YL(L+1)+R*(XL(L+1)-XR(L+1))
300 GOSUB 190
310 XL(L)=XR(L):: YL(L)=YR(L):: XR(L)=.667*XR(L+1)+.333*XL(L+1)
320 YR(L)=.667*YR(L+1)+.333*YL(L+1)
330 GOSUB 190
340 REM RIGHT BRANCH
350 XL(L)=XR(L):: YL(L)=YR(L):: XR(L)=XR(L+1):: YR(L)=YR(L+1)
360 GOSUB 190
370 L=L+1
380 RETURN
390 RETURN
400 RETURN ! do we need all these? check and delete if required!
```

PROGRAMMING BASIC...

Way back in TI*MES Issue 34, Autumn 1991, I gave a program that would add fractions, but not reduce the result (Listing One, below), and asked if you could better it.

* My mother was in her final weeks of cancer, it hit me quite hard.ss

I also, in the same issue, gave a more complex program that would reduce the result (Listing Number Two, below).

These entries created the usual lack of response from the UK, but have stirred up quite a response from our overseas friends.

In MICROpendium March 1993 issue, Dean S Mah of Red Deer, Alberta gave some modifications to listing one- see listing three below, to be read in conjunction with list one.

Meanwhile one of our leading programmers has been attacking the problem from several angles- see listings four onwards.

Consider the problem. eg to add 11/13 to 17-32 etc and display the result as a fraction (some fractions cannot be displayed too well as a decimal!).

Try to solve the problem.

Look at the various attempts below. Which program is the easiest to follow? Which program uses the least memory? And which program gives the fastest result? Which of these three options would you prefer?

Now for the listings, in 80 column format to reduce space...

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

LISTING ONE- THE ORIGINAL:

```
1 ! ADDING FRACTIONS
2 ! result is not reduced       eg 12/16 would usually
be shown as 3/4
100 CALL CLEAR
110 DISPLAY AT(10,5):"--- + --- = ---"
120 ACCEPT AT(9,5)SIZE(3)VALIDATE(DIGIT):A
130 ACCEPT AT(11,5)VALIDATE(DIGIT)SIZE(3):B
140 ACCEPT AT(9,11)SIZE(3)VALIDATE(DIGIT):C
150 ACCEPT AT(11,11)SIZE(3)VALIDATE(DIGIT):D
160 GOSUB 230
161 ! PRINTER CHECK: WHEN YOU SEE #### USE SHIFT 3
170 DISPLAY AT(9,16):USING "####":N
180 DISPLAY AT(11,16):USING "####":L
190 DISPLAY AT(14,1):"ENTER KEY FOR ANOTHER"
200 DISPLAY AT(1,1):"NORMAL RESULT=";A/B+C/D
210 ACCEPT AT(24,12):A$ :: GOTO 100
220 STOP
230 FOR X=2 TO B*D
240 IF INT(X/B)<X/B THEN 260
250 IF INT(X/D)=X/D THEN 270
260 NEXT X
270 L=X
280 N=INT(L/B)*A+INT(L/D)*C
290 RETURN
```

LISTING TWO- reduces answer...

```
100 ! FRACTIONAL + AND -
110 ! R CALDWELL JAN 91
120 ! FOR TI99/4A BY
130 ! S SHAW APR 91
140 DIM Q(102)
150 DISPLAY AT(1,1)ERASE ALL:"FRACTIONAL + & -"
160 DISPLAY AT(7,1):"---+---"
170 DE,X=1
```

```
180 ACCEPT AT(6,2)VALIDATE(DIGIT,"+-")SIZE(4):N(X)
190 IF N(X)=0 THEN 310
200 ACCEPT AT(8,2)VALIDATE(DIGIT)SIZE(4):D(X)
210 IF D(X)=0 THEN 310
220 DP(X)=N(X):: DE=DE*D(X)
230 CALL HCHAR(6,1,32,12)
240 CALL HCHAR(8,1,32,12)
250 DISPLAY AT(5,1):"PLUS:"
260 DISPLAY AT(3,1):"ENTER 0 TO TOTAL"
270 X=X+1 :: DISPLAY AT(7,15):"ITEM";X :: DISPLAY AT(8,15):"-MAX 10-"
280 IF X=11 THEN 310
290 GOTO 180
300 REM
310 F,X=X-1
320 FOR S=1 TO F :: FOR J=1 TO F
330 Z=P+1 :: K=S+J-1
340 P(S,J)=K-F*INT(K/F):: IF P(S,J)=0 THEN P(S,J)=F
350 Q(Z)=P(S,J):: NEXT J :: NEXT S
360 FOR X=1 TO F :: Y=F*X-F+1 :: FOR C=1 TO F-1
370 Y=Y+1 :: DP(X)=DP(X)*D(Q(Y)):: NEXT C
380 NU=NU+DP(X):: NEXT X :: DD=2
390 DISPLAY AT(10,1):"SUM IS:" :: DISPLAY AT(11,5):NU;"/";DE
400 IF NU/DD=INT(NU/DD)AND DE/DD=INT(DE/DD)THEN NU=NU/DD ::
DE=DE/DD :: GOTO 400
410 DD=DD+2+(DD=2)
420 IF ABS(NU)>DE THEN A=ABS(NU)ELSE A=DE
430 IF DD<=SQR(A)THEN 400
440 IF NU/DE=INT(NU/DE)THEN 540
450 IF NU>0 THEN WN=INT(NU/DE):: NU=NU-WN*DE :: PN=1
460 IF NU<0 THEN WN=INT(NU/DE)+1 :: NU=ABS(NU-WN*DE):: PN=2 ::
IF WN=0 THEN NU=-NU
470 REM CHAR ##### IS SHIFT 3
480 DISPLAY AT(13,5):USING "#######":NU
490 IF PN=1 THEN DISPLAY AT(14,1):WN :: NU=NU+WN*DE
500 IF PN=2 THEN DISPLAY AT(14,1):WN :: IF WN<0 THEN NU=WN*DE-NU
510 PN=0
520 DISPLAY AT(14,5):"-------"
530 DISPLAY AT(15,5):USING "#######":DE
540 DISPLAY AT(18,4):NU/DE
550 DISPLAY AT(24,4):"ANY KEY FOR ANOTHER"
560 CALL KEY(1,G,H):: IF H<1 THEN 560
570 RUN
580 END
```
==

LISTING THREE- modifications to LIST ONE,
from Micropendium March 1993:

```
165 CALL REDUCE(N,L)
300 SUB REDUCE(A,B)
310 X=A :: Y=B :: IF X<Y THEN TEMP=X :: X=Y :: Y=TEMP
320 CALL MOD(X,Y,REMAIN)
330 IF REMAIN>0 THEN X=Y :: Y=REMAIN :: GOTO 320
340 A=A/Y :: B=B/Y :: SUBEND
350 SUB MOD(A,B,REMAIN)
360 REMAIN=A-INT(A/B)*B :: SUBEND
370 END
```

LIST FOUR- WITH MACHINE CODE LINK (machine code source at end):

```
90 CALL INIT :: CALL LOAD("DSK1.FRED/O")
100 CALL CLEAR
110 DISPLAY AT(10,5):"--- + --- = ---"
120 ACCEPT AT(9,5)SIZE(3)VALIDATE(DIGIT):A
```

```
130 ACCEPT AT(11,5)SIZE(3)VALIDATE(DIGIT):B
140 ACCEPT AT(9,11)SIZE(3)VALIDATE(DIGIT):C
150 ACCEPT AT(11,11)SIZE(3)VALIDATE(DIGIT):D
160 GOSUB 230 :: CALL LINK("FRED",N,L)
170 DISPLAY AT(9,16):USING "####":N ! # IS SHIFT 3
180 DISPLAY AT(11,16):USING "####":L
190 DISPLAY AT(14,1):"ENTER KEY FOR ANOTHER"
200 DISPLAY AT(1,1):"NORMAL RESULT=";A/B+C/D
210 ACCEPT AT(24,12):A$ :: GOTO 100
220 STOP
230 FOR X=2 TO B*D
240 IF INT(X/B)<X/B THEN 260
250 IF INT(X/D)=X/D THEN 270
260 NEXT X
270 L=X
280 N=INT(L/B)*A+INT(L/D)*C
290 RETURN


----machine code source:

* FRED/S
* TO REDUCE FRACTIONS
* CODE BY B. HARRISON
* PUBLIC DOMAIN
*
DEF    FRED
NUMASG EQU  >2008
NUMREF EQU  >200C
XMLLNK EQU  >2018
CFI    EQU  >12B8
CIF    EQU  >20
       CLR  R0
       LI   R1,1
       BLWP @NUMREF
       BLWP @XMLLNK
       DATA CFI
       MOV  @>834A,@NUMER
       MOV  @>834A,@TNUM
       INC  R1
       BLWP @NUMREF
       BLWP @XMLLNK
       DATA CFI
       MOV  @>834A,@DENOM
       MOV  @>834A,@TDEN
       C    @DENOM,@NUMER
       JLT  IMPROP
       MOV  @NUMER,@LIMIT
       JMP  PROPER
IMPROP MOV  @DENOM,@LIMIT
PROPER
       INCT R0
RPT    C    R0,@LIMIT
       JGT  ANSWER
TRIAL  MOV  @NUMER,R3
       CLR  R2
       MOV  @DENOM,R5
       CLR  R4
       DIV  R0,R2
       MOV  R3,R3
       JNE  NXT
       DIV  R0,R4
       MOV  R5,R5
       JNE  NXT
       MOV  R2,@TNUM
       MOV  R4,@TDEN
NXT    INC  R0
       JMP  RPT
ANSWER
       CLR  R0
       LI   R1,1
       MOV  @TNUM,@>834A
       BLWP @XMLLNK
       DATA CIF
       BLWP @NUMASG
       INC  R1
       MOV  @TDEN,@>834A
       BLWP @XMLLNK
       DATA CIF
       BLWP @NUMASG
       LWPI >83E0
       B    @>6A
WS     BSS  32
NUMER  DATA 0
DENOM  DATA 0
TNUM   DATA 0
TDEN   DATA 0
LIMIT  DATA 0
END
```

==

LIST FIVE- moving back to pure XB-
amend the basic portion of list 4 as below:

```
1 ! ADD FRACTIONS WITH REDUCTION
2 ! LINE 290 REMOVED & NEW LINES 300-360 ADDED BY B. HARRISON
3 ! ORIGINAL PROGRAM BY S. SHAW
290 REM
300 LIM=MIN(N,L):: TN=N :: TL=L
310 FOR Y=2 TO LIM
320 IF N/Y<>INT(N/Y)THEN 350
330 IF L/Y<>INT(L/Y)THEN 350
340 TN=N/Y :: TL=L/Y
350 NEXT Y
360 N=TN :: L=TL :: RETURN
```

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

LISTING SIX- modifications to listing 4:

```
90 ON WARNING NEXT
290 REM
300 TN=N :: TL=L
310 FOR Y=2 TO MIN(N,L)
320 IF N/Y>INT(N/Y)THEN 350
330 IF L/Y=INT(L/Y)THEN TN=N/Y :: TL=L/Y
350 NEXT Y :: RETURN
```

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

LISTING   - modifications to listing 4:

```
90 ON WARNING NEXT
290 REM
300 REM
310 FOR Y=2 TO MIN(N,L)
320 IF N/Y>INT(N/Y)THEN 350
```

continued....

