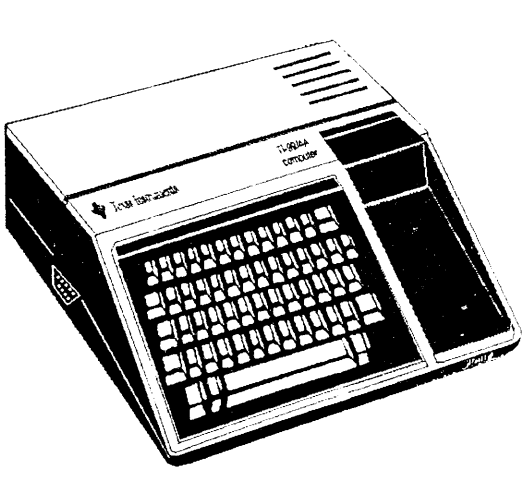
TI*MES

SPRING 1998

60th Issue



Position	Name	Address	Telephone
Chairman & Sysop of group BBS	Trevor Stevens	249 Southwell Road East, Rainworth, Notts. NG21 0BN	☎ 01623 793077 🖳BBS: 01623 491282
Vice Chairman	Mark Wills	41 Broxtons Wood, Westbury, Nr. Shrewsbury, Shropshire. SY5 9QR	☎ 01743 884869
General Secretary & Co-editor	Richard Twyning	24 Peel Road, Mansfield, Notts. NG19 6HB	☎01623 627670 🖹 FAX/Voice Msg. 01623 453934 (9am-5pm) mobile: 0467 445658 🖹Mobile FAX: 0467 449009
Co-Editor	Ian Pare	10 Sotheby Avenue, Sutton-In-Ashfield, Notts. NG17 5JX	☎01623 552549 (BT) ☎01623 452728 🖹FAX: 01623 452729
Treasurer	Alan Rutherford	13 The Circuit, Wilmslow, Cheshire. SK9 6DA	☎01625 524642
Disk Librarian	Stephen Shaw	10 Alstone Road, Stockport, Cheshire. SK4 5AH	
Module and Cassette Librarian	Francesco L. Lama	14 Granville Court, Cheney Lane, Oxford. OX3 0HE	☎01865 721582
		20 Oak Avenue, Romiley,	☎0161 4307298

CONTENTS

THE CHAIRMANS CHAIR

BY T.STEVENS

APRIL 1998

Well I write again with some more news on the Group. Firstly the AGM. For those that go onto the BBS you will find in the NEWS LETTER area a full description of the dat a etc. However those parties that do not the date and place is as follows:

10am Saturday 30th May 1998
St John Ambulance Station
Trinity Street
Derby

The short news letter also gave this information but the TI*MES is the place to find it. However the News letter also has the MAP!!! We will be discussing most of the normal group problems. However if you have anything that you wish to ask or shout about, come down to the AGM and have some fun. At the show I will be showing the new windows 98 on a PC so anyone out there who wants to see the FULL working version then come along.

Some time ago Simon BURFORD said that he had a writer and wanted to put together a CD ROM of TI files which would include PC 99 files of most of the groups programs. I told him I had most of the mainstream files on disk. I thought it was such a good idea as I could back up all my stuff on CD ROM and be safe, as floppies do tend, over a period of time, to loose data. Simon did not have a Full TI System so I lent him mine, which he has had for sometime so that he can do the conversions and ARK files onto the said CD.

So with Simon I will be adding the other shareware files I have on the BBS to the CD, but leave it open if we require to add more files. The disk should hold a lot of data and will be available on the BBS soon. In addition I hope to sell copies of it to group members only, for GROUP funds. If anyone wants a copy that is. I have not set a price but I expect around the £25.00 mark. There is so much on there that the cost of the disk is peanuts to what it should cost. However to get the data back off you will require A) a TI SCSI card with CD ROM drive, or B) A PC with CD ROM and a direct cable between it and your beloved TI using a comms program on both machines to pass the data. When passed un-ark with archiver in the A) or B) set up to a disk.

I will also be answering questions and doing some BBS work. So you can log in and have some fun with the BBS. If you already use the BBS you can try out some of the fancy stuff which I have now put on line. To enable me to do this I will be putting the **Old BBS** online on the AMIGA which will act as an "I am here server" just for the day.

Talking of the BBS, I have had trouble AGAIN! With the damned thing. However I now have Wildcat VERSION 4.04 installed. This replaces the version being used up to the 17th April 1998. The new system is more friendly and I hope will serve us better than the old one.
The new system has a better interface and you can get round it with ease. The only strange thing is when you **locally** upload, the machine asks for a MASK. This means that you have to path your source file on your machine if you are uploading locally. However this will not cause you the remote user any problems whatsoever. This system, with some thinking about will allow 40 col displays so we can get the TI USER to read the menu properly.
I have discussed with Richard Twyning about getting a new BBS such as POWER BBS which I am told is very good. However it is the cost. So this will be discussed at the AGM.
Some of the benefits with this BBS is that it is Internet compatible and could link the TI GROUP to the outside world with EMAIL. In addition it could pull information down off the net when required. However this is still the old question of cash as everything has its price.
The MODEM the group has, has got to a point where for some users it is too slow. So this option of updating will also be discussed at the AGM. This also will be an extra expense but the new version could be off set by the sale of the old one. However it all lies round the fact that the BBS is run on a 486 and the MODEM must have software to run in dos and not be Pentium CPU dependable. However I think I can get round this problem.