```
330 IF L/Y=INT(L/Y)THEN N=N/Y :: L=L/Y :: GOTO 310
350 NEXT Y :: RETURN
========================================
LISTING NINE- complete listing:
2 ! MODIFIED FOR REDUCTION AND SPEED ENHANCEMENT BY BRUCE HARRISON
4 ! VERSION ADFR6 OF 1 APRIL 93
90 ON WARNING NEXT
100 DISPLAY AT(10,5)ERASE ALL:"--- + --- = ---"
110 ACCEPT AT(9,5)SIZE(3)VALIDATE(DIGIT):A
120 ACCEPT AT(11,5)SIZE(3)VALIDATE(DIGIT):B :: IF B=0 THEN 120
130 ACCEPT AT(9,11)SIZE(3)VALIDATE(DIGIT):C
140 ACCEPT AT(11,11)SIZE(3)VALIDATE(DIGIT):D :: IF D=0 THEN 140
150 FOR L=MAX(B,D)TO B*D STEP MAX(B,D)
160 IF INT(L/B)<L/B THEN 180
170 IF INT(L/D)=L/D THEN 190
180 NEXT L
190 N=L/B*A+L/D*C
200 FOR Y=2 TO MIN(N,L)
210 IF N/Y>INT(N/Y)THEN 230
220 IF L/Y=INT(L/Y)THEN N=N/Y :: L=L/Y :: GOTO 200
230 NEXT Y
240 DISPLAY AT(9,16):USING "####":N
250 DISPLAY AT(11,16):USING "####":L
260 DISPLAY AT(14,3):"PRESS ENTER FOR ANOTHER"
270 DISPLAY AT(1,1):"NORMAL RESULT=";A/B+C/D
280 CALL KEY(0,K,S):: IF S<>1 THEN 280 ELSE IF K=13 THEN 100
====================
```

There you have it. There are almost always MANY ways to tackle ANY programming problem. Some will be better than others because they are easier to follow OR use up less memory OR work faster. The programmer must decide his priorities and program accordingly.

So called programmers using macroassemblers with vast routine libraries already written tend to end up with programs which are hard to follow, hard to modify, are slow, and use up lots of memory, which is why PC users now tend to go for machines with 4MB of memory running at 32Mhz or more! You don't need all that if you know how to program.
And you can avoid using up 29k just to do a PRINT "HI"!

PRINTER INTERFACE FOR CONSOLE ONLY ?

BY GARY SMITH.

Where did the idea come from?

It all stems from the AGM in Derby. People were a little thin on the ground but there was quite an amount of new and interesting items. When the meeting had started it opened my eyes to a few things. Many good points were brought out and I found out that over 52% of people in the group were unexpanded which came as a shock.

I put some thought into this on the train on the way home and remembered when I had to pay out for the expansion box. It is quite expensive but never the less worth while and I have never looked back since. The other option is to build your own expansion box which I believe that Dave Hewitt did quite sucessfully. The problem here is that if you've never tried anything like this before it could be a bit tricky.

The title of this article may be a bit missleading. To get this interface working you will need either TI Extended Basic with 32K or Mini Memory (how many times have you heard that!) It has to be said either or both of these are a great plus for the console, it opens it up for so much more like machine code, larger programs and printer interfaces! There is no excuse at the moment as I believe that there are copies of both TI Extended Basic and Mini Memory in the module library!

For those of you without the 32K there are a few things you can do. I can recommend the Matchbox expansion done by Phil West and Bernie Elsner which works perfectly with every thing that I have tried. Since technology has moved on this expansion can be done with just one chip! The other option is to buy one which fits in the side expansion slot on the console. Mike Goddard may have one or two of these. If not, let me know as I am quite willing to do the matchbox expansion or build the add on card for you.

Down to the real business!

How many people are sitting there with a tape machine and a word processor thinking a printer would be very handy but there is no way I'm going to spend $$$$? How many people have a printer or could beg/borrow one? I imagine this would be quite a high number.

You're probably thinking 'not more vapourware' or 'this sounds like its going to cost a packet!'. You would be wrong on both accounts. I expect this to cost about :

1.50	for connectors
0.72	for the chips
0.80	about for transistors
?.??	for wire/solder/etc

Total --- 5-6 pounds!!!!!!

No one can tell me that this will break the bank! This interface will be connected to the tape port on the console and will use the two motor controls to provide a data/clock serial interface. This is then changed to centronics standard and sent out. This means that any printer which can connect to the 16 way connector on the RS-232 card will connect to this!

Due to an inactive limb keeping me from work I have been able to design this and partly build it. It will need some drivers to be written which will be mostly basic apart from the code which operates the motors.

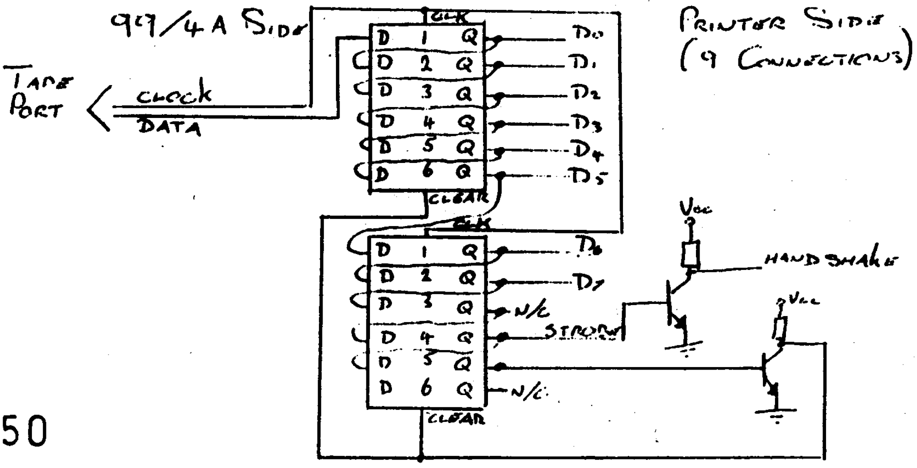
This motor control code will come from an article by Mike Goddard back in 1988 where he controls a robot by the tape port. The rest of the program will be splitting the word processed file into separate characters and converting each character to ASCII code and shifting this out the port with a few hand shaking bits in front to strobe the data into the printer.

I would also think it should be possable to print out programs this way and also graphics but these would need a more complex driver which is a little over my head but maybe Mark or Richard could get their minds around this one.

Here's a quick run down as to how it works:

It uses two D-Type flip-flops (74LS174 type) as serial to centronic convertors. These two chips give 12 bit resolution in total. Eight of these are the character data. One bit is used as a stobe to send the data to the printer and another bit is used to reset the interface ready to send another character to the printer. The code sent would look something like :
10 01000001 -- 10 is handshake -- 01000001 is 'A'.

I would be interested in any comments you may have about this, good or bad! Heres the preliminary circuit if anyone fancies trying it out.



HIGHLIGHTING
By Earl Raguse

I saw a demonstration that allowed one to switch the foreground colors of certain characters to make them stand out from others like O vs 0, and 1 vs l, or for trouble shooting of bad typing, something I do real well. Aha, you say, I can do that with CALL COLOR. True, but its not permanent. I don't like having to embed trouble shooting routines in my programs if there is an easier way.

The following program called HIGHLIGHT makes permanent foreground/background color changes and can be controlled ON and OFF at will. Once executed, the program can be deleted with NEW before you start entering a new program. I sometimes put this in my LOAD program, its easy to turn off if you don't want it. I found the basic program idea in the Tacoma 99ers Newsletter of December 1987, the article was by Joe Nolan, who credits Harry Wilhelm of the Twin Tiers UG with the original idea. I don't have any idea how much evolution has gone on, but I added my two cents also.

Lines 130 and 140 do all the work, and if you wish to transfer this effect to one of your own programs, that's all you need. The following tells you how you can change these lines to suit your needs. If you study it a bit, you can see the potential for other purposes.

In line 130,
===========
(1) Change the eighth number, from the address, 17, to the number of the first character set you want to change PLUS 15, The current program is 15+2=17 for character set 2.

(2) Change the eighth number after that, 3, to the number of character sets to change. The current program is 3 for character sets 2, 3 and 4.

In line 140,
===========
(1) Load a number, (in this case 244) for each character set to be changed. That number is computed as (16*(FG-1)) + (BG-1) where FG and BG are the Foreground and Background color numbers as defined in the XBASIC manual. Each character set could have a different combinaton of colors. The program as written is for all characters white on blue, ie (16*(16-1)) + (5-1) = 244.

(2) The effect is turned ON by CALL LOAD(-31804,63) and OFF by CALL LOAD (-31804,0). This can be done either in a program or from the keyboard. I added the lines 150 and 160 for easy control of the effect on or off. These can be deleted if not wanted.

```
100 ! SAVE DSK1.HIGHLIGHT
110 !By Joe Nolan, Tacoma 99 ers UG Newsletter Dec 87, Original idea by
Harry Wilhelm of Twin TiersUG
120 !Modified by E Raguse UGOC 1/87
130 CALL INIT :: CALL LOAD
(16128,2,224,38,0,2,0,8,17,2
,1 ,63,36,2,2,0,3,4,32,32,
36,2,224,131,192,3,128)
140 CALL LOAD(16164,244,244,
244 ::CALL LOAD(-31804,63)
150 PRINT "TURN IT OFF? PRES
S SPACE,   ELSE ANY"
160 CALL KEY(0,K,S):: IF S=0
 THEN 160 ELSE IF K<>32 THEN
 END ELSE CALL LOAD(-318
04,0)
```

===

TIGERCUB TIPS #12

Copyright 1984 Tigercub Software
156 Collingwood Ave.,
Columbus Ohio 43213

If you have taken a course in computer programming, one of your homework assignments was probably to write a program that would find all the possible combinations of letters in a 5-letter word.

The following version can handle words of 3 to 6 letters, lists the combinations alphabetically, eliminates duplicates (when the word has two of the same letter), does not require a DIM statement, and is fast. It also works with numbers. If you work those scrambled-word puzzles in the newspapers, you'll find it handy.

```
100 CALL CLEAR :: PRINT TAB(
5);"TIGERCUB ANAGRAMMER": :!
by Jim Peterson
110 INPUT "TYPE A 3-,4-,5- O
R 6-LETTER WORD  ":A$ :: W=L
EN(A$):: IF (W<3)+(W>6)THEN
110
120 PRINT :: FOR J=1 TO W ::
 B$(J)=SEG$(A$,J,1):: NEXT J
 :: FOR J=2 TO W :: IF B$(J)
>=B$(J-1)THEN 160
130 T$=B$(J):: FOR L=J-1 TO
1 STEP -1 :: B$(L+1)=B$(L)
140 IF B$(L-1)>=T$ THEN 150
:: B$(L)=T$ :: GOTO 160
150 NEXT L
160 NEXT J
170 FOR A=1 TO W :: FOR B=1
TO W :: IF B=A THEN 340
180 FOR C=1 TO W :: IF (C=A)
+(C=B)THEN 330
190 IF W=3 THEN 250
200 FOR D=1 TO W :: IF (D=A)
+(D=B)+(D=C)THEN 320
210 IF W=4 THEN 260
220 FOR E=1 TO W :: IF (E=A)
+(E=B)+(E=C)+(E=D)THEN 310
230 IF W=5 THEN 270
240 FOR F=1 TO W :: IF (F=A)
+(F=B)+(F=C)+(F=D)+(F=E)THEN
300 ELSE 280
250 W$=B$(A)&B$(B)&B$(C):: I
F W$<=V$ THEN 330 ELSE 290
260 W$=B$(A)&B$(B)&B$(C)&B$(
D):: IF W$<=V$ THEN 320 ELSE
290
270 W$=B$(A)&B$(B)&B$(C)&B$(
D)&B$(E):: IF W$<=V$ THEN 31
0 ELSE 290
280 W$=B$(A)&B$(B)&B$(C)&B$(
D)&B$(E)&B$(F):: IF W$<=V$ T
HEN 310
290 PRINT W$&" ";:: G=G+1 ::
 V$=W$ :: ON W-2 GOTO 330,32
0,310,300
300 NEXT F
310 NEXT E
320 NEXT D
330 NEXT C
340 NEXT B
350 NEXT A
360 PRINT : :"  ";G;"TOTAL C
OMBINATIONS.": : :: G=0 :: V
$="" :: GOTO 110
```

And still another automatic music-maker. This one doodles around the keyboard in the key of A, with autmatic bass accompaniment.

```
100 RANDOMIZE
110 DIM N(30)
120 F=220
130 FOR J=0 TO 36
140 X=X+1+(X=12)*12
150 IF (X=2)+(X=5)+(X=7)+(X=
10)+(X=12)THEN 180
160 Y=Y+1
170 N(Y)=INT(F*1.059463094^J
)
180 NEXT J
190 K=8
200 K=K-INT(5*RND+1)+INT(5*R
ND+1)+(K>21)*2-(K<1)*2
210 IF (K<1)+(K>21)THEN 200
220 CALL SOUND(-999,N(K),0,N
(K)*2,0,N(K)*3.75,30,-4,5)
230 GOTO 200
================
100 CALL CLEAR
110 REM - programmed by Jim
Peterson May 20, 1984
120 PRINT "TIGERCUB MAGIC SQ
UARE MAKER": :" A magic squa
re is a conse-":"cutive seri
es of numbers":"arranged in
```

```
a square in such"
130 PRINT "a way that each h
orizontal":"row, vertical ro
w, and long":"diagonal row w
ill add up to":"the same tot
al.": :
140 PRINT " This little prog
ram will":"create an odd-ord
er magic":"square of any des
ired size,":"starting with a
any desired":"number.": :
150 PRINT " Squares of 3,5,7
 or 9 size":"will be printed
 on the":"screen. The progra
m can be":"modified to outpu
t larger"
160 PRINT "sizes to a printe
r.": :
170 INPUT "SIZE OF SQUARE?(o
dd number) ":S
180 IF (S<3)+(S/2=INT(S/2))T
HEN 170
190 INPUT "STARTING NUMBER?
":SN
200 N=SN-1
210 CALL CLEAR
220 DIM G(31,31)
230 R=1
240 C=INT(S/2)+1
250 N=N+1
260 IF N=S^2+SN THEN 450
270 G(R,C)=N
280 IF (R-1=0)+(C+1>S)THEN 3
50
290 IF G(R-1,C+1)<>0 THEN 33
0
300 R=R-1
310 C=C+1
320 GOTO 250 ! # is shift 3
330 R=R+1
340 GOTO 250
350 IF (R=1)*(C=S)THEN 400
360 IF (R>1)*(C=S)THEN 420
370 R=S
380 C=C+1
390 GOTO 250
400 R=2
410 GOTO 250
420 R=R-1
430 C=1
440 GOTO 250
450 IF (LEN(STR$(SN+S^2))+1)
*S>28 THEN 530
460 FOR R=1 TO S
470 FOR C=1 TO S
480 PRINT STR$(G(R,C));" ";
490 NEXT C
500 PRINT : :
510 NEXT R
520 GOTO 550
530 PRINT "TOO LARGE FOR SCR
EEN."
540 REM - ADD PRINTER ROUTIN
E HERE - # is shift 3
550 PRINT : :"PRESS ANY KEY
TO CHECK"
560 CALL KEY(0,K,ST)
570 IF ST=0 THEN 560
580 FOR R=1 TO S
590 FOR C=1 TO S
600 X=X+G(R,C)
610 NEXT C
620 PRINT "ROW #";STR$(R);"
=";X
630 X=0
640 NEXT R
650 FOR C=1 TO S
660 FOR R=1 TO S
670 X=X+G(R,C)
680 NEXT R
690 PRINT "COLUMN #";STR$(C)
;" =";X
700 X=0
710 NEXT C
720 R=1
730 C=1
740 FOR J=1 TO S
750 X=X+G(R,C)
760 R=R+1
770 C=C+1
780 NEXT J
790 PRINT "RIGHT DIAGONAL=";
X
800 X=0
810 R=1
820 C=S
830 FOR J=1 TO S
840 X=X+G(R,C)
850 R=R+1
860 C=C-1
870 NEXT J
880 PRINT "LEFT DIAGONAL=";X
890 END
```

TIGERCUB TIPS #13

I'm told that someone actually found a practical use for my number-scrambling rutine, so here is an expanded version.

It will scramble any sequence beginning with 1 and ending with any number less than 256 or any number greater than 256 which is evenly divisible by any number less than 256 and greater than 1, within the limits of computer memory.

In Extended Basic with Memory Expannsion, the limit is about 10,700; if you reformat it to Basic and run it bare bones, you might get close to 13,000.

```
100 CALL CLEAR :: OPEN #1:"P
IO",OUTPUT
110 INPUT "HIGHEST NUMBER? "
:HN :: IF HN<256 THEN TN=HN
:: XX=1 :: GOTO 150
120 FOR TN=255 TO 2 STEP -1
:: IF HN/TN=INT(HN/TN)THEN 1
40
130 NEXT TN :: PRINT HN;"IS
NOT DIVISIBLE BY":"ANYTHING
LESS THAN 256  - ":"CANNOT U
SE" :: GOTO 110
140 XX=HN/TN
150 DIM M$(50)
160 CALL CLEAR :: FOR J=1 TO
 TN :: M$(1)=M$(1)&CHR$(J)::
 NEXT J :: FOR J=1 TO XX ::
M$(J)=M$(1):: NEXT J :: FOR
J=1 TO HN :: TT=1+INT((J-1)/
255)
170 RANDOMIZE :: X=INT(XX*RN
D+1):: IF LEN(M$(X))=0 THEN
170 :: ! # is shift 3
180 Y=INT(LEN(M$(X))*RND+1)
190 PRINT #1:ASC(SEG$(M$(X),
Y,1))+TN*(X-1);
200 M$(X)=SEG$(M$(X),1,Y-1)&
SEG$(M$(X),Y+1,LEN(M$(X)))::
 NEXT J
================
```

Here's a little routine you can use to jazz up your title screen or text.

```
100 CALL CLEAR
110 DATA "THIS IS A DEMONSTR
ATION","OF THE","TIGERCUB SO
FTWARE","TWO-WAY PRINT ROUTI
NE"
112 FOR T=1 TO 4
113 READ M$
120 IF LEN(M$)/2=INT(LEN(M$)
/2)THEN 135
130 M$=M$&" "
131 GOTO 140
135 M$=M$&"  "
140 L=LEN(M$)
150 C=16-L/2
160 FOR J=L/2 TO 1 STEP -1
170 CALL HCHAR(10+T*2,C+J,AS
C(SEG$(M$,J,1)))
180 CALL HCHAR(10+T*2,16+L/2
-J,ASC(SEG$(M$,L-J,1)))
190 NEXT J
200 NEXT T
=================
```

Did you ever go through your checkbook 5 times in order to add up your gas bill, then your electric bill, etc.? With this little handy- dandy, you can do it all in one pass.

```
100 CALL CLEAR
110 REM -  ADDER-UPPER by Ji
m Peterson
120 A$="ABCDEFGHIJKLMNOPQRST
UVWXYZ"
130 DIM C$(26),T(26)
140 PRINT "          ADDER-UPP
ER": : :
150 PRINT "WITH THIS PROGRAM
 YOU CAN GO THROUGH YOUR CHE
CKBOOK, OR ANYTHING ELSE, AN
D ADD UP   AMOUNTS IN SEVERA
L CATE-"
160 PRINT "GORIES ALL AT ONE
 TIME.": :
170 PRINT " FIRST, LIST THE
CATEGORIES":"YOU WANT TO ADD
 UP.":" TYPE 'END' WHEN FINI
SHED.": :
180 PRINT " NEXT, ENTER THE
CATEGORY":"CODE AND AMOUNT F
OR EACH":"BILL."
190 PRINT : :"WHEN YOU HAVE
ENTERED ALL":"THE BILLS, TYP
E =": :
200 N=N+1
210 PRINT "CATEGORY #";N
220 INPUT "              ":C$(N
)
230 IF C$(N)="END" THEN 340
240 W$=SEG$(C$(N),1,1)
250 IF POS(A$,W$,1)<>0 THEN
290
260 PRINT :"CODE LETTER ";W$
;" ALREADY  USED - PICK A CO
DE LETTER."
270 INPUT W$
280 GOTO 250
```

```
290 X=POS(A$,W$,1)
300 A$=SEG$(A$,1,X-1)&SEG$(A
$,X+1,LEN(A$))
310 X$=X$&W$
320 PRINT :"CODE LETTER FOR
";C$(N);" WILL BE ";W$: :
330 GOTO 200
340 C$(N)=""
350 N=N-1
360 X$=X$&"="
370 IF FLAG=1 THEN 420
380 FLAG=1
390 PRINT : :"READY TO START
 - ": : :
400 PRINT "WHEN FINISHED, TY
PE =": :
410 INPUT "DO YOU WANT TO VE
RIFY EACH  INPUT? ":V$
420 PRINT :"CODE (";X$;")"
430 INPUT Q$
440 IF Q$="=" THEN 600
450 IF POS(X$,Q$,1)<>0 THEN
510
460 PRINT "THAT IS NOT ONE O
F THE CODES": :
470 INPUT "IS IT A NEW CATEG
ORY?(Y/N) ":Q$
480 IF SEG$(Q$,1,1)<>"Y" THE
N 420
490 X$=SEG$(X$,1,LEN(X$)-1)
500 GOTO 200
510 Y=POS(X$,Q$,1)
520 INPUT "AMOUNT ?":A
530 IF SEG$(V$,1,1)="N" THEN
 580
540 PRINT :C$(Y);A: :
550 INPUT "CORRECT? (Y/N)":L
$
560 IF SEG$(L$,1,1)="Y" THEN
 580
570 IF SEG$(L$,1,1)="N" THEN
 420 ELSE 550
580 T(Y)=T(Y)+A
590 GOTO 420
600 FOR J=1 TO N
610 PRINT :C$(J);T(J)
620 TT=TT+T(J)
630 NEXT J
640 PRINT :"GRAND TOTAL OF A
LL IS";TT
650 END
```

===========

And, did you ever wish that you could make numbers smaller, so that you could squeeze more of them onto a chart or graph? The problem is that resolution is so poor, at least on my TV screen, but maybe you'll find a use for this.

```
100 REM - NUMBER SCRUNCHER -
 programmed by Jim Peterson
110 CALL SCREEN(5)
120 FOR S=2 TO 14
130 CALL COLOR(S,15,1)
140 NEXT S
150 CALL CLEAR
160 RANDOMIZE
170 DATA 75557,22222,25127,6
1216,55571,74616,74757,71222
,75257,75711
180 FOR J=0 TO 9
190 READ C$
200 CH$(J)="00"&C$
210 NEXT J
220 CH=91
230 INPUT "NUMBER? ":RX
240 N$=STR$(RX)
250 IF LEN(N$)/2=INT(LEN(N$)
/2)THEN 270
260 N$="0"&N$
270 FOR J=1 TO LEN(N$)STEP 2
280 P1=VAL(SEG$(N$,J,1))
290 P2=VAL(SEG$(N$,J+1,1))
300 FOR T=1 TO 7
310 Z$=Z$&SEG$(CH$(P1),T,1)&
SEG$(CH$(P2),T,1)
320 NEXT T
330 CALL CHAR(CH,Z$)
340 Z$=""
350 P$=P$&CHR$(CH)
360 CH=CH+1
370 NEXT J
380 PRINT N$;" ";P$
390 P$=""
400 N$=""
410 GOTO 230
```

Almost OUT OF MEMORY.

Jim Peterson

TIPS FROM THE TIGERCUB
#33
Copyright 1986

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

Did you ever wonder how a computer sort actually worked? This program will let you actually see it in action. It will also show you the value being held in the temporary variable T$, and the total number of

swaps and comparisons made.
  Then you can change any of the variables and resort. Try AAA in the last position or ZZZ in the first. You will find that some of the fastest sorts are not so fast when a list is already almost in sequence.

```
100 CALL CLEAR :: CALL SCREE
N(16):: FOR SET=2 TO 9 :: CA
LL COLOR(SET,5,16):: NEXT SE
T :: ON WARNING NEXT :: RAND
OMIZE
110 DISPLAY AT(21,1)ERASE AL
L:">>>TIGERCUB SORT WATCHER<
<<": :"Wait, please - genera
ting":"random array...." ::
DIM A$(101),B$(101),ST(25,2)
120 FOR J=1 TO 100 :: FOR L=
1 TO 3 :: B$(J)=B$(J)&CHR$(I
NT(26*RND+65)):: NEXT L :: X
=J :: A$(X)=B$(X):: GOSUB 10
20 :: NEXT J
130 DISPLAY AT(3,1)ERASE ALL
:"(1) BUBBLE SORT": :"(2) SH
AKER SORT": :"(3) SWAP SORT"
: :"(4) SHUTTLE SORT": :"(5)
 EASY SORT"
140 DISPLAY AT(13,1):"(6) QU
ICK SORT": :"(7) RESORT SORT
": :"(8) SHELL SORT": :"(9)
RESERVED": :"Type number of
choice"
150 ACCEPT AT(21,23)VALIDATE
(DIGIT)SIZE(2)BEEP:K :: IF K
<1 OR K>10 THEN 150
160 DISPLAY AT(24,1):"Size o
f array? (10-100)" :: ACCEPT
 AT(24,25)VALIDATE(DIGIT)SIZ
E(3):G :: IF G<1 OR G>100 TH
EN 160
170 ON K GOSUB 230,300,430,5
00,550,650,850,910,25000 ::
DISPLAY AT(22,1):W;"SWAPS":C
;"COMPARISONS" :: C,W=0
180 DISPLAY AT(24,1):"Choose
 (1)Menu or (2)Resort" :: AC
CEPT AT(24,7)VALIDATE("12")S
IZE(1):Q :: IF Q=1 THEN 130
190 DISPLAY AT(24,1):"Change
 which position? 0" :: ACCEP
T AT(24,24)VALIDATE(DIGIT)SI
ZE(-3):P :: IF P=0 THEN 210
ELSE IF P<1 OR P>G THEN 190
200 DISPLAY AT(24,1):"Change
 to?" :: ACCEPT AT(24,12)SIZ
E(3):A$(P):: X=P :: GOSUB 10
20 :: GOTO 190
210 DISPLAY AT(22,1):" ":" "
 :: GOSUB 1010 :: N=G :: ON
K GOSUB 240,310,440,510,560,
660,860,930,25010 :: DISPLAY
 AT(22,1):W;"SWAPS":C;"COMPA
RISONS" :: C,W=0 :: GOTO 180
220 REM *BUBBLESORT*
230 CALL CLEAR :: GOSUB 980
240 FOR J=2 TO N :: C=C+1 ::
 IF A$(J)>=A$(J-1)THEN 260
250 T$=A$(J):: GOSUB 1050 ::
 A$(J)=A$(J-1):: X=J :: GOSU
B 1020 :: A$(J-1)=T$ :: X=J-
1 :: GOSUB 1020 :: W=W+1 ::
F=1
260 NEXT J :: C=C+1 :: IF F=
0 THEN 280
270 W=W+1 :: F=0 :: W=W+1 ::
 N=N-1 :: GOTO 240
280 RETURN
290 REM *SHAKERSORT*
300 CALL CLEAR :: GOSUB 980
310 W=W+1 :: L=1 :: W=W+1 ::
 R=N
320 W=W+1 :: F=0 :: FOR J=L
TO R-1 :: C=C+1 :: IF A$(J)<
=A$(J+1)THEN 340
330 T$=A$(J):: GOSUB 1050 ::
 A$(J)=A$(J+1):: X=J :: GOSU
B 1020 :: A$(J+1)=T$ :: X=J+
1 :: GOSUB 1020 :: W=W+1 ::
F=1
340 NEXT J :: C=C+1 :: IF F=
0 THEN 410
350 W=W+1 :: R=R-1 :: C=C+1
:: IF R=L THEN 410
360 W=W+1 :: F=0 :: FOR J=R
TO L+1 STEP -1 :: C=C+1 :: I
F A$(J)>=A$(J-1)THEN 380
370 T$=A$(J):: GOSUB 1050 ::
 A$(J)=A$(J-1):: X=J :: GOSU
B 1020 :: A$(J-1)=T$ :: X=J-
1 :: GOSUB 1020 :: W=W+1 ::
F=1
380 NEXT J :: C=C+1 :: IF F=
0 THEN 410
390 W=W+1 :: L=L+1 :: C=C+1
:: IF L=R THEN 410
400 GOTO 320
410 RETURN
420 REM *SWAPSORT*
430 CALL CLEAR :: GOSUB 980
440 FOR J=1 TO N-1 :: W=W+1
:: R=J :: FOR JJ=J+1 TO N ::
 C=C+1 :: IF A$(R)<=A$(JJ)TH
EN 460
450 W=W+1 :: R=JJ
460 NEXT JJ :: C=C+1 :: IF R
=J THEN 480
470 T$=A$(J):: GOSUB 1050 ::
 A$(J)=A$(R):: X=J :: GOSUB
1020 :: A$(R)=T$ :: X=R :: G
OSUB 1020
480 NEXT J :: RETURN
490 REM ***SHUTTLE SORT*****
500 CALL CLEAR :: GOSUB 980
510 FOR J=1 TO N-1 :: FOR JJ
=J TO 1 STEP -1 :: C=C+1 ::
IF A$(JJ)<=A$(JJ+1)THEN 530
:: T$=A$(JJ):: GOSUB 1050 ::
 A$(JJ)=A$(JJ+1):: X=JJ :: G
OSUB 1020
520 A$(JJ+1)=T$ :: X=JJ+1 ::
 GOSUB 1020 :: NEXT JJ
530 NEXT J :: RETURN
540 REM ****EASY SORT******
550 CALL CLEAR :: GOSUB 980
560 W=W+1 :: D=1
570 W=W+1 :: D=2*D :: C=C+1
:: IF D<=N THEN 570
580 W=W+1 :: D=INT(D/2):: C=
C+1 :: IF D=0 THEN 630
590 FOR J=1 TO N-D :: W=W+1
:: Y=J
600 W=W+1 :: Z=Y+D :: C=C+1
:: IF A$(Y)<=A$(Z)THEN 620 :
: T$=A$(Y):: GOSUB 1050 :: A
$(Y)=A$(Z):: X=Y :: GOSUB 10
20 :: A$(Z)=T$ :: X=Z :: GOS
UB 1020
610 W=W+1 :: Y=Y-D :: C=C+1
:: IF Y>0 THEN 600
620 NEXT J :: GOTO 580
630 RETURN
640 REM *QUICKSORT*
650 CALL CLEAR :: GOSUB 980
660 W=W+1 :: L=1 :: W=W+1 ::
 R=N :: W=W+1 :: T=0
670 T$=A$(INT((L+R)/2)):: GO
SUB 1050 :: W=W+1 :: J=L ::
W=W+1 :: JJ=R
680 C=C+1 :: IF A$(J)>=T$ TH
EN 710
690 W=W+1 :: J=J+1
700 GOTO 680
710 C=C+1 :: IF A$(JJ)<=T$ T
HEN 730
720 W=W+1 :: JJ=JJ-1 :: GOTO
 710
730 C=C+1 :: IF A$(J)<>A$(JJ
)THEN 760
740 C=C+1 :: IF J>=JJ THEN 7
60
750 W=W+1 :: J=J+1 :: GOTO 7
30
760 C=C+1 :: IF J>=JJ THEN 7
80
770 W=W+1 :: H$=A$(J):: A$(J
)=A$(JJ):: X=J :: GOSUB 1020
 :: A$(JJ)=H$ :: X=JJ :: GOS
UB 1020 :: GOTO 680
780 W=W+1 :: J=J+1 :: W=W+1
:: JJ=JJ-1 :: C=C+1 :: IF J>
=R THEN 800
790 W=W+1 :: T=T+1 :: W=W+1
:: ST(T,0)=J :: W=W+1 :: ST(
T,1)=R
800 W=W+1 :: R=JJ :: C=C+1 :
: IF L<R THEN 670
810 C=C+1 :: IF T=0 THEN 830
820 W=W+1 :: L=ST(T,0):: W=W
+1 :: R=ST(T,1):: W=W+1 :: T
=T-1 :: GOTO 670
830 RETURN
840 REM ***RESORT SORT******
850 CALL CLEAR :: GOSUB 980
860 FOR J=2 TO N :: C=C+1 ::
 IF A$(J)>=A$(J-1)THEN 900
870 T$=A$(J):: GOSUB 1050 ::
 FOR L=J-1 TO 1 STEP -1 :: A
$(L+1)=A$(L):: X=L+1 :: GOSU
B 1020
880 C=C+1 :: IF A$(L-1)>=T$
THEN 890 :: A$(L)=T$ :: X=L
:: GOSUB 1020 :: GOTO 900
890 NEXT L
900 NEXT J :: RETURN
910 REM *SHELLSORT*
920 CALL CLEAR :: GOSUB 980
930 W=W+1 :: M=N
940 W=W+1 :: M=INT(M/3)+1
950 FOR J=1 TO N-M :: FOR JJ
=J TO 1 STEP -M :: C=C+1 ::
IF A$(JJ)<=A$(JJ+M)THEN 970
:: T$=A$(JJ):: GOSUB 1050
960 A$(JJ)=A$(JJ+M):: X=JJ :
: GOSUB 1020 :: A$(JJ+M)=T$
:: X=JJ+M :: GOSUB 1020 :: N
EXT JJ
970 NEXT J :: C=C+1 :: IF M>
1 THEN 940 :: RETURN
980 REM *RENEW ARRAY*
990 FOR J=1 TO G :: A$(J)=B$
(J):: X=J :: M$=A$(J):: GOSU
B 1020
1000 NEXT J :: N=G
1010 DISPLAY AT(24,1):"A to
abort   P to pause" :: RETUR
N
1020 RR=X
1030 IF RR>20 THEN RR=RR-20
:: GOTO 1030
1040 CC=1-(X>20)*5-(X>40)*5-
(X>60)*5-(X>80)*5 :: DISPLAY
 AT(RR,CC):A$(X);:: W=W+1 ::
 GOSUB 1060 :: RETURN
1050 DISPLAY AT(22,14):"T$="
;T$ :: W=W+1 :: GOSUB 1060 :
: RETURN
1060 CALL KEY(3,K1,SS):: IF
```

```
SS=0 THEN 1090
1070 IF K1=65 THEN 130
1080 CALL KEY(3,K2,SS):: IF
SS<1 THEN 1080
1090 RETURN
```

Don't try timing these sorts, because the screen display distorts the speed. Option 9 has been left open so that you can add your own favorite sort routine, in the same format, starting in line 25000.

These routines may not be the most efficient forms, and their names may not be correct. If you know better ones, let me know!

=======