Of late I have not touched a TI as it has been away with Simon BURFORD (see above for reason).

How has your programming been coming on? Managed to sort through my articles and do a little of your own work? I hope so. No one has said one way or other if they like the articles however I will crack on and do some more for you. So here goes.

ASSEMBLY PROGRAMMING

The last time we discussed the how to put the name into the program and how we used the REF DEF table. We now move onto controls and inputs. IE The keyboard and Joysticks. In Basic programs we use a separate routine in the form of a CALL to use the various input routines. Such as INPUT, ACCEPT AT, CALL JOYST etc. In assembly we only use one.
The assembly routine is called KSCAN or keyboard scan for short. This utility in assembly is stored at the memory location > 6020. This routine work the same as the CALL KEY routine in Basic. What it does is take the input from any device check to see if the input is valid printing them on the screen or storing them in a memory location until ENTER key is pressed. Thus we have an INPUT in its various forms. So CALL KEY and CALL JOYST are easy to write as well.

Preparing the KSCAN Routine

The KSCAN routine needs to know the keyboard device number when it is called, so this value has to be put into memory before the branch (BLWP) to the KSCAN is executed.
This value is the same as the CALL KEY(0,K,S) 0 being the value we are talking about.
So if you look at the TI Extended Book there is the full values and keyboard selections list on page 201. In this case 0 means the standard keyboard. 1 the left side and 2 the right side and so on. So we use 1 and 2 for our joysticks. This value has to be put into the memory location >8374. To

place the 0 just clear the memory address.
i.e. (CLR @>8374)
To place another value, load it into the left (most significant)byte of the register and then move that byte into the corresponding memory location.
i.e. LI R7 >0200
MOVB R7,@>8374

The above line places 2 (reading Joystick No 2) into byte >8374
Once this has been done, you can then branch to the KSCAN routine in location >6020
i.e. BLWP @>6020

If a key is pressed when the program runs, its hexadecimal ASCII code value is placed in BYTE >8375. You can detect if the key has been pressed by simply checking the *status* byte, at the location >837C, just as you do with the status in the Basic CALL KEY(0,K,S)
S= Status. In BASIC if this variable is 0 then no key has been pressed. If it is -1 a key has been pressed. In assembly it is different, but similar if you know what I mean!!!

Checking the Status BYTE
Before we go any further we need to be reminded that a BYTE is divided into 8 bits. These 8 bits, are numbered 0 to 7 (Base 0). The numbering is done from left to right (the convention used by TI) maybe set (contain a 1) or reset (contain 0) the following example shows bits 2 and 4 set and the rest reset. Just like a light switch!!!

0 1 2 3 4 5 6 7
0 0 1 0 1 0 0 0

The values of each set bit double as you move from **right to left**. Bit 7, the bit on the far right has value of 1 and 0 has a value of 128.

0	1	2	3	4	5	6	7
128	*64*	*32*	*16*	*8*	*4*	*2*	*1*

To get the total value of the bits set just add together the bits set. So if 2 and 4 as above are set that is 32 and 8 which equals 40 (32+8)

What makes the status byte so important for the KSCAN routine is that bit number 2 is set if a key IS pressed. Otherwise, it is reset. The other bits in that BYTE do not interest us. By checking the second bit in the status byte, you can know when a key is pressed or not. To do this checking you will use the COC (Compare Ones Corresponding) instruction. This instruction compares the bits set in the first operand to the bits set in the second operand. If all the bits set to **1 in the first operand** are the same **set to one** in the second operand, the operands are considered equal. It does not have to be reciprocal. For example these two bytes are considered equal by the COC instruction:

First op Byte 10100100
Second Op Byte. 11101101
All the bits set in the first byte having corresponding bit set in the second byte. If the bytes were reversed (the first becomes the second, and the second the first):

11101101
10100100

The bytes would be considered unequal by the COC instruction because not all the set bits in the first byte have a corresponding set bit in the second byte. So remember it has to be set in the first and the second to ONE to make it.

So how can all this help us? If you create a byte with only the second bit set like this

0 0 1 0 0 0 0 0

and compare it to the status byte with the COC instruction the bytes will be considered equal if the second bit of the status byte and different if it is reset, regardless of the other bits. Then you will know that if the two compared bytes are equal, a key was pressed. If they are different then a key was not pressed.

This instruction places a byte with only bit two set in the left byte of memory location MR

```
MR DATA >2000
```

>20 in Hexadecimal is 32 in decimal that is the binary number 00100000 our value needed. The right byte of the location was filled with zeros so as not to have any effect on the comparison. Be assured that that nothing is amiss with you if you feel a bit confused with this last section. If you do just read over it again slowly and all will be revealed. THINK ASSEMBLY and you will crack it. For now it is enough to understand the method that the KSCAN performs.

The labeled memory location with the comparison byte is usually placed at the end of the program where it will not be executed by any instruction. Then, when you check whether a key was pressed in the KSCAN loop, you can clear a register to zeros, then move the status byte into the register, and then compare (COC) it to the labeled value like this:

```
CLR R1               CLEAR R1
MOVB @>837C,R1        Move status byte into left byte R1
COC @MR,R1            Compare bits set to 1 of the value set at MR to bits set
to 1 in R1
```

If the bits match the key was pressed, else not pressed.

Displaying a message

The next example demonstrates the technique to perform a KSCAN, checking the status byte with the COC instruction. (note the way the KSCAN operation is written) The following program waits for you to press a key and then display the message KEY DETECTED on the screen. So here goes....

```
7D00          LWPI>70B8          Load memory area for registers
7D04           CLR@8374                        Clear Byte >8374 standard
K/Board scan
7D08          LP         BI WP @>6020      Branch to Scan Routine
7D0C                     MOVB @>837C,R1  Move status byte into R1
7D10                     COC @BT,R1        Compares set bits of the
comparison
```

```
                                            value added at the end of the program with
                                            the label BT, to the set bits of the word R1

7D14        JNE LP                          If COC not equal no key pressed goto LP

7D16        LT R0,298                       Key pressed Load screen location for message

7D1A        LI R1,TX                        Load position of text in memory

7D1E        LI R2,12                        Load length of the text (12)

7D22        BLWP @>6028                     Display text on screen

7D26        B*R11                           Return to easy bug
7D28        BT      DATA >2000              Comparison value for the KSCAN loop

7D2A        TX      TEXT 'KEY DETECTED'


7D36                                END
```

Now run this program. The message will be displayed on the press of any key.

Well that is about it for this time as I have run out of spare time to do any more. However we will be doing some more assembly next time which will enable you to write a program that makes your computer a simple type-writer. We will look further at the INPUT operations, repeating keys and joystick routines which will build into a program that will enable us to move a cross around on the screen.

I do hope you have understood what has been happening so far. If you have not re read the explanation if however you still do not get what is going on then give me a ring at the telephone number at the index in TI*MES. Or better still wait till the AGM and we can discuss it there.

FCTN QUIT   MEMORY FULL   TURN OFF BRAIN!!!!!!!!!!!

TI-99/4A User Group U.K.

Proudly presents

The 13th

Annual

TI-TREF

International
TI-99/4A
Convention

The Beeches Hotel and Leisure Club, Wilford Lane,
West Bridgford, NOTTINGHAM, NG2 7RN

In the heart of Robin Hood Country,
Nottingham, England
Fri. 9th to Sun. 11th of October 1998

From the keyboard of Richard Twyning

On Sunday morning Trevor and I had breakfast with Mr. Muys and his wife. Before we left the table, Trevor nipped in to see if the TI room had been opened, and when he returned, he gave me the bad news that the chappy who had all the cartridges had already left. The tables he was using were empty.

Luckily on the Sunday morning there was another family there who were selling cartridges, and I cleaned 'em out. I think they were German, and they were trying to help there son with his English lessons, because they had put him in charge of selling the cartridges!

They were charging 3 Guilders a cartridge, so I had 20 of the Shockers!, for which Ian paid me £20, which wasn't bad because the haul included THREE TI-EXTENDED BASIC'S WITH MANULS!!!!! Because I had bought so many cartridges, they also let me take an extra one for free!

I suppose I must thank Trevor for selling one of my TI T-Shirts to Sven Dyroff for 25 Guilders, otherwise I wouldn't have been able to afford the cartidges!

It was just after dinner time when we said our goodbyes, after Sven Dyroff had collected all of our details for his definitive list of mainland Europe TI users. I also collected all the necessary information I needed from Berry Harmsen.

DAVE SCRIVER

says

HELLO

Greetings to all members of the UK TI users' group! My name is David Scriver.

I am an American living in Cumbria. I am 30 years old and a long time enthusiast for the TI-99/4A home computer.
I recently contacted some of the officers of the UK group and asked what I could do to contribute to the group as a whole. It was agreed that because I no longer have a TI, and do all my /4A work using an emulator, that perhaps I could write a column dedicated to emulators and those of us who use them. Now then, to the purists and alarmists out there who fear the emulators, all I can say is that I am not trying to weigh the use of "true" TI hardware against the use of emulators.

My intention is only to make another aspect of the TI community come alive and by doing so hopefully breathing some new life into our collective hobby.

Introductions and disclaimers thus said, may I welcome you to my first in the TI emulator series of columns. I will begin by explaining what an emulator is and why they are used.