```
100 !BASKET WEAVING by Jim P
eterson
110 CALL CLEAR :: W=11 :: T=
2 :: CH$="A5A5A5A5A5A5A5FF
00FF0000FF00FF" :: CALL CHAR
(142,CH$):: CALL COLOR(14,2,
W,13,2,W):: CALL SCREEN(W)
120 CALL HCHAR(1,1,143,768):
: CALL CHAR(134,CH$):: CH=14
2
130 FOR C=1 TO 31 STEP T ::
FOR R=1 TO 23 STEP T :: CALL
 HCHAR(R,C,CH):: NEXT R :: F
OR R=24 TO 2 STEP -T :: CALL
 HCHAR(R,C+1,CH):: NEXT R ::
 NEXT C
140 CH=ABS((CH=142)*135+(CH=
134)*143):: RANDOMIZE :: T=I
NT(3*RND+2)
150 FOR R=1 TO 23 STEP T ::
FOR C=2 TO 32 STEP T :: CALL
 HCHAR(R,C,CH):: NEXT C
160 FOR C=31 TO 1 STEP -T ::
 CALL HCHAR(R+1,C,CH):: NEXT
 C :: NEXT R :: CH=CH-1 :: W
=INT(14*RND+3):: T=INT(3*RND
+2)
170 IF CH=134 THEN CALL COLO
R(13,2,W):: GOTO 130 ELSE CA
LL COLOR(14,2,W):: GOTO 130
```

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

The following routine will create a D/V80 file named GRAPHPAGE, to be loaded into TI-Writer as a 77x57 grid numbered along the left and bottom. Arrow keys can then be used to create a line graph of asterisks or whatever, annotated with text as desired.! # is shift 3

```
100 OPEN #1:"DSK1.GRAPHPAGE"
,OUTPUT :: PRINT #1:TAB(4);R
PT$("_",75):: FOR J=57 TO 1
STEP -1 :: J$=STR$(J)
105 IF J<10 THEN J$=" "&J$
110 PRINT #1:J$&RPT$("ø_",38
)&"ø" :: NEXT J
120 FOR T=1 TO 2 :: PRINT #1
:"  ";:: FOR J=1 TO 77 :: J$
=STR$(J)&" " :: PRINT #1:SEG
$(J$,T,1);:: NEXT J :: PRINT
 #1 :: NEXT T :: CLOSE #1
```

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

```
1 !TO PRINT A HANDY REFERENC
E CHART OF ASCII TO HEX CODE
- MODIFIED FROM READING-BERK
S AUG 85 - # is shift 3
90 OPEN #1:"PIO" :: PRINT #1
:CHR$(27);CHR$(77);CHR$(5)
100 FOR X=32 TO 63 :: FOR Y=
X TO X+64 STEP 32 :: CALL CH
ARPAT(Y,Y$):: PRINT #1:Y;" "
;CHR$(Y);" ";Y$;:: NEXT Y ::
 PRINT #1:"" :: NEXT X
```

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

```
100 CALL CLEAR :: CALL MAGNI
FY(2):: RANDOMIZE :: DISPLAY
 AT(3,2):"TIGERCUB SPEED TYP
ING TEST": :TAB(12);"SPEED"
:: T=10
110 DISPLAY AT(5,18):100-T :
: X=INT(26*RND+65):: CALL SP
RITE(#1,X,2,96,120):: FOR D=
1 TO T :: CALL KEY(3,K,ST)::
 ON (K=X)+2 GOTO 120,130
120 T=T-1 :: GOTO 110
130 NEXT D :: T=T+1 :: GOTO
110
```

TIPS FROM THE TIGERCUB #34

Copyright 1986

Steven Shouse of TIRUG sent this improvement to the GRAPHPAGE in Tips #33 -

```
100 OPEN #1:"DSK1.GRAPHPAGE"
,OUTPUT :: PRINT #1:TAB(4);R
PT$("_",75):: FOR J=57 TO 1
STEP -1 :: J$=STR$(J)
```

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

```
100 CALL CLEAR
110 REM - SONG OF SAMARKAND
programmed by Jim Peterson -
 Version 3
120 RANDOMIZE
130 CALL CHAR(94,"00")
```

```
140 CALL CHAR(95,"00")
150 CALL SCREEN(11)
160 PRINT "From the Third Mo
vement of":"":"    THE NEVER
-ENDING SONG":"":"          b
y Emir Abdul Aziz":"":".....
......................."
170 PRINT : : : : : : : : : :
: : : : : :
180 FOR J=1 TO 23
190 CALL HCHAR(12,5+J,ASC(SE
G$("^THE^SONG^OF^SAMARKAND^"
,J,1)))
200 NEXT J
210 CALL HCHAR(11,6,94,23)
220 CALL HCHAR(13,6,94,23)
230 M$="187EFF42668124C3DB66
5A18423C5AA542817E995A001800
24BDBD3C667E66668100243C0042
187E5AA53CC3427E3C81817E5AE7
669924187E429924008181DBC3"
240 DIM N(30),S(21)
250 F=220
260 FOR J=0 TO 36
270 X=X+1+(X=12)*12
280 IF (X=2)+(X=5)+(X=7)+(X=
10)+(X=12)THEN 310
290 Y=Y+1
300 N(Y)=INT(F*1.059463094^J
)
310 NEXT J
320 CALL HCHAR(1,1,32,320)
330 CALL VCHAR(1,31,95,96)
340 CALL HCHAR(24,1,95,64)
350 CV=2
360 K=8
370 K=K-INT(5*RND+1)+INT(5*R
ND+1)+(K>21)*2-(K<1)*2
380 IF (K<1)+(K>21)THEN 370
390 CALL SOUND(-999,N(K),0,N
(K)*CV,0,N(K)*3.75,30,-4,5)
400 X=INT(40*RND)
410 IF X>12 THEN 370
420 ON X+1 GOTO 430,490,540,
580,660,730,770,850,870,970,
990,1040,1060
430 IF INT(4*RND)<3 THEN 390
440 FOR T=K TO 20
450 CALL SOUND(-999,N(T),0)
460 NEXT T
470 K=1
480 GOTO 390
490 FOR T=K TO 1 STEP -1
500 CALL SOUND(-999,N(T),0)
510 NEXT T
520 K=T+1
530 GOTO 390
540 FOR T=K TO 1 STEP -1
550 CALL SOUND(-999,30000,30
,30000,30,N(T)*3.75,30,-4,0)
560 NEXT T
570 GOTO 370
580 FOR TT=K TO K-INT(5*RND+
1)STEP -1
590 IF TT<2 THEN 370
600 FOR T=1 TO INT(7*RND+3)
610 CALL SOUND(-999,N(TT),0,
N(TT)*2,0)
620 CALL SOUND(-999,N(TT)*1.
03,0,N(TT)*2.06,0)
630 NEXT T
640 NEXT TT
650 GOTO 370
660 FOR T=K TO K-INT(3*RND+3
)STEP -1
670 IF T<2 THEN 370
680 FOR D=0 TO 15 STEP 2
690 CALL SOUND(-999,N(T)*2,D
,N(T)*3,D,N(T)*3.75,30,-4,0)
700 NEXT D
710 NEXT T
720 GOTO 370
730 FOR X=1 TO 15
740 CALL SOUND(-999,N(X),0,N
(16-X),0,N(1),30,-4,5)
750 NEXT X
760 GOTO 370
770 FOR T=K TO K-INT(4*RND+1
)STEP -1
780 IF T<2 THEN 370
790 CALL SOUND(100,N(T),0,N(
T)*2,0,N(T)*3.75,30,-4,5)
800 FOR TT=N(T)TO N(T-1)STEP
 -10
810 CALL SOUND(-999,TT,0,TT*
2,0,TT*3.75,30,-4,5)
820 NEXT TT
830 NEXT T
840 GOTO 370
850 CALL CHAR(32,SEG$(M$,INT
(57*RND+1)*2-1,16))
860 GOTO 370
870 IF INT(4*RND)<3 THEN 390
880 CALL SOUND(-3000,N(K),0,
N(K)*2,0,N(K)*3.75,30,-4,0)
890 FOR J=1 TO INT(5*RND+5)
900 S(J)=INT(21*RND+1)
910 NEXT J
920 CALL SOUND(-1,30000,30)
930 FOR T=1 TO J-1
940 CALL SOUND(-999,N(S(T)),
0,N(S(T))/1.68,0,N(S(T))*3.7
5,30,-4,0)
950 NEXT T
960 GOTO 370
970 CALL CHAR(95,SEG$(M$,INT
(57*RND+1)*2-1,16))
980 GOTO 370
990 IF INT(4*RND)<3 THEN 390
1000 FOR J=220 TO 660 STEP 2
```

```
0
1010 CALL SOUND(-999,J,0,880
-J,0,N(12)*3.75,30,-4,0)
1020 NEXT J
1030 GOTO 370
1040 CALL CHAR(32,"0")
1050 GOTO 390
1060 CV=CV+(CV=2)/2-(CV=1.5)
*.5
1070 GOTO 370
```

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

For those of us who are still struggling along with one disk drive, this routine will transfer any number of D/V80 files, totalling up to about 42 sectors, from one disk to another in one pass, and will optionally save under changed names.

```
100 DIM M$(2000),F$(25),C$(2
5):: CALL CLEAR :: T$=CHR$(1
)
110 DISPLAY AT(8,6):"TIGERCU
B FILEMOVER" :: DISPLAY AT(1
5,1):"PRESS ENTER WHEN FINIS
HED"
120 F=F+1 :: IF F>25 THEN 13
0 :: DISPLAY AT(12,1):"FILEN
AME? DSK"&T$ :: ACCEPT AT(12
,14)SIZE(-12)BEEP:F$(F):: IF
 F$(F)<>T$ THEN 120
130 F=F-1 :: FOR J=1 TO F ::
 ON ERROR 260 :: OPEN #1:"DS
K"&F$(J),INPUT :: DISPLAY AT
(12,1):"READING "&SEG$(F$(J)
,3,255)
140 X=X+1 :: LINPUT #1:M$(X)
:: C=C+LEN(M$(X))
150 IF C>10000 THEN DISPLAY
AT(20,1):"INSUFFICIENT MEMOR
Y FOR "&SEG$(F$(J),3,255)::
GOTO 190
160 IF EOF(1)<>1 THEN 140
170 X=X+1 :: M$(X)=T$ :: CLO
SE #1
180 W=W+1 :: NEXT J
190 X=0 :: DISPLAY AT(15,1):
"" :: DISPLAY AT(12,1):"INSE
RT COPY DISK AND PRESS":"ENT
ER"
200 CALL KEY(0,K,ST):: IF ST
=0 THEN 200 :: DISPLAY AT(13
,1):""
210 FOR J=1 TO W :: IF F$(J)
=CHR$(2)THEN 230
220 DISPLAY AT(12,1):"FILENA
ME? DSK"&F$(J):: ACCEPT AT(1
2,14)SIZE(-12)BEEP:C$(J)230
NEXT J :: FOR J=1 TO W :: IF
 F$(J)=CHR$(2)THEN 250 :: OP
EN #1:"DSK"&C$(J),OUTPUT ::
DISPLAY AT(12,1):"SAVING "&S
EG$(C$(J),3,255)
240 X=X+1 :: IF M$(X)<>T$ TH
EN PRINT #1:M$(X):: GOTO 240
 ELSE CLOSE #1
250 NEXT J :: END
260 ON ERROR STOP :: DISPLAY
 AT(22,1):"CANNOT OPEN "&SEG
$(F$(J),3,255):: F$(J)=CHR$(
2):: RETURN 180
```

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

Here is a very ingenious idea published in the Corpus Christi UG newsletter by H. Macdonald. He could not find the author/newsletter which gave him the idea, so if you know, tell me and I'll print due credit.

I have modified it a bit. This short routine will load quickly and enable you to bypass loading and running the Menu Loader program on a disk when you already know the filename of the program you want to run.

Save the Menu Loader under the filename MENULOADER and save this routine under the filename LOAD - be sure to save it before you try it, because it erases itself!

```
100 CALL INIT :: CALL LOAD(-
31806,16):: DISPLAY AT(12,1)
ERASE ALL:"RUN MENULOADER? (
Y/N)"
110 CALL KEY(3,K,S):: IF S=0
 THEN 110 ELSE IF K=78 THEN
130 ELSE DISPLAY AT(12,1)ERA
SE ALL:"LOADING MENULOADER"
:: RUN "DSK1.MENULOADER"
130 CALL CLEAR :: CALL LOAD(
-31952,55,215,55,215):: END
```

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

Here is one with a bit of a surprise at the end. Key the v,A in line 190 as FCTN V, CTRL comma, CTRL A.

```
100 CALL CLEAR :: CALL SCREE
N(16)
110 DATA 80C0A09088445269,00
00000000007EB1,0103050911224
A96,0000000101010100,21409C2
A492A1CC0,9999336600001824
120 DATA 8482395492543903,00
00000000808080,E0B09880E7702
010,18244281423C0000,0F19030
```

```
7E1020408,000000FF80808080
130 DATA 000F13E620221D00,0C
FB34670A22DC00,814224FF,30DF
2CC641443B00,00F0C86F0447B87
F,000000FF01F901F9
140 DATA 80FF808686808686,00
FF00666600666,00FF003F3F3F3
F3F,01FF01F9F9F9F9F9,8086868
086868093,006666006666600FF
150 DATA 0066660066660OE6,3F
3F3F3F3F3F3F3F,F9F9F9F9F9F9F
9F9,00000000E01C3AE2,9380FF,
FF00FF,E600FF00070B0807
160 DATA 3F00FF00FF1988FF,F9
01FF00FF8744FF,1F09090FF3198
AFC
170 FOR CH=96 TO 129 :: READ
 CH$ :: CALL CHAR(CH,CH$)::
NEXT CH
180 DISPLAY AT(1,14)ERASE AL
L:"`ab" :: DISPLAY AT(2,13):
"cdefg" :: DISPLAY AT(3,14):
"hij" :: DISPLAY AT(4,12):"k
lmnopq"
190 DISPLAY AT(5,12):"rsssst
u" :: DISPLAY AT(6,12):"vwww
xyzæ" :: DISPLAY AT(7,12):"ø
åàå~v,A" :: DISPLAY AT(9,12)
:"TIGERCUB"
200 DISPLAY AT(11,12):"SOFTW
ARE" :: DISPLAY AT(13,7):"15
6 COLLINGWOOD AVE." :: DISPL
AY AT(15,7):" COLUMBUS OH 43
213" :: CALL HIGHCHAR
210 GOTO 210
220 SUB HIGHCHAR :: FOR CH=3
2 TO 129 :: CALL CHARPAT(CH,
CH$):: X$=SEG$(CH$,3,12)&SEG
$(CH$,13,4):: CALL CHAR(CH,X
$):: NEXT CH :: SUBEND
```

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

Thanks to Ramon Martinez in the Orange County UG news letter - a double NEXT is accepted if the pre-scan is turned off.