An emulator is a program that is run on one computer such that it can execute the machine language of another. Every computer has a native machine code that its main processor(s) understands. No matter what language a program may be written in, it eventually has to be translated into machine code in order for the computer to execute it. TI-Basic, for example, must eventually be translated into TMS-9900 code in order for it to be run. The TMS-9900
being the central processing chip used in the TI-99/4A home computer. This chip dependency is the reason why most programs written for one computer won't run on another. With an emulator, however, a given computer can "translate" the native code from one machine into the native code of another. There are other considerations as well, such as the video,disk & I/O differences between computer systems. There are some programs that claim to be emulators but unless they can actually take the machine code from one machine, translate it and run it "on-the-fly" then it isn't a true emulator. Why use an emulator? Emulators are generally used to lengthen the life of programs written for legacy hardware when upgrades/ changes are made. It is generally accepted that the "value" of any given computer system is NOT necessarily the hardware that makes it up, but the software (both code and data) that is run on it. Therefore it can be cheaper to use older software for given tasks rather than laying out the cash necessary to upgrade the whole system at once. Another use of an emulator has only recently become a "main stream" item. This use is spurred on by nostalgia/history and means that you can have a "virtual" copy of your favorite old computer running on your shiney new whiz bang dream

machine.
Or perhaps you may be curious about REALLY old machines like the ENIAC or UNIVAC computers that helped to spark the computer age. You can run emulators for these machines to learn how it was all done. The reason for this being obvious. you can't currently purchase/
rent or steal a UNIVAC computer!

One point that must be mentioned. In order for an emulator to run with any semblance of speed with respect to the original, it must be run on a faster, more powerful machine than the original.
This is because of the overhead required to actually do the translation. Having said this, it is possible to have an emulator running on a platform that is fast enough to effectively outperform the original!

In my next column I will discuss emulators that are specific to the TI-99/4A Home Computer. I will also write about some of the pros and cons of running an emulator. Until then I recommend those of you with World Wide Web access to visit a site called The Archaic Ruins at: http://archaic-ruins.parodius.com/

Greetings and welcome to the second in my series of articles on TI emulation.

Before we get started, I would like to just take a second and don my flame retardent gear ("kit" for the benefit of the British readers!).

The purpose of these articles is NOT to weigh the use/non-use of TI-emulators against the TI-users who still use and love the original TI-99/4A computer. In fact, I often say to the "purists" that knock the emulator users, that mimicry is often the purist form of flatery.

My goal is to further our hobby and help keep it alive by revealing another aspect of it that may or may not be known by the entire TI community.

Phew, I sometimes wonder if the pre-amble to a TI article can be larger
than its body! Oh well, let's get on with it and discuss some emulation issues. In my last article I shed some light on what an emulator is and why you would want to use one. In this article we will be briefly covering some of the emulators available for the TI-99/4A Home

Computer. I will also briefly touch on what a TI emulator can/can't do (at least at this time).
What's available?

Currently there are basically two major TI-99/4A emulators available. These are PC99 by CaDD Electronics, and V9T9 by Edward Swartz. While PC99 is commercial software (i.e. it's not free), V9T9 is freely available on the Internet along with its source code. The purpose in this article is not to compare the two. (That will come later on), rather to get you familiarized with the names as they will be mentioned over and over again throughout these articles. Just for the record I use V9T9, cheifly because the price was right! To run either of these emulators you will need to have an Intel PC compatible computer having at least a 386DX/33 CPU. Any slower than that and you will not get acceptable performance with respect to the original TI-99/4A computer. In the case of V9T9 you also need to have images of the ROM data contained in the TI-99/4A console.
The author included a set of utility programs that are run on both the PC and the TI-99/4A that allow you to transfer these ROM images to V9T9.

Once you get either emulator up and running, what can I expect?

Well, first of all, operating the emulator is much like operating a regular TI-99/4A. This is because the code that is being run is native TI. The programs themselves don't "care" what hardware executes them. There are exceptions however. The biggest one being the lack of native TI disk support. What I mean by this, is that you can't place a TI-99/4A floppy disk into a PC disk drive and expect to read/write it. However this isn't as big a hurdle as you might think.
You can transfer most any disk from the TI to the PC using a terminal emulator such as TELCO. You then must run a special PC utility that will be included with either emulator to make the transfered files usable. This is a one time only operation. You can also find "disk images" for

use with
the emulators on the Internet. Disk Images are PC files that when used with an emulator, expand out to contain all the files that would be on a TI-99/4A disk. (This process will be discussed in more detail in the next article in this series). There are other limitations to using the emulators, but they mostly involve the lack of support for third party hardware. What you DO get is a basic TI-99/4A Home Computer with 32K memory expansion, speech synthesizer, disk drive support (although it can be said to not be TRUE support. See above.) and in the case of PC99 you also get P-Code card support. What you DON'T get are Horizon RAM Disks, AMS cards, P-GRAM, cards, 80-column support etc. (although future versions may include some of these features.)

So what can I do with it?

You can run most any program that is written for the TI-99/4A Home Computer. (except those that expect specific hardware that is not yet supported.) This includes cartridge software that has been dumped to disk and prepared for use by the emulator. You can also run: BASIC, Extended BASIC, Assembly Language programs, and Forth programs.
In essence you will notice very little difference in using the emulator once it is running. I have tried using a few of Jim Peterson's tips from the TigerCub programs that had CALL POKES and CALL PEEKS embeded in them and have found no problems with V9T9. I have also tested quite a few different cartridges with V9T9 and have not found a single one to date that would not work. I have even been able to run
most of the functions of the various Disk Managers that exist for the TI
on the Disk Images that the emulators use without difficulty. This includes copying, initializing, deleting etc.
With the included utilities, you can convert text and certain graphics

the PC. For example, you can place your Disk Images on a hard disk drive. This not only allows you access to the vast storage space that a PC hard drive offers, but also allows for very fast file access. Also, because the emulator is running on a PC, you view the graphics on a PC style monitor. The image quality is very sharp as compared to the composite image that you would ordinarily get from a TI-99/4A connected to a Television or a composite color monitor. One other interesting capability is the fact that you can run the emulators under the MS-Windows operating systems which, in turn, allows for multitasking. The
list goes on, but these and other points will be spelled out in more detail in later columns.

Where can I get these things?

V9T9 can be found on a number of local BBS systems or alternatively you can download it from the Internet using FTP at: ftp::// ftp.premierweb.com

PC99 is a commercial product. It can be obtained from CaDD Electronics. Contact Mike Wright at:

CaDD Electronics
45 Centerville Drive
Salem, NH 03079-2674
USA

Or altemtatively E-mail him at:
mjmw@xyvision.com

Ask for current version and pricing information.
In the next column I will discuss some of the specifics on how to get your
files and programs ready for use with an emulator. I will be using V9T9

as an example but most procedures have their converse in PC99. I will also briefly discuss some of the legal concerns some people have concerning emulator use. If you have any questions or comments, feel free to drop me an E-mail at d.scriver@ukonline.co.uk
or alternatively you can write to:

David A. Scriver
Rectory Lodge
Gilcrux, Cumbria CA5 2QN

RAMBLES By Stephen Shaw

sent to
ti99@infoserv.demon.co.uk
and to Ian Pare
on
10th February 1998

RAMBLES.
Stephen Shaw.
Written December 1997-
February 1998.

Hi.
This is to confirm that my web page is now up and running, and my book on the TI can be found at:
http://www.btinternet.com/~shawweb/stephen/book.htm ✲
My web site is designed to be used by the disabled (no picture map menus)
and by any browser (no frames, cookies or java).
If anyone has any TI related text on PC disk (preferably in TEXT format) - especially older material - I'd be happy to consider adding it to my site either on a permanent or revolving basis. At the time of writing the site had clocked over 450 hits.

Happy to add a page advertising this group or any group info if required. (what information would be appropriate?).

I have already posted the group details in the TI usenet group and e-mailed them to an enquirer who responded to my web page.

My book is also still available on retail sale! from amazon.com - a quite amazing bookstore which it has taken me a few months to visit, but VERY impressed - just visit www.amazon.com and take a look

✲ [2023 update: http://shawweb.myzen.co.uk/stephen/book.htm]

Charles Good (of the Lima Ohio TI user group) has been in touch and I have added a link to a web page he runs on his home village, to my own site. Nothing on the TI in his site, but his village is a remarkable find compared to the America we so often see on television.

I have heard from Gary Marshall, who was a group member a while back - his TI gear is at his mums and still in occaisional use. He is now working with computers professionally - it is always interesting to see how the
toy computer has encouraged so many to move on to use computers in an advanced manner. Here is what Gary says...

"I am currently working as a sub-contracter to Eastern Electricity using Oracle and Delphi 3 for developing client/server systems. I am one of two senior developers in a team of 22, and have responsiblity for defining standards, mentoring other developers and developing the custom database connectivity software we needed to cope with running client server applications efficiently over Eastern's WAN and VIA dial-up.
(Internet technology will be introduced during this year using Domino server).

I am exteremely lucky as my job allows me to do one of the few things I am good at and interest me (Apart from my wife Michelle and my lovely daughter Laura).

My fascination with computing started firmly with the 99/4A and I am fairly sure things would have worked out slightly different if my parents had purchased a Vic 20 which was the other option at the time.

My 99/4a currently lives with my Mum in Scotland, though I do boot it up most visits just for old times sake. I did download the v9t9 emulator after mailing you the other night. I find it bizarre (pleasantly so) to run Parsec, TI-Invaders and Munchman on my IBM Thinkpad."

===

Whilst TI is this year celebrating its 20th Birthday, in Manchester this June we celebrate the 50th Anniversary of the first program running on the first computer capable of running an electronically stored program.
To celebrate the computer has been rebuilt by Manchester University, using authentic period parts whenever possible and is to be displayed at Manchester Science Museum. It used a memory of 32 x 32 bit words and an easily remembered instruction set of seven instructions- including HALT.
Check out the web site http://www.cs.man.ac.uk/prog98 and look at their DOS PC emulator (it isn't a very big program).

===

The Science Museum in London built a Babbage analytical engine some years back- quite a few years after the first design! but could not afford the printer for it. I saw a report that a Microsoft employee was paying for a printer for them- in exchange for which they would build another engine and printer for him. The ultimate retro-computer.

===

Hectic this month at work as we switch to a new OS (MS NT 4) and new software (Office 97) - it keeps the brain alive anyway. We needed new PCs to run it (200Mhz MMX Pentium) which came with W95 installed, and then had to be wiped out... the supplier had a licence which said W95 had to be installed on every PC they sold, regardless.... (anti competitive???).
Of course moving from 33 MHz to 200MHz there is no perceptual increase in speed thanks to larger slower programming...!