```
100 J=1
110 !@P-
120 FOR J=1 TO 100 :: IF J/1
0<>INT(J/10)THEN NEXT J ELSE
 PRINT J :: NEXT J
```

=======

A computer without a program is like a car without gas. If everyone who filled up at a self-service pump drove away without paying, how soon would all the gas stations be closed?

MEMORY FULL!

Jim Peterson
TIPS FROM THE TIGERCUB

No. 69

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

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

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

In Tips #68 I published my solution to Dr. Ecker's challenge to alternately assign X the value of A and B without using IF...THEN or any outside help. Computer Monthly has arrived again and his solution is better than mine. Try it with any two numbers -

```
100 A=2.765 :: B=-10
110 X=A+B-X :: PRINT X :: GO
TO 110
```

There has been controversy for years as to whether the TI's psuedorandom number generator is truly random. Dr. Ecker's "Computer Fun & Learning" column in Computer Monthly had a question - if you randomly generate numbers between 0 and 9, how often will you get the same number twice in succession? Three times in succession? And etc. Since there are 10 numbers to choose from, it seems to me you would get 2 in a row 10% of the time, 3 in a row 1% of the time, 4 in a row .1%...etc. I wrote this to prove it -

```
100 RANDOMIZE
110 C=C+1 :: X=INT(RND*10)::
 PRINT X;:: IF X=F THEN FL=F
L+1 :: CL(FL)=CL(FL)+1 :: PR
INT "":FL;"=";CL(FL):"C=";C:
"%=";CL(FL)/C :: GOTO 110 EL
SE FL=0 :: F=X :: GOTO 110
```

After 10,000 tries, I had 2 in a row 8.75% of the time and 3 in a row .83% and 4 in a row .07% . Does that prove anything? I don't know.

(Dr. Ecker points out that those percentages could not ever quite add up to 100%!)

Here is another of my XBasic programs to write assembly source code -

```
100 DISPLAY AT(2,1)ERASE ALL
:"ASSEMBLY HELP SCREEN WRITE
R":"":" This program will wr
ite the":"source code for an
 assembly":"routine which ca
n be linked"
110 DISPLAY AT(7,1):"from Ex
tended Basic to dis-":"play
any one of several help":"sc
reens at any designated":"ke
y press or input at any":"po
int in a program."
120 DISPLAY AT(12,1):" The o
riginal source code,":"autho
r unknown, was improved":"by
 Karl Romstedt and further":
"modified by Bruce Harrison.
"
130 DISPLAY AT(20,1):"How ma
ny help screens?" :: ACCEPT
AT(20,24)SIZE(1)VALIDATE(DIG
IT)BEEP:N
140 FOR J=1 TO N :: H$=H$&"H
ELP"&STR$(J)&"," :: NEXT J :
: H$="       DEF     "&SEG$(H$,
1,LEN(H$)-1)
150 DATA VMBW    EQU   >2024,V
MBR    EQU   >202C,KSCAN   EQU
 >201C,STATUS EQU   >837C
160 OPEN #1:"DSK1.HELP/S",OU
TPUT :: PRINT #1:H$ :: FOR J
=1 TO 4 :: READ M$ :: PRINT
#1:M$ :: NEXT J
170 FOR J=1 TO N :: H$="HELP
"&STR$(J):: PRINT #1:H$&"  L
WPI WS":"        LI    R13,HEL
PS"&STR$(J)
180 IF J<N THEN PRINT #1:"
     JMP  SAVSCR"
190 NEXT J :: H$=RPT$(" ",7)
200 PRINT #1:"SAVSCR CLR  R0
":H$&"LI    R1,SAVIT":H$&"LI
  R2,768":H$&"BLWP @VMBR":H$
&"LI    R9,NEWSCR":H$&"MOV   R
9,R1":H$&"MOV  R2,R4"
210 PRINT #1:H$&"LI    R3,>60
00":"ADDOFF MOVB *R13+,*R9":
H$&"AB    R3,*R9+":H$&"DEC   R
4":H$&"JNE .ADDOFF":H$&"BLWP
 @VMBW"
220 PRINT #1:"KEYLOO BLWP @K
SCAN":H$&"BLWP @KSCAN":H$&"C
B    @ANYKEY,@STATUS":H$&"JNE
  KEYLOO"
230 PRINT #1:"REPL   LI    R1
,SAVIT":H$&"BLWP @VMBW":"RET
N    LWPI >83E0":H$&"B     @>6
A"
240 PRINT #1:"WS      BSS  32
":"SAVIT  BSS  768":"NEWSCR
BSS  768":"ANYKEY BYTE >20":
H$&"EVEN"
250 DISPLAY AT(3,1)ERASE ALL
:" Enter data just as you":"
want it to appear, in 24":"l
ines. Press Enter for blank"
:"lines."
260 FOR J=1 TO N :: DISPLAY
AT(12,1):"Ready for screen #
"&STR$(J):"":"Press any key"
270 CALL KEY(0,K,S):: IF S=0
 THEN 270 ELSE CALL CLEAR
280 ACCEPT AT(1,0):M$ :: PRI
NT #1:"HELPS"&STR$(J)&" TEXT
 ' "&M$&RPT$(" ",30-LEN(M$))
&" '"
290 FOR K=2 TO 24 :: ACCEPT
AT(K,0):M$ :: PRINT #1:H$&"T
EXT ' "&M$&RPT$(" ",30-LEN(M
$))&" '"
300 NEXT K :: NEXT J :: PRIN
T #1:H$&"END"
310 DISPLAY AT(3,1)ERASE ALL
:" Source code has been writ
-":"ten to DSK1 as HELP/S. T
o":"assemble, insert Editor/
":"Assembler module."
320 DISPLAY AT(7,1):"Insert
Assembler disk in    drive 1
```

```
.":"Select 2 ASSEMBLER":"Loa
d Assembler? Y":"Source file
name DSK2.HELP/S"
330 DISPLAY AT(12,1):"Object
 file name? DSK2.HELP/O":"Li
st file name? Press Enter":"
Options? R"
340 DISPLAY AT(15,1):"Load t
he resulting object":"file i
nto your program by":"CALL I
NIT ::":"CALL LOAD(""DSK1.HE
LP/O"") or,"
350 DISPLAY AT(19,1):"much b
etter, imbed it with":"ALSAV
E or SYSTEX."
360 DISPLAY AT(21,1):"Access
 the screens in your  progra
m by":" CALL LINK(""HELP1"")
":"CALL LINK(""HELP2""), etc
."
370 CALL KEY(0,K,S):: IF S=0
 THEN 370 ELSE CALL CLEAR
```

For instance, at any point in a program where keyboard input is required and user may not know what to do - ACCEPT AT(24,1):M$ :: IF M$="HELP" THEN CALL LINK("HELP1") and the first help screen will pop up to give instructions. Press any key and the previous screen reappears.

This time I am borrowing heavily from the TI*MES news letter of England, which has also borrowed from the REC newsletter.

This one is useless, but is a remarkable example of compact complex programming. It shows that there is an algorithm for everything. See if you can figure out how it works -

```
100 CALL CLEAR :: FOR A=1 TO
 2 :: FOR B=1 TO 4 :: X=2-AB
S(SGN(B-3)):: FOR C=1 TO X :
: PRINT CHR$(84-7*A+5*B-8*X)
;:: NEXT C :: NEXT B :: PRIN
T CHR$(A+31):: NEXT A
```

Another useless one that is easier to figure out -

```
100 DISPLAY AT(1,1)ERASE ALL
:"NUMBER OF MONTH(1-12)"
110 ACCEPT AT(2,12)SIZE(2)VA
LIDATE(DIGIT):A :: IF A<1 OR
 A>12 THEN 110
120 DISPLAY AT(3,1):A;"x 4="
;A*4 :: A=A*4
130 DISPLAY AT(4,1):A;"+13="
;A+13 :: A=A+13
140 DISPLAY AT(5,1):A;"x 25=
";A*25 :: A=A*25
150 DISPLAY AT(6,1):A;"-200=
";A-200 :: A=A-200
160 DISPLAY AT(8,1):"Input d
ate (1-31):" :: ACCEPT AT(8,
19)SIZE(2)VALIDATE(DIGIT):B
:: IF B<1 OR B>31 THEN 160
170 DISPLAY AT(10,1):A;"+";B
;"=";A+B :: A=A+B
180 DISPLAY AT(11,1):A;"x 2=
";A*2 :: A=A*2
190 DISPLAY AT(12,1):A;"-40=
";A-40 :: A=A-40
200 DISPLAY AT(13,1):A;"x 50
=";A*50 :: A=A*50
210 DISPLAY AT(15,1):"Input
last two digits of    year e
g 91:"
220 ACCEPT AT(16,16)SIZE(2)V
ALIDATE(DIGIT):B
230 DISPLAY AT(18,1):A;"+";B
;"=";A+B :: A=A+B
240 DISPLAY AT(19,1):A;"-105
00=";A-10500 :: A=A-10500
250 DISPLAY AT(24,1):"ANY KE
Y FOR ANOTHER"
260 CALL KEY(5,A,B)
270 IF B<1 THEN 260
280 RUN
290 END
```

One for the little ones - change the string to anything you want.

```
1 REM SILLY PROG BY S SHAW
  MARCH 1991
2 ! did you see COMPUTER WAR
S-the film? It is said that
the star, who was required t
o type fast into a computer
3 ! could not type, so a pro
gram just like this one was
used to give a good effect!
4 ! now adjust it how you wi
sh and show your friends how
 fast you can type
5 ! at end of text string pr
ogram will just stop with th
is listing but can be modifi
ed to do anything you wish!
6 !
100 A$="This is how a non-ty
```

```
pist canproduce information
on      screen quickly,witho
ut      "
110 A$=A$&"having to look at
 what keys are being bashed!
 Just bash keys and watch ho
w perfect  text appears no m
atter what you press."
120 CALL CLEAR :: PRINT A$:
: : : : :
130 CALL KEY(5,A,B):: IF B<1
 THEN 130
140 C=C+1 :: PRINT SEG$(A$,C
,1);:: IF C=LEN(A$)THEN 160
150 GOTO 130
160 GOTO 160
```

And a very fast routine to find prime numbers -

```
100 ! FIRST 100 PRIMES
    -QUICKLY-
110 ! Dr H B Phillips
  from THE REC NEWSLETTER
 March 1988  Vol 3 #2
120 DIM P(300),X(12)
130 A=0 :: B=1 :: D=0.5 :: E
=180
140 M=100 :: L=3 :: F=0
150 ! increase M for more- a
lso increase DIMs.
160 PRINT 2;:: C=B :: IF M=B
 THEN END
170 L=INT((M/C)*L+F):: N=L+L
+B
180 FOR I=B TO INT((SQR(N)-B
)*D):: PP=P(I)
190 IF PP=B THEN 230
200 IF PP=A THEN PP=I+I+B ::
 PRINT PP;:: P(I)=PP :: C=C+
B :: IF C=M THEN END
210 IF X(I)=A THEN X(I)=(PP*
PP-B)*D
220 FOR J=X(I)TO L STEP PP :
: P(J)=B :: NEXT J :: X(I)=J
230 NEXT I :: IF F=0 THEN S=
I
240 FOR I=S TO L
250 IF P(I)=A THEN PP=I+I+B
:: PRINT PP;:: P(I)=PP :: C=
C+B :: IF C=M THEN END
260 NEXT I :: F=(M-C)*L/E ::
 S=L+B
270 GOTO 170
```

And a demonstration of how the INTERRUPT routine works independently of whatever else the computer is doing -

```
100 REM interrupt demo
110 REM
120 REM MACHINE LANGUAGE
130 REM ROUTINE LOADED AT
140 REM >2600 XB OR E/A WITH
 32K
150 REM >7200 MINI MEM NO 32
K
160 REM
170 CALL INIT
180 XM=9728
190 MM=29184
200 LAD=XM
210 REM TEST XB OR MM?
220 CALL LOAD(XM,170)
230 CALL PEEK(XM,X)
240 IF X=170 THEN 270
250 REM NO 32K MUST BE MM
260 LAD=MM
270 A=LAD
280 REM LOAD M/C
290 CALL CLEAR
300 FOR D=540 TO 630 STEP 10
310 CHECK=0
320 FOR N=1 TO 10
330 READ X
340 CALL LOAD(A,X)
350 CHECK=CHECK+X
360 A=A+1
370 NEXT N
380 READ X
390 IF CHECK<>X THEN 490
400 NEXT D
410 REM POKE INTERRUPT
420 REM ROUTINE ADDRESS
430 REM INTO >83C4
440 CALL LOAD(-31804,LAD/256
)
450 REM JUST IDLE AWAY TIME
460 FOR N=1 TO 9940
470 NEXT N
480 STOP
490 PRINT "ERROR IN DATA STA
TEMENT ";D
500 STOP
510 REM EACH DATA STATEMENT
520 REM HAS 10 DATA BYTES
530 REM PLUS A CHECK SUM
540 DATA 192,236,000,092,004
,194,005,131,002,131,987
550 DATA 000,060,026,003,004
,195,006,236,000,094,624
560 DATA 203,003,000,092,060
,172,000,090,006,002,628
570 DATA 017,015,019,010,006
,002,019,004,002,000,94
580 DATA 002,039,010,083,016
,002,002,000,002,086,242
590 DATA 096,003,016,007,002
,000,000,119,010,083,336
```

```
600 DATA 016,002,002,000,000
,072,160,003,002,096,353
610 DATA 064,000,006,192,215
,192,006,192,215,192,1274
620 DATA 016,000,216,044,000
,094,140,000,004,091,605
630 DATA 000,015,000,000,138
,128,000,000,000,000,281
640 END
```

Run that, then press FCTN 4. Enter LIST. Enter NEW. To stop it, enter BYE.

This is an oldie, but well worth repeating. You can use it to turn your cassette recorder on and off, to add speech or music from tape to a running program. With the proper hardware, you could write a program to control almost anything from the cassette port. If it doesn't work, reverse the polarity of the remote. Ed Hall wrote this -

```
100 CALL INIT
110 CALL LOAD(16368,79,70,70
,32,32,32,36,252)
120 CALL LOAD(16376,79,78,32
,32,32,32,36,244)
130 CALL LOAD(8194,37,4,63,2
40)
140 CALL LOAD(9460,2,12,0,45
,29,0,4,91,2,12,0,45,30,0,4,
91,203,78)
150 PRINT "PRESS":" P Play":
"S Stop"
160 CALL KEY(3,A,B)
170 IF B<1 THEN 160
180 ON POS("PS",CHR$(A),1)+1
 GOTO 160,190,200)
190 CALL LINK("ON"):: GOTO 1
60
200 CALL LINK("OFF"):: GOTO
160
```

And that is just about -

MEMORY FULL!

Jim Peterson

----

sjs- tips 69 repeats material from earlier TI*MES but is unedited to benefit readers who are new to us or who missed this items first time round. Other readers- please bear with us! Ta. sjs

AN OLD NUMBER PUZZLE.

Older members may recall the following number puzzle bobbing up at intervals, from the late fifties for about twenty years, in magazines and coffee-break chat. The solution was actually printed in 1969 but where most of us were unlikely to see it. I worked at the puzzle, on and off, for fifteen years and learned a lot of mathematics in the process. If I had owned a micro at the time, no doubt my programming skills would have received a boost as well.

In the hope of similarly benefitting others to whom the puzzle will be new, I give it here:

1 3 8 120 ....

Observe that the product of any pair of these numbers is one less than a perfect square. Can you find a number (or numbers) to continue the sequence and maintain the same property? I warn you: the problem is difficult.

If the Editor agrees, I will report the solution in the next TI*MES and add some related material.

Walter Allum

ASSEMBLY - BRUCE HARRISON - PART 3

The Art of Assembly - Part 3

Starting at the Top

By Bruce Harrison
Copyright 1991, Harrison Software

In Part 2 of this series, we discussed and showed some small "primitive" subroutines and the methods for nesting them. In this article, we are going back to the "Top Down" part of writing Assembly programs. We will use for our example the Harrison Golf Score Analyzer, since its development went pretty much along the lines we're trying to encourage.

One of the first decisions you should make is how the user will interact with your program. In many games, for example, the principal means of interaction is the joystick. In a program like a Golf Score Analyzer, however, that would be a very poor interface for user input. Our preference is for simple menu interaction at the top level, so each main function of the program is readily apparent to the user, and selection of a function is just one keystroke away. In today's world of "Graphical User Interface" (GUI), where functions are represented by pictures, not words, this makes us very old-fashioned, but we do have a reason for being that way. GUIs normally require a mouse to select options, and one can't count on every customer having a mouse. Further, a mouse can't be used to input names, numbers, and other data, so with or without a mouse one still needs the keyboard. Our choice has been to require only the keyboard, and that makes "plain English" menus the natural choice for selecting functions.

Given that, we must make a decision as to what functions belong on the main menu. In the Golf Score Analyzer, we settled on eight functions for the Main Menu, and made each require only a single keystroke to select. The eight look like this:

```
1  ADD ROUNDS
2  LOAD FILE
3  DELETE DATA
4  ANALYZE DATA
5  SAVE FILE
6  ADD/EDIT COURSES
7  REVIEW COURSES
8  EXIT PROGRAM
```

It's important to always include an exit selection, so the user can easily get out of your program when he wants to. It's equally important to make it difficult or impossible to get out of the program by accident. In this program, selecting item 8 from the Main Menu is the only way to get out. We made Funtion-Quit inactive in this program. As an aside, when users are looking at subsidiary menus, Function-9 (BACK) will get back to the previous menu, but that will not get them out of the program.

In this particular program, we had a special reason for making one and only one exit point. When the user selects item 8, we perform a check to see whether the user has modified the file currently in memory. If he's not made any changes to the file, or if he's saved it since making changes, we simply return him to either XB or E/A, depending on how he entered the program. If changes have been made, we produce a prompt asking whether he'd like to save the changed file before exiting. Any answer other than N or n is taken as Yes, and he's placed in the SAVE FILE function. We take these precautions as part of our concept of "User Friendliness".

Perhaps we could illustrate the concept of User Friendliness by an example drawn from experience. In many instances on the TI, one will encounter an error in execution of some program. Let's say we're working in XB or E/A, and try to get a nonexistent file to open for INPUT. What the TI folks will give you is a numbered "ERROR CODE", which you'll have to look up in a book. When we write our own programs, we like to provide a more definitive error indication, like "THAT FILE DOES NOT EXIST ON DRIVE x" or "THERE IS NO DISK IN DRIVE x". This way the user has a very definite idea of what's wrong. Doing this of course eats memory, since those error messages have to be stored somewhere in the computer and printed to the screen, but we think that's a worthwhile use of memory.

But we digress. Once one has decided upon a menu, the top part of the flow chart is readily apparent. There will need to be an opening section of code that sets up such things as screen mode, color scheme, and such, then displays our copyright notice. Next is a delay loop so the user can read the copyright notice, and then we clear the screen and produce the main menu. Here we had to make a decision. Since we knew there would be more than one menu, we could have each menu produced by a separate section of code, or we could provide a "Menu Driver" section of code that would produce all the required menus simply by using different data with the same code. We chose the latter, and believe that was a wise decision, because we used less memory to do it this way. Our Word Processor, which we use to prepare these articles, also has a central menu driver, but the one in the Golf Score Analyzer is better, taking lessons learned from the WP program into account.

Each menu we use has a section of data associated with it, which includes the title for the top of that menu, the selections, and a "branching" lookup table, which indicates where the program will go to when it exits that menu. The legend "SELECT BY NUMBER" goes at the bottom of each menu, so the menu driver itself places that legend on each menu it displays. In our Golf Score Analyzer, by the way, we separated the code from the data into sections of memory. That is, all the executable instructions are together in a block of memory, then all the data, including text for messages and menus, is in its own block of memory. This makes a somewhat neater arrangement for the programmer, in that separate source files contain the data, and it becomes a bit easier to keep track of what one is doing while developing the program. It also makes it easier when one comes back six months later to change something in the program.

Actually, there's no reason you can't scatter data all over the place, between sections of the executable code, but our thinking on the subject has been colored by the fact that we also program in PC Assembly language, where different memory segments are (and must be) allocated for code and data. This becomes a habit that carries over to the TI.

There we are digressing again. Just for the heck of it, let's look at some of the source code. In the sidebar is the annotated source code associated with the menu driver for the Golf Score Analyzer. The first two executable lines are the required setup before branching to the driver. These lines set R9 to point to the data for the menu itself, and R13 to point to the lookup table for branching out of the Main Menu.

In the Driver itself (MENDRV), the first order of business is to clear the screen. The CLS subroutine is similar to the one shown in our last article, except that, since GSA was written to operate from Extended Basic, it adds an offset of >60 to the spaces it writes into SCRLI. As an ironic sidelight, we later added a loader so that GSA could be run from E/A, and in that loader we had to, among other things, re-arrange the tables in VDP so it would need the character offset.

Before delving further into the code, let's look at the structure of the data for the menu, at label MENDAT. It starts with a byte giving the length of the title for the menu. Next is the text of the title, then two bytes. The first of these is the number of items in this menu (8), and the second is the length of the first item description (13). After this is the text for the first item, followed by the strings for the rest of the items (a length byte, then text content). By organizing the data this way, we can make a loop in the menu driver that minimizes the memory used for the driver's code.

The business of getting the menu on-screen now proceeds by taking the length of the title line and manipulating that to position R0 so the title appears centered on Row 2 of the screen. The subroutine DISLI could also be called DISSTR, since what it does is take a string pointed to by R9 and display it at the screen location pointed to by R0. Another irony here is that, had we done this in E/A only, we could have used R1 as the pointer to the string, then DISLI would reduce to:

```
DISLI    MOVB *R1+,R2   Get length byte into R2
         SRL  R2,8      Right justify R2
         BLWP @VMBW     Write characters to screen
         A    R2,R1     Advance R1 beyond text
         RT             Return
```

But we didn't do that, because we wanted GSA to be available to those who don't have the E/A module, but only the XB module. Thus we're stuck with that offset, even when the user enters the program from E/A. Live and Learn!

Another small note before we examine the rest of the source code. There is no such thing as a perfect program. As your author looks at his own sidebar, he can see several places where it could be improved. For example, the line just before label LOP1 in the CLS subroutine could be eliminated if the line at LOP1 said MOVB @SPACE,*R6+. Our good friend Jim Peterson (TIGERCUB), calls this kind of thinking Elegant Programming, where the programmer not only wants it to work, but wants it to be fully optimized in all respects. Maybe our next program will be better, but we're not going to re-assemble GSA just for that one small possible change.

Okay, so after the title is on the screen, we have a section of code that picks up the byte just after the title text, transfers that to R8, right justifies it, then stashes it at NOITEM. As it happens, the main menu has eight items, which is the most of any menu used in the program. The next section of code does some math with R0 and R8 to position the bulk of the menu vertically centered between the top and bottom of the screen. At label MEN1, we enter a loop which prints all the selections on the menu. Each call to DISLI leaves R9 pointed at the length byte for the next item, so the loop can proceed very quickly and efficiently.

Once all the items have been displayed (after JNE MEN1) there's another of our little tricks. We want the legend to appear at row 23, column 9. To do that, we let the assembler do the math for us. The assembler multiplies 22 by the width of the screen (this would place us at row 23, column 1), then adds eight to that number. The result is an immediate value placed in R0 which puts R0 just where we wanted it. This trick can be used in many ways, but here we've used it for positioning on the screen. One takes the number of the desired row, subtracts one, then tells the assembler to multiply by SCRWID, and then to add one less than the desired column. In addition to saving us some math, this also saves some time in program execution, because the math is performed during the assembly, and all the computer has to do at running time is load that one value into R0.

Now, once the legend is on the screen, all we need do is wait for the user to press a key. KEYLOO is the subroutine that does this for us, (see Part 2 for that subroutine) and in addition places the ASCII value of the struck key in R8. Given a keystroke, the menu driver checks it against the value 15. Fifteen happens to be the ASCII value for Function-9. In any of the menus in this program, striking Function-9 makes a branching to the last label in the lookup table for that menu. In the case of this main menu, that simply takes us back to KYIN for another keystroke. In other menus, that last label in the lookup table takes us back to a previous menu.

Having found some key value other than 15, the program must now make sure that the key struck in within the correct range for this menu. In this case that's 1 through 8. We move the keystroke to R5, subtract >30 so the number in R5 will be 1 through 8, not >31 through >38. Now we check for a result zero or less than zero. If either happens, the key struck was out of range, so we ignore it and jump back to label KYIN. Finally, we compare R5 to the data at NOITEM, which in this case contains 8. If it's greater than that, we again ignore the keystroke. While this menu is on-screen, hitting any key other than the numbers 1 through 8 will have no effect whatsoever.

Immediately after the operation JGT KYIN, we know that the number in R5 is a number in the range 1 through 8, so we can proceed to branch out from the menu. First we must DEC R5, so that the range is actually 0 through 7. (If Function-9 had been struck, we'd jump to label ACC2 with 8 in R5.) Now, since we're going to index a table of words, not bytes, we must double the number in R5. The easiest way to double a simple integer like this is to Shift it left by one bit, and that's what we do at label ACC2. Now the number in R5 has a range of 0 through 14 (by twos), or 16 if Function-9 has been pressed.

R5 now has the index value for the member of the lookup table we want. We add R13, which contains the address of the start of the lookup table. The next operation, MOV *R5,R5, takes the number at that address in the lookup table and places it in R5. Finally, we branch to the address contained in R5, and that takes us into the selected function. In effect, we have performed an ON-GOTO function based on the key struck.

In this article we looked at some overall principles for Top-Down program design, then we presented one alternative for user interaction through a Menu Selection, and showed the source code for a reasonably effective menu driver. There are many other ways to implement a menu system, and we can be sure that some of our readers will come up with better ones than ours. Our purpose in these articles is mainly to teach principles of using Assembly, so the reader can use his own creativity in this language.

In the next installment, we'll try to concentrate on ways to make code as efficient in memory use as possible, with some "wrong way" and "right way" examples.

```
* PORTIONS OF SOURCE CODE FROM GOLF SCORE ANALYZER
*       > AS MENTIONED IN OPENING PARAGRAPH <
* EQUATE FOR 32 CHARACTER SCREEN
SCRWID EQU  32
*
* SETUP FOR ENTERING MENU DRIVER TO MAKE MAIN MENU
       LI   R9,MENDAT
       LI   R13,MAINBR
       B    @MENDRV
*
```

```
*
* MENU DRIVER SOURCE CODE STARTS
MENDRV
       BL   @CLS          CLEAR THE SCREEN
       LI   R0,SCRWID     SET R0 TO SCREEN WIDTH
       MOVB *R9,R1        GET LENGTH OF TITLE IN R1
       SRL  R1,8          RIGHT JUSTIFY LENGTH
       S    R1,R0         SUBTRACT LENGTH FROM SCREEN WIDTH
       SRL  R0,1          CUT THAT NUMBER IN HALF
       AI   R0,SCRWID     ADD ONE SCREEN WIDTH
* THE ABOVE SECTION SETS R0 AT A VALUE WHICH WILL AUTO-CENTER THE TITLE
* IN ROW 2 OF THE SCREEN
       BL   @DISLI        DISPLAY THAT LINE OF TEXT
* DISLI ADVANCES R9, SO IT NOW POINTS TO BYTE BEYOND END OF TITLE'S TEXT
       MOVB *R9+,R8       GET NUMBER OF ITEMS FOR MENU
       SRL  R8,8          RIGHT JUSTIFY IN R8
       MOV  R8,@NOITEM    STASH THE NUMBER OF ITEMS AS DATA
       LI   R0,8          LOAD R0 WITH MAXIMUM NUMBER OF ITEMS IN ANY MENU
       S    R8,R0         SUBTRACT THE NUMBER OF ITEMS
       AI   R0,4          ADD FOUR
       LI   R3,SCRWID     GET R3 TO EQUAL NUMBER OF CHARACTERS IN SCREEN WIDTH
       MPY  R3,R0         MULTIPLY BY WIDTH OF SCREEN
       AI   R1,8          ADD 8 FOR COLUMN POSITIONING
       MOV  R1,R0         PLACE THIS NUMBER IN R0
* THE CODE ABOVE SETS R0 TO VERTICALLY CENTER THE NUMBER OF ITEMS IN THE MENU
* FOR A MORE CONSISTENT SCREEN APPEARANCE.
MEN1   BL   @DISLI        DISPLAY A LINE OF THE MENU
       AI   R0,SCRWID*2   MOVE DOWN-SCREEN BY TWO LINES
       DEC  R8            DECREMENT COUNTER FOR NUMBER OF ITEMS
       JNE  MEN1          IF NOT ZERO, JUMP BACK TO DISPLAY NEXT ITEM
       LI   R0,22*SCRWID+8  SET R0 FOR ROW 23, COLUMN 9
       LI   R9,SELEC      POINT TO STRING FOR "SELECT BY NUMBER"
       BL   @DISLI        DISPLAY THAT LEGEND
KYIN   BL   @KEYLOO       GET A KEYSTROKE
       CI   R8,15         WAS FUNCTION-9 STRUCK?
       JNE  ACC1          IF NOT, JUMP AHEAD
       MOV  @NOITEM,R5    ELSE PUT NUMBER OF ITEMS IN R5
       JMP  ACC2          THEN JUMP
ACC1   MOV  R8,R5         PLACE KEYSTROKE IN R5
       S    @NUMASK,R5    SUBTRACT >30 SO R5=NUMBER
       JEQ  KYIN          IF R5 ZERO, GO GET ANOTHER KEYSTROKE, IGNORE THIS ONE
       JLT  KYIN          IF R5 < ZERO, IGNORE
       C    R5,@NOITEM    ELSE COMPARE TO NUMBER OF ITEMS
       JGT  KYIN          IF GREATER, IGNORE
* AT THIS POINT, WE KNOW A NUMBER KEY WITHIN THE CORRECT RANGE HAS BEEN STRUCK
       DEC  R5            ZERO-BASE THE VALUE IN R5
ACC2   SLA  R5,1          DOUBLE THAT NUMBER, SINCE WE'RE INDEXING BY WORDS
       A    R13,R5        ADD TO R5 THE START OF THE BRANCHING TABLE
       MOV  *R5,R5        GET THE ADDRESS OF THE SELECTED CODE SECTION INTO R5
       B    *R5           AND BRANCH TO THAT ADDRESS
* END OF MENU DRIVER SOURCE CODE
* SUBROUTINE TO CLEAR SCREEN WITH OFFSET FOR XB
CLS
       LI   R4,SCRWID     SET R4 TO WIDTH OF SCREEN
       MOV  R4,R2         MAKE R2 ALSO = WIDTH OF SCREEN
       LI   R6,SCRLI      POINT R6 AT SCREEN LINE STORAGE
       MOV  R6,R1         PLACE THAT ADDRESS IN R1 ALSO
       MOVB @SPACE,R5     PUT A SPACE WITH OFFSET INTO R5
* THE BYTE AT LABEL SPACE IS >20 + >60 FOR XB'S OFFSET
*
```