===

I received a desperate e mail from the University of Helsinki - does anyone know of a TI 900 disk controller going spare? Here is what I received:

From mylikosk@alpha.hut.fi Tue Jan 13 02:25:41 1998

Hello!
My name is Marko and I am in big trouble.
We lost in University's laboratory TI900's floppy controller.
(TI900 is used to control furnaces and gas systems and program is in EPROM so we have to use it in future if possible.)

Floppy controller's whole width is 11.00in. and main connector is 6.355in.

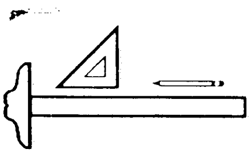
If you can not tell, please could you tell somebody who knows about that.
Yours
Marko Yli-Koski
Helsinki University of Technology
Finland

Can anyone help Marko?

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

Stephen Shaw. Stockport.

Trigonometric Functions in C99



By Francesco L. Lama

TRIGONOMET-
RIC FUNC-
TIONS IN C99

For someone who is used to writing in Extended Basic or FORTRAN, C99 presents a few frustrating aspects. I am not talking about the neccessity of declaring any variables you intend to use in your programme (after all you need to do this in FORTRAN too, and, as far as I know in all compilable languages), nor am I talking about the neccessity of loading a number of other object files (at least the CSUP library) besides the one the compiler creates from your C99 coded text file, when attmpting to run your programme.

I am talking about the unavailability of standard mathematical functions, which are certainly present in any other high level language for the TI99/4A I have ever used.

This makes translating even very simple programmes from another language into C99 extremely tedious.

I have decided to do away with this problem by producing a Functions Library which works through the Floating Point Library supplied by Tom Bentley. As a result of the rather long-winded nature of the expressions written with the Floating Point Library, these functions are rather long, and I recommend that only the ones which are going to be used should be attached as subroutines to the calling programme.

*b2?4Weee1☺ ■☺ ?αnon▸n⌂Lo?≡ ç☺ ·▸⌂αç■☺α▾o⌂L? n · ↑ ├■♥■n

I will now give a brief description of the sin(x,si) programme that follows, so that its main features are made reasonably clear, and a better programmer than me may upgrade this rather long-winded version of it.

sin(x,si) is the only subroutine used to compute the value of SIN(x); this value is returned in the variable si, which, as all C99 variables not defined as GLOBAL, is a LOCAL variable to the function sin(x,si) of type FLOAT (floating point). The reason why the two varibles, x and si, must appear before the curly braket which defines the beginning of the subroutine is that, if sin(x,si) is invoked by the calling programme as sin(a,b), then the values of a and b at the time of calling are assigned to x and si respectively.

After the curly braket, all the remaining varibles used in the subroutine sin(x,si) are defined: one integer, one pointer (used by the Floating Point routines), one string (16 characters long), and 10 Floating Point variables. Of these c0, c1, c2, c3, and c4 are used as coefficients of the polynomial used to approximate SIN(X) for -pi/2 < X < pi/2, and pi2 and pid2 are 2*pi and pi/2 respectively.

In the next seven sections of programme the above variables are assigned their numerical values. This is rather laborious since C99 has neither a DATA statement (like the ones used in FORTRAN and BASIC to assign values to constants), nor the equivalent of the variable assignment statemnt (such as c0=2.345) for floating point varibles, which could also be found in most other compilers. This implies that every time this subroutine is invoked values of all the constants have to be reassigned (the DATA statement in FORTRAN assigns the values during compilation, not during execution).

Since floating point constants can only be assigned their values through either a string variable or one or more integers (using the Floating Point Library functions stof and itof respectively), the choice was made, for the sake of greater speed, to assign the desired values to s[16], and then transfer them one by one to the relevant floating point constants.

In the section of program entitled "REDUCTION OF ARGUMENT" the value of the argument x is reduced to the equivalent angle between -2*pi and 2*pi.

The subsequent section normalizes x, that is it divides it by pi/2.

The next part tests whether the argument x is less than 0 and if so adds 4 to transform it into the equivalent positive normalized angle.

The following series of nested if statements is aimed at further modifying the argument x so that it is finally the equivalent normalized angle in the range $-1 < x < +1$ (i.e. the equivalent angle in the range $-pi/2 < X < pi/2$). This is possible because all the possible values of sine are obtained within this range.

After calculating x^2 and putting the result in r0 the actual evaluation of the formula starts by the following polynomial:

SIN(X) = ((((c4*X^2 + c3)*X^2 + c2)*X^2 + c1)*X^2 + c0)*X.

Finally the "return si" statement takes care of returning the value of SIN(X) to the calling program in the variable si. This ensures that the value of si be passed back to the corresponding variable in the function call. Note that all angles must be supplied in radians and not in degrees:

(ANGLE IN RADIANS) = (PI/180)*(ANGLE IN DEGREES).

The programme listing follows.

```
 /* This program calculates SIN(X) to the maximum precision available
*/

                         /* INSTRUCTIONS */
 sin(x,si)        /* x is in radians and is the argument of sine and */
 float x[8],si[8]; /* si returns the value of sine to the calling funct. */
 {
  int i;
```

```
char *c,s[16];
float r[8],r0[8],r1[8];
float c0[8],c1[8],c2[8],c3[8],c4[8],pi2[8],pid2[8];

s[0]='+';       /* These next seven segments of program are */
s[1]='1';       /* used to assign values to the constants. */
s[2]='.';
s[3]='5';
s[4]='7';
s[5]='0';
s[6]='7';
s[7]='9';
s[8]='6';
s[9]='3';
s[10]='2';
s[11]='7';
s[12]='';
s[13]='';
s[14]='';
s[15]='';
c=stof(s,c0);

s[0]='-';
s[1]='6';
s[2]='.';
s[3]='4';
s[4]='5';
s[5]='9';
s[6]='6';
s[7]='4';
s[8]='0';
s[9]='9';
s[10]='6';
s[11]='E';
```

```
s[12]='-';
s[13]='1';
s[14]='';
s[15]='';
c=stof(s,c1);

s[0]='+';
s[1]='7';
s[2]='.';
s[3]='9';
s[4]='6';
s[5]='9';
s[6]='2';
s[7]='5';
s[8]='8';
s[9]='2';
s[10]='7';
s[11]='E';
s[12]='-';
s[13]='2';
s[14]='';
s[15]='';
c=stof(s,c2);

s[0]='-';
s[1]='4';
s[2]='.';
s[3]='6';
s[4]='8';
s[5]='1';
s[6]='2';
s[7]='6';
s[8]='6';
```

```
s[9]='3';
s[10]='7';
s[11]='E';
s[12]='-';
s[13]='3';
s[14]='';
s[15]='';
c=stof(s,c3);

s[0]='+';
s[1]='1';
s[2]='.';
s[3]='5';
s[4]='8';
s[5]='2';
s[6]='0';
s[7]='6';
s[8]='5';
s[9]='2';
s[10]='4';
s[11]='E';
s[12]='-';
s[13]='4';
s[14]='';
s[15]='';
c=stof(s,c4);

s[0]='+';
s[1]='6';
s[2]='.';
s[3]='2';
s[4]='8';
s[5]='3';
s[6]='1';
```

```
        s[7]='8';
        s[8]='5';
        s[9]='3';
        s[10]='0';
        s[11]='7';
        s[12]='';
        s[13]='';
        s[14]='';
        s[15]='';
        c=stof(s,pi2);

        s[0]='+';
        s[1]='6';
        s[2]='.';
        s[3]='3';
        s[4]='6';
        s[5]='6';
        s[6]='1';
        s[7]='9';
        s[8]='7';
        s[9]='7';
        s[10]='2';
        s[11]='4';
        s[12]='E';
        s[13]='-';
        s[14]='1';
        s[15]='';
        c=stof(s,pid2);

                /* REDUCTION OF ARGUMENT */
c=fexp(x,"/",pi2,r);    /* x is divided by 2pi and the result put in r */
i=ftoi(r);              /* the integer part of the above is put into i */
c=itof(i,si);           /* i is made into a fp number called si */
```

```
   c=fexp(si,"*",pi2,si);   /* si x 2pi is put into si */
   c=fexp(x,"-",si,x);      /* this is subtracted from x and the result */
                       /* is assigned to x */

   c=fexp(x,"*",pid2,x);   /* NORMALIZATION OF ARGUMENT
(division by */
                      /* pi over 2) and the result is put in x */

   i=0;                 /* prepare test for argument <0 */
   c=itof(i,si);          /* the fp number si is set = 0 */
   i=4;
   c=itof(i,r1);          /* the fp r1 is set = 4 */
   if(fcom(x,"<",si))     /* if the argument is <0 */
   c=fexp(x,"+",r1,x);    /* add four to it */

   i=1;                 /* prepare for other test (argument >1) */
   c=itof(i,si);        /* set si=1 */
   if(fcom(x,">",si))
     {                  /* if true check */
     i=3;
     c=itof(i,r1);
     if(fcom(x,"<=",r1))  /* is x<=3 ? */
       {
       i=2;
       c=itof(i,si);
       c=fexp(si,"-",x,x);  /* if so x=2-x */
       }

     if(fcom(x,">",r1))      /* is x>3 ? */
       {
       i=4;
       c=itof(i,si);
       c=fexp(x,"-",si,x);  /* if so x=x-4 */
       }
```

```
    }                    /* end of main if statement */

   c=fexp(x,"*",x,r0);   /* calculate the argument squared and put in r0
*/

                      /* BEGIN EVALUATING FORMULA */
                      /* SIN(X)=((((c4xr0+c3)xr0+c2)xr0+c1)xr0+c0)xX */
   c=fexp(c4,"*",r0,r);  /* multiply c4 by x^2 */
   c=fexp(r,"+",c3,r1);  /* add c3 to the result */
   c=fexp(r1,"*",r0,r);  /* multiply result by argument squared */
   c=fexp(r,"+",c2,r1);  /* add c2 to the result */
   c=fexp(r1,"*",r0,r);  /* multiply the result by argument squared */
   c=fexp(r,"+",c1,r1);  /* add c1 to the result */
   c=fexp(r1,"*",r0,r);  /* multiply result by argument squared */
   c=fexp(r,"+",c0,r1);  /* add c0 to the result */
   c=fexp(r1,"*",x,si);  /* multiply the result by the argument */

   return si;          /* returns the sine to the calling program */
  }
                        /* END OF COMPUTATION */
```

Having worked through all this lot you may feel that tackling COS(X) and TAN(X) would be tedious in the extreme. Fortunately this is not the case, thanks to the following trigonometric identities:

COS(X) = SIN(PI/2-X), and TAN(X) = SIN(X)/COS(X).