```
LOP1   MOVB R5,*R6+      MOVE ONE SPACE WITH OFFSET, INC R6
       DEC  R4           DECREMENT COUNTER
       JNE  LOP1         IF NOT ZERO, REPEAT
       CLR  R0           SET R0 TO SCREEN ORIGIN
       LI   R4,24        24 ROWS TO CLEAR
LOP2   BLWP @VMBW        WRITE ONE LINE OF SCRWID SPACES
       A    R2,R0        ADD SCRWID TO R0
       DEC  R4           DECREMENT ROW COUNT
       JNE  LOP2         IF NOT ZERO, REPEAT
       RT                ELSE RETURN
* SUBROUTINE TO DISPLAY ONE STRING ON THE SCREEN
DISLI  LI   R10,SCRLI    POINT AT OUR BUFFER SCRLI
       MOV  R10,R1       MAKE R1 POINT AT THAT ADDRESS ALSO
       MOVB *R9+,R4      MOVE THE LENGTH BYTE INTO R4
       SRL  R4,8         RIGHT JUSTIFY
       MOV  R4,R2        PLACE THAT NUMBER IN R2 FOR VMBW
       JEQ  DISLIX       IF THAT LENGTH WAS ZERO, GET OUT OF SUBROUTINE
DIS1   MOVB *R9+,*R10    MOVE ONE BYTE OF CONTENT, INCREMENTING R9
       AB   @OFFSET,*R10+ ADD THE >60 OFFSET, AND INCREMENT R10
       DEC  R4           DECREMENT LENGTH COUNT
       JNE  DIS1         IF NOT ZERO, REPEAT
       BLWP @VMBW        WRITE THE STRING WITH OFFSET TO SCREEN
DISLIX RT                RETURN

*  FOLLOWING LINES ARE FROM THE DATA SECTION OF SOURCE CODE
*  DATA FOR PRODUCING THE MAIN MENU
MENDAT BYTE 19           LENGTH OF TITLE
       TEXT 'GOLF SCORE ANALYZER' TITLE TEXT
       BYTE 8,13         NUMBER OF ITEMS, LENGTH OF FOLLOWING TEXT
       TEXT '1  ADD ROUNDS' TEXT LINE
       BYTE 12           LENGTH OF TEXT FOLLOWING
       TEXT '2  LOAD FILE' SECOND TEXT LINE
       BYTE 14
       TEXT '3  DELETE DATA'
       BYTE 17
       TEXT '4  ANALYZE SCORES'
       BYTE 12
       TEXT '5  SAVE FILE'
       BYTE 19
       TEXT '6  ADD/EDIT COURSES'
       BYTE 17
       TEXT '7  REVIEW COURSES'
       BYTE 15           LENGTH OF LAST TEXT LINE
       TEXT '8  EXIT PROGRAM' LAST TEXT LINE
* DATA FOR PRODUCING THE LEGEND AT BOTTOM OF ANY MENU
SELEC  BYTE 16           LENGTH OF LEGEND
       TEXT 'SELECT BY NUMBER' TEXT OF LEGEND
* LOOKUP TABLE FOR BRANCHING OUT FROM MAIN MENU
* EACH DATA ITEM AT MAINBR GIVES AN ADDRESS OF A LABEL TO WHICH CODE
* BRANCHES WHEN A SELECTION IS MADE FROM THE MAIN MENU
* THE LAST ENTRY IN THE TABLE IS WHERE THE CODE BRANCHES WHEN
* FUNCTION-9 WAS STRUCK.  IN THIS CASE, WE EFFECTIVELY IGNORE THAT
* KEYSTROKE BY BRANCHING TO LABEL KYIN, WHICH SIMPLY WAITS FOR ANOTHER
* KEY TO BE STRUCK
MAINBR DATA NRIN,FILGET,SELCRD,SELCRS
       DATA FILSAV,NCIN,CRSLST,BYE,KYIN
* MISCELLANEOUS DATA ITEMS
NUMASK DATA >30
NOITEM DATA 0
SCRLI  BSS  SCRWID
OFFSET BYTE >60
SPACE  BYTE >20+>60     <--eof-->                    -[END]-
```

TI USER'S GROUP HOLLAND——80 COLUMN CARD.

PRESENTED BY TON DAMEN

Translated by Leon Burger (the flying Dutchman, June 1992)

Additional changes and notes by Derek Hayward of the TI99/4A USER'S GROUP (UK).

August 1992.

NEWS FROM THE HARDWARE GROUP

You must have been wondering what the Hardware Group have been doing over the past few months? We have been working very hard, so it is with great pleasure we announce the completion of our own 80 column card.

The word 80 column is not actually the right word, because the 80 column mode is only a very small aspect of the possibilities the video chip we have used can offer. You will be impressed with its graphical screen modes and the stationary screen presentation. The RGB-Encoder (which most TI users have instead of a TV.) is very much inferior when compared to the colours and brightness the video chip gives. I immediately sent my RGB-encoder into retirment.

HISTORIC OVERVIEW

In 1988 we first noticed the development of the 80 column card, which was being designed and produced by Megatronics (in Germany) and later on by Digit (in the USA). Both these cards used the Yamaha V9938 video chip. You will also find this same chip in the Myarc computer, the MSX-2 also the Nintendo games computer.

The difference between the German and American cards is that the Megatronic's card must be inserted sideways into the console and the Digit is fitted into the PE-Box. Both constructions have their good and bad points, the German design requires no modification to the computer motherboard. the disadvatage is the console becomes 10 cm wider and a separate feed/power supply is required. The Digit card just plugs into the PE-Box, but the big disadvantage is that two modifications to the motherboard must be made.

Both cards have never been great sales successes because especially the German card was very expensive and of low quality.

PLEASE NOTE
This article is about three pages of circuit diagram that the magazine editor never printed. There was no update or any further news of this project in TI*MES.

The American card should cost $200, a normal price in the USA, but for us here in Holland, this would mean a price 1.7 times higher because the value of the Valuta. Therefore we have been looking for a Dutch alternative.

At first (1988) it did not seem worthwhile to design and build our own 80 column card because the price of the parts were to high, (in 1988 a 4464 chip was fl 25.00, at this present time it is now fl 5.00) and the video chip was very hard to find and when we did it was very expensive.

With some pushing from some of the other Club members, we reviewed the situation to see if there was the possiblity of manufacturing/building a card as the price of the chips started to come down. The biggest problem at this time (1990) was and still is the advalability of the V9938 chip. Via some contacts in the trade we came by some second hand chips, this gave us the opportunity to consider the development and constuction of the board.(NOTE 1) The circuit design is our own and meets the following demands:-

1) The board must be completely hardware/software compatible for the TMS 9929.

2) It must either fit the TI PE-Box or Mini PEbox.

The basic design dates from April 1990, soon after this two test boards were constructed, because of the very hot summer no work was carried out during June-September. By the meeting in October 1990 we had finished the two test boards. At this stage they did no work completely satisfactory, we still had problems with the oscillator circuit (we discovered this afterwards). Following two extra sessions with part of the hardware group all the remaining problems were solved. So now we have two good working boards.

THE SCHEMATICS DIAGRAMS

As stated, the circuit is built around the V9938 (U13) video chip, which is a 64 pin shrink DIP IC. This chip has the ideal characteristics as it is 100% compatible with the TM 9929. This means that the chip can inplanted into the existing TI hardwear with no problems.

The circuit consists of the three parts:-

Buffer and memory decoder logics (page 2)

Video chip and video memory (page 3)

Monitor connections (page 4)

Page one is missing this is just a title page, used to transform

NO CIRCUIT DIAGRAMS WERE PRINTED

the schematic to print layout.

BUFFER AND MEMORY DECODER LOGICS.

Reference to page 2 you will see the IC's U4,U10 and U11, these separate the signals from the PE-Box I/O bus. The chips U1,U2,U5a and U6c will take care of the correct positioning of the 9938 and the Eprom in the memory map of the TI-99/4A.

As you might know, the video chip (TMS9929) is in the memory range >8800->8FFF. Because the V9938 has several read and write registers, this chip's memory map is extended. In the table below the differences can be seen.

TMS 9929	V9938	DESCRIPTION
>8800	>8800	READ DATA
>8802	>8802	READ STATUS
>8C00	>8C00	WRITE DATA
>8C02	>8C02	WRITE ADDRESS
	>8C04	ACCESS PALETTE REGISTERS *
	>8C06	REGISTER INDIRECT ADDRESSING *

* SEE YAMAHA'S TECHNICAL DATA BOOK FOR DETAIL INFORMATION.

The main feature of this schematic is the function of C15, this, at first sight may seem useless, this is not the case, it's of vital importance for the whole of the circuits function, as there is a BUG in the hardware of texy!

The A15 signal at the beginning of the memory cycle is not completely stable, therefore the A15 is a pseudo (fake) signal, consisting of the CRU-OUT clock and a dummy A15. Because the TMS9900 is a 16 bit microprocessor it will process at anytime 16 bits, this applies for each byte operation as well. This suggests that an A15 signal is needed to indicate if an even or uneven byte has to be processed. That combined signal gives problems to the new video chip. A number of times the VDPRD line will be active because A15 is not stable during a read operation. Because of this, non-authorized data will be read from the video chip and will result in the wrong data.

Although the V9938 is 100% software compatible with the current video chip we still require an EPROM chip in the circuit. The video board would still work satisfactory without the eprom, but as there is a BUG in GROM 0 on the motherboard, the title screen will be misformed. Solving this, is one of the functions of the eprom. This problem is also known to exist in the TI BASIC and CHESS MODULE CHIPS. The solution to this problem is not to change the hardware of the computer, but rectify it by including the changes in the software. We can confirm that EXTENDED BASIC and

EDITOR/ASSEMBLER are working without any problems.

The eprom is situated at the normal position in the memory map, at address >4000. This eprom can be switched by a selector switch, thus giving another CRU address. This control is taken care of by U7,U8, U6A/B and SW1. Be WARNED only ONE switch can be closed at any time, if not problems may occure with the hardware. You must also note that other cards might also be using the same CRU address. e.g.: the Disk control card uses CRU address >1100. The following list shows the selectable CRU address:-

SWITCH NUMBER	CRU ADDRESS
1	>1000
2	>1400
3	>1200
4	>1600
5	>1100
6	>1500
7	>1300
8	>1700

My advise is to switch the card to lowest CRU-address >1000, because of possible video memory problems, which depend on the type of software in the eprom.

The power supply stabiliser U12 supplies the right voltage. This transistor must be mounted onto the board using a U shaped cooling plate.

VIDEO CHIP AND VIDEO MEMORY.

This part of the circuit seems complicated because of the many connections but the idea is actually quite simple. Most of the work is done by the video chip U13.It takes care of the RAMS memory refresh, the reading of the light pen and/or mouse, digitizing of data and the supply of signals for the monitor. The X1 crystal takes care of the video's timing. The frequency is near the upper value of Yamaha's advised value. This has been done for two reasons:
By choosing the highest possible frequency, theoretically the screen image will be improved. (NOTE 2)

The availablity and cost of the crystal. Because this part has a standard frequency and cost only f13.00.

The VIDINT signal deserves extra attention, this must be connected to the mother board of the TI-99/4A. Therefore a wire has to be connected between the PE-Box and the console. Another way to achieve this is to use one of the cores which is not used on the ribbon cable which runs between the two units. (NOTE 3)

With reference to the K1 connector, all the signals not used by

U13 have been connected to this pinout. So in the future the board is ready for other hardware to be added. These signals are for the light pen, mouse and colour bus. The last one can be used for digitizing video pictures.

A good tip for Megatronics card owners, the vertical rectangles on the screen will decrease if a condenser like C23 is connected to pin 33 of the video chip V9938. I must comment, our two prototypes do not have any rectangles or stripes on the screen!

MONITOR CONNECTIONS

On page 3 you will see the relative simple circuit to interface with the monitor. The basic circuit is repeated five times for each signal, the transistors acting as buffers between the signal from the V9938 and the monitor. These circuits makes our graphics card suitable for normal RGB monitors (and television) with SCART type connections. (NOTE 4) The composite video signal is an extra, and is black and white suitable for monochrome monitors.

Although the V9938 can control direct so called PC-monitors (EGA/VGA with independent horizontal and vertical sync.), this mode has not been assembled nor tested. This can be achieved by adding an additional transistor stage to the HSYNC.

The 9 pin DIN plug is prepared for connecting to a PC and for normal type monitors. With pin out K3 it is possible to choose to transmit the composite video signal or a 5 volt supply which is required for Scart plugs on some televisions.

MODIFICATIONS TO THE MOTHER BOARD OF THE TI-99/4A

Because of the circuit design of our graphics card we must do two modifications on the motherboard.

1. The multiplexer on the board has to be modified. In the current situation you can not do a read action in the memory's video range (>8800->8802) via the I/O -bus and thus the PE-box. Therefore the printed circuit track on the mother board, which is connected to pin 13 of the 74LS138 -U606 (see arrow on photo 1) (NOTE 5) has to be cut. The best place to do this is near to U606. After this a wire connection must be made from pin 13 to the +5V line. e.g. at pin 16 of the same chip U606.

2. Because of the above modification, the TMS 9929 chip does not provide a good interrupt signal any longer. Therefore you must carefully lift the TMS 9929 out of its socket and bend pin 16 outwards, so that when you refit it, this pin lays out the side of the socket. When the chip is replaced it is only used for the supply of the GROM CLOCK.

Now a wire has to be connected to a plug and K3 on the new card. The position from which this connection is made on the motherboard is shown by the arrow on photo 2. (see FIG B) You will see when you look closer to pin 16 on chip TMS9929 that there is a spare hole adjacent to the pin. Into this hole you solder the wire. With this connection made the VIDEO INTERRUPT signal of the V9938 is transmitted. (NOTE 3)

THE RESULTS

We now have two good working (MINI) PE-box versions, all the programs we have for 80 column are working OK without any errors.

But we must make the following comments:-

- The sound signal is not yet transmitted via the board; you still have to do this yourself. (NOTE 3)

- Without a modified EPROM, the title screen of TI BASIC and CHESS MODULE are not readable. Though they operate OK. (NOTE 6)

- Because there is no standards (of course) for the different video cards, it is possible that some programs may not work with this card (the other way round aswell)
These problems occure because the programs direct the V9938 incorrectly, the displacement of the disk buffers in the VIDEO RAM and the wrong software in the EPROM.

WHAT TO DO NOW?

The MINI PE-box versions are working without any problems, and are except for the connector and supply stabiliser, identical to the TI- PEbox version. (NOTE 7 and 8)
But first I want to develop two TI-PEbox versions. This will not take long as there is little difference between the two units.

It is very tempting to make this print available to any body, but there are a number of reasons why we should not do this at the moment:

- The availability of the V9938 chip and its price. There are a number of different possibilities but no serious solutions have been found yet.

- I do not have a good price for the manufacture of the printed circuit board.

- Because there are a number of BUGS in the V9938, and now there is an improved chip V9958. Some of the bugs in the V9938, have been solved in the V9958, it has 19200 colours compared with 512 in the V9938. Also the hardware commands have been extended.

- Assuming one can obtain a circuit board, one must still make the two earlier mentioned modifications to the mother board, this could be a problem for most club members.

I can give you an approximate cost of the unit. (NOTE 9)

PART	APPROX PRICE(DUTCH GUILDERS)
Printed Circuit Card	75.00
I.C. V9938/V9958	75.00
MEMORY CHIPS	40.00
REST OF PARTS (excl. EPROM)	25.00
PLUGS etc.	25.00
OPTIONAL EPROM	15.00
	TOTAL... 255.00

To make it clear, these amounts are global prices, the more that are purchased the lower the price.

Although one may have to pay quite a high price for the printed circuit board, I think this is one of the few additions which will pay off in the end. I can't do without this up grade in my TI.

In the next edition I hope I will be able to publish a purchase form.

Finally I would like to thank the co-thinkers and builders (David, Piet (England) and Ton) for all their effort and time in ensureing the success of this project.

From this lonely place on a stormy evening in November, behind my key board, I would like to wish you a good start to 1991 with your TI-99/4A.

Ton Damen.

NOTES BY DEREK HAYWARD

From the outset I must make it very clear that these notes are to be read inconjunction with the original Dutch literature (translated). Its hoped, more information will encourage more TI users to stay with their machines and at the sametime enhance the performance of the TI-99/4A.

As stated by Ton Damen in his text, this card is one of the most impressive and final improvements you can make to your computer.

It must be understood that the comments I have made are my own observations (while typing in this text on my TI), from the translation which Leon Burger (a student over from Holland on Industrial Training with the Company I work for) completed.

At this date (January 1993) no board has been constructed here in the UK. The plan is to build three prototypes, and as this work proceeds make further notes on the construction and any problems we may encounter.

NOTE 1.

The supply of the chip V9938 has been no problem here in the UK. A telephone call to Yamaha-Kemble and my order was placed. The information to order is as follows:-

YAMAHA-KEMBLE MUSIC (UK) LTD.
SHERBOURNE DRIVE
TILBROOK, MILTON KEYNES.

TEL 0908 366700

Order part REF XA037001 VIDEO CHIP V9938.

NOTE 2

This crystal at first, seemed a problem, the frequency stated is NON standard for the UK. But while checking back over old documents concerning the chip and 80 column card, I found a report in MICROpendium, dated September 1988 by Tony Lewis, in which he states that Yamaha had designed the chip to operate with a crystal at 22 MHZ (I do not have the manual on the V9938). This crystal is standard and can be easily purchased.

NOTE 3

The wire link between the motherboard and the PE-box could be a problem. While considering this, remember a further wire will be required for the sound signal. If you want the TI sound - first your monitor must have an audio amplifier fitted into it. Then the cable/flex connecting the computer (PE-box) to the monitor must also carry the AUDIO signal. With the new card fitted into the PE-box then its assumed that one cable will be used for this purpose, therefore the AUDIO signal must be connected onto the 80 column card from the motherboard, before connecting the 80 column card to the monitor.

There is no out going AUDIO signal sent along the multi-core ribbon cable (hose pipe), between the computer and the PE-box. So two wires are required, one for the VIDANT and the other for the AUDIO signal. You could connect a two core screened cable between the two units, independent of the 'hose'. Or alternatively (on checking the Technical Data Sheets on the TI), I see TI use 4 cores of the ribbon cable for the earth return between the two units, core numbers 21,23,26 and 27 at the I/O socket of the computer. On arrival at the PE-box end they become cores 3,5,7,20,27,47,49 and 53. Maybe 2 of these cores can be separated from the common earth connection and used for these two signals.

I do not have this problem as my TI has been converted by fitting the RAVE 99 Keyboard interface, so now I have the standard PC 101 keyboard and no computer console. By adding an additional metal panel on the back of my PE-box (making it slightly deeper by 2.5 ins) onto which I have mounted my motherboard still screened by its metal cannister. With the unit so close to the back of the PE-box there is no need for the 'hose' connection, just a short ribbon cable is used. Back to the 80 column card, as the new card will be just in front of the motherboard, I will run a short length of two core screen cable between the two boards. The AUDIO signal being taken from the back of the 6 pin DIN socket on the motherboard.

NOTE 4

In another write-up (supplied on disk via the TI-USERS (UK) CLUB disk library - thanks to Stephen Shaw) DIJIT have made comments as to the type of monitor that must be used with their 80 column card. The point made, being that a monitor with a reasonable high resolution must be used to get the benifit of the more detailed screen format. I plan to use my TATUNG model TM01, which may or may not be good enough. I will report on this later.

NOTE 5

As both the photographs in the oringinal manuscript were very poor reproduction, I have checked the component layout on the motherboard and have made two illustrations, they show exactly at which points the mods must be made on the board, see FIG A and FIG B.
In Ton's write-up he has designated the chip as U606, while the official technical data manual from TI is a poor print, I think this chip should be U506. This is further confirmed by Eric Zeno (designer of the Zeno Board) he also numbered the chip as U506. There will be no problem with this change in chip numbers providing you use the illustration FIG A and B to find the location of the parts on the motherboard.

NOTE 6

If there is still a problem with the screen presentation on TI-BASIC, its hoped to change the program which is burnt into the EPROM. Further news on this later.

NOTE 7

Following a telephone call from Ton Damen (in Holland), he has confirmed the design of the printed circuit board included in the oringinal manuscript (Ref TI80COP6 PO and P1) is for the standard TI PE-box. As to the stabilised power supply, that is still not resolved. I make this comment because the assembled board which Berry Haman brought to the TI-USERS AGM had an additional diode fitted to one of the legs of the power transistor, this must be checked.

NOTE 8

While checking out the board design to establish whether it was designed for the mini or standard PE-box, I noticed that the component layout board TI890C0P6 P2 had some minor mistakes when compaired with the Schmactic diagrams. I dont think this is a major problem, but I have revised the layout to match the schematic diagram. See FIG C.

Reference to the back face of the board (solder side) on DRG TI80COP6 P1. Part of one track is missing (this might have been lost during printing) in the upper top R.H.corner, close to the 3 holes into which you solder Q1 transistor. I can confirm the missing track is connected to the hole directly above it. This point inturn connects to the emitter of Q1 and resistor R4 (per my new layout).

NOTE 9

Listed is the price I have, or expect to pay to complete my 80 column card. Please note (as Ton mentioned in his report) the major problem will be the manufacture of the circuit board. Mike Goddard has made enquiries, and to date the information is this- 40 to 50 boards must be made with the cost coming out at about £25 per board. The problem is what are we going to do with 40 or so boards??

PART	APPROX PRICE (£)
PRINTED CIRCUIT BOARD	25.00
I.C. V9938	25.17
MEMORY CHIPS 4464	20.16
CHIPS U01 to U12	10.56
EPROM U3 2764	3.45
Q1 to Q3 BC547	1.20
HARDWARE ETC.	18.61
TOTAL..........	93.59

Most of the above prices are based on those quoted in the Maplin Catalogue 1992

Derek Hayward.........January 1993.

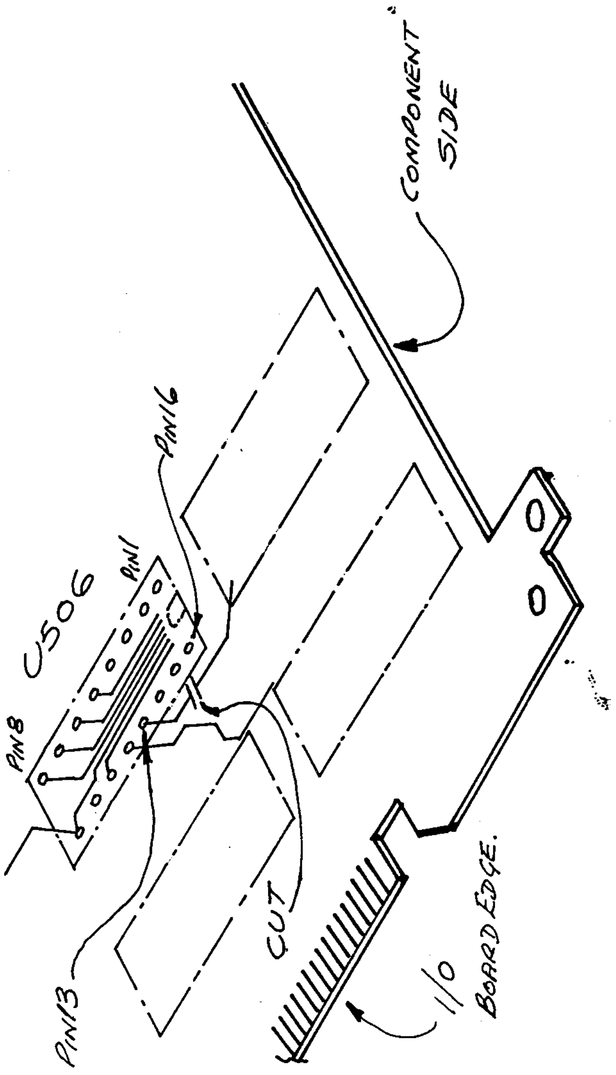


FIG A

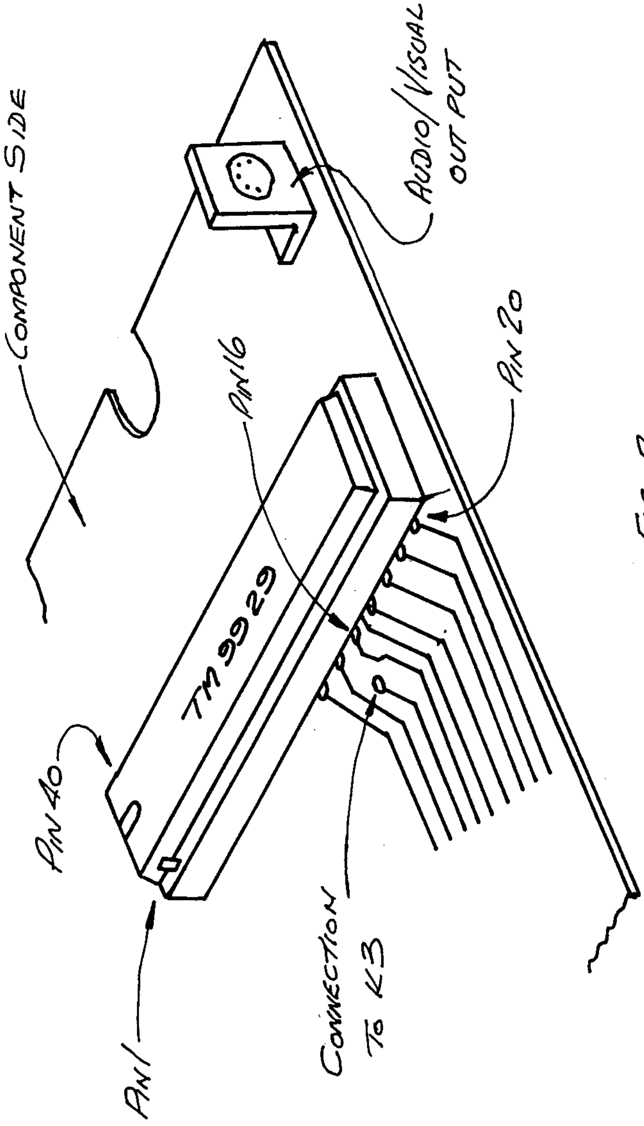
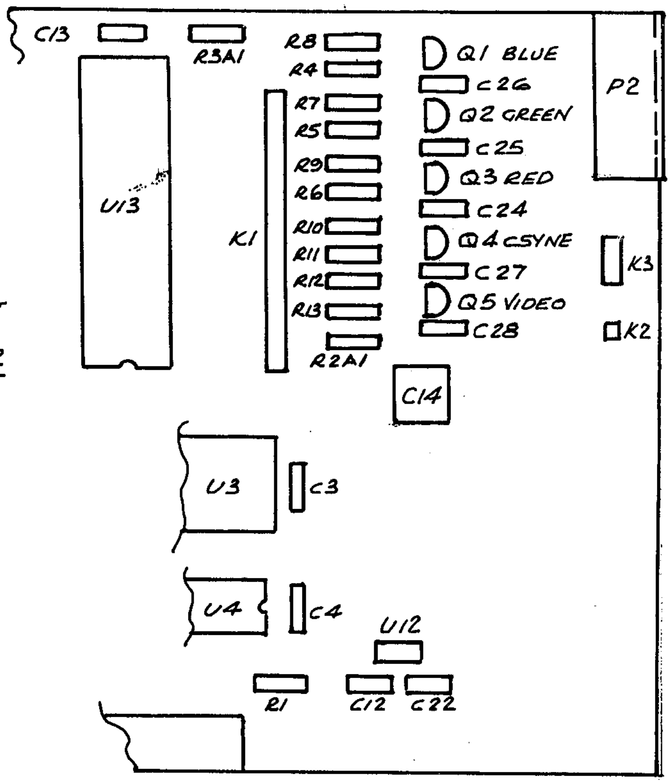

COMPONENT SIDE
AUDIO/VISUAL OUT PUT
PIN 16
PIN 20
TM 9929
PIN 40
PIN 1
CONNECTION TO K3

FIG B

REVISED COMPONENT LAYOUT

REF DRG TI80COP6.P2

FIG C



TIPPSY

This is a small artical which we hope will give you little tips so that you can use you machine in a better way. The tips of course can't come from one source, so we ask you to write into the editor with YOUR tips.

Here are a few to start the ball rolling.

TI WRITER (T.S)

Have you ever tried to print out a multi document through the PF (print file) command, and suddenly realised that on page two of four there is a mistake. Or you want to print out only a paragraph, or prehaps save a line to disk without CR marks in them. Well you can. I have looked at the TI WRITER manual and can find no reference to the following command in it. I found it out by mistake when tried to save a file to disk.

What you do is take note of the line numbers (inclusive) that you want to print or save and then return to the E(dit) Mode, which is on the top line. Then type PF (Print file). You will then be presented with PIO or RS232.300 or similar. In my case I use PIO. You then back space over the PIO to the begining of the entry field. Say you want to print line 20 through to 40, you will then type 20 40 PIO. Press return and the lines 20 to 40 are only transmitted to your printer and printed. For a single line type the single line number only. eg 20 PIO. Remember however to put the spaces between as given in the above examples. If you want it to disk, replace PIO with the DSKx. command.

TI EXTENDED BASIC

Did you know that you can chain link cassette programs? What is chain link you may ask, well it is where you can write a basic program from EXB which can load and run another program from your cassette recorder. When this happens the original is over written and the new one then replaces the old and then runs. You must however be at the begining of the program on the tape for this to happen. You do this with the RUN command. If you branch in your program to RUN "CS1" you will be able to chain for as long as you like. This can increase memory storage. The only down side is that tape takes a long time to load.

DEMO to the above TIP.

```
100 CALL CLEAR                   }  Save to CS1 first.
110 PRINT "THIS IS PROGRAM 1"    }  Rewind AFTER saving second
120 FOR A=1 TO 1000 :: NEXT A    }  part below and load this
130 RUN "CS1"                    }  program.

100 CALL CLEAR                   }  Save to CS1 second
110 PRINT "THIS IS PROGRAM 1"    }
120 FOR A=1 TO 1000 :: NEXT A    }
130 PRINT "FINISHED"             }
```

MEMORY FULL   FUNCTION QUIT!!!!!!!!