Therefore the following two short programmes can be used as subroutines in conjunction with sin(x.si) to produce the complete set of direct trigonometric functions (since COT(X) = 1/TAN(X), SEC(X) = 1/COS(X), etc...).

```
 /* This program calculates COS(X) to the maximum precision
available */

                        /* INSTRUCTIONS */
                  /* In order to compute the cosine the function */
                  /* sin(x,y) must be included in the code by INSERT */
                  /* during editing. cos(x,co) uses sin(x,y) by setting */
                  /* COS(X)=SIN(PIover2-X). To obtain the value of
COS */
                  /* just invoke cos(a,b), where a is the argument of */
  cos(x,co)          /* the cosine and b returns its value to the calling */
  float x[8],co[8];  /* program */
  {
   char *c,s1[12];

   s1[0]='+';       /* the value of PIover2 is put into s1 */
   s1[1]='1';
   s1[2]='.';
   s1[3]='5';
   s1[4]='7';
   s1[5]='0';
   s1[6]='7';
   s1[7]='9';
   s1[8]='6';
   s1[9]='3';
   s1[10]='2';
   s1[11]='7';
   c=stof(s1,co);          /* s1 is then transformed into the fp co */

   c=fexp(co,"-",x,x);          /* the equivalent argument for sin is
assigned */
                     /* to x */
   sin(x,co);
  return co;
  }
                    /* END OF COSINE SUBROUTINE */


 /* This program calculates TAN(X) to the maximum precision avail-
able */

                        /* INSTRUCTIONS */
 tan(x,ta)          /* in order to obtain the value of TAN(X) simply */
 float x[8],ta[8];    /* invoke tan(a,b), in which a is the argument of */
 {                 /* TAN and b is the variable to which the value of */
  char *c,s1[12];       /* TAN(X) is returned. remember that sin(x,y)
must */
  float xc[8];       /* also be part of the program for tan(a,b) to work */

  s1[0]='+';         /* s1 = PIover2 */
  s1[1]='1';
  s1[2]='.';
  s1[3]='5';
  s1[4]='7';
  s1[5]='0';
  s1[6]='7';
  s1[7]='9';
  s1[8]='6';
  s1[9]='3';
  s1[10]='2';
  s1[11]='7';
  c=stof(s1,ta);      /* give the value PIover2 to fp ta */

  c=fexp(ta,"-",x,xc); /* xc=PIover2 - x */
  sin(x,ta);          /* invoke sin(x..) first */
  sin(xc,x);           /* then sin(xc..), which is equivalent to cos(x..) */

  c=fexp(ta,"/",x,ta); /* the ratio of ta and x is sin()overcos() ie tan()
*/

  return ta;
 }
```

NOTE: it is important to remember that, when running a programme containing these subroutines, two INCLUDE statements should be found at the beginning of the calling programme, in order to incorporate the Standard Input Output library via STDIO (this one may not be absolutely necessary), and the Floating Point library via FLOATI (this one is). Let me remind you also that, when running your Object file from Editor Assembler Option 3, you need to load at least two other object files: FLOAT;O, and CSUP (if you have not yet compiled and assembled FLOAT;C, you need to do so and produce FLOAT;O).

I hope this effort of mine will be of some use to somebody in the TI community, and I will be back in the next issue with the inverse trigonometric functions. Good luck.

I MUST THANK DAVE SCRIVER FOR THE FOLLOWING COMMENT HE MADE IN AN E-MAIL HE POSTED TO THE TI LIST SERVER. HE'S TRIED TO GENERATE SOME ENTHUSIASM IN THE UNITED STATES FOR OUR TI-TREF IN OCTOBER.

HERE'S A COPY OF HIS E-MAIL....

Hi all,

I just wanted to drop a line and share some recent experiences I have had here in the UK. I am an American living in Northern England and the last thing that I expected to find was an active TI User Group here. But, much to my surprise I was dreadfully wrong. The members of the UK TI Community are knowledgeable, friendly and eager to promote their hobby/passion. In fact, the co-editor of the TI Times Newsletter made a rather long trek here to Cumbria from Mansfield (Richard Twyning) to loan me a full TI system so as to facilitate the conversion of my old TI software over to an emulator.
In addition other members such as Mark Wills, Trevor Stevens, Francesco Lama and Stephen Shaw have all lent their help in achieving this goal and my heartfelt thanks go to each and everyone of them.

Now then, as I am sure you have all been made aware, their is a TI Faire called the TI-Tref scheduled for this October. As an American, I would like to encourage all that can make the trip to please attend!
If you have never been abroad, or have never been to England, you would really be missing out. As for the TI end of things, the group here is vibrant, helpful and a model for the way User Groups should be organized.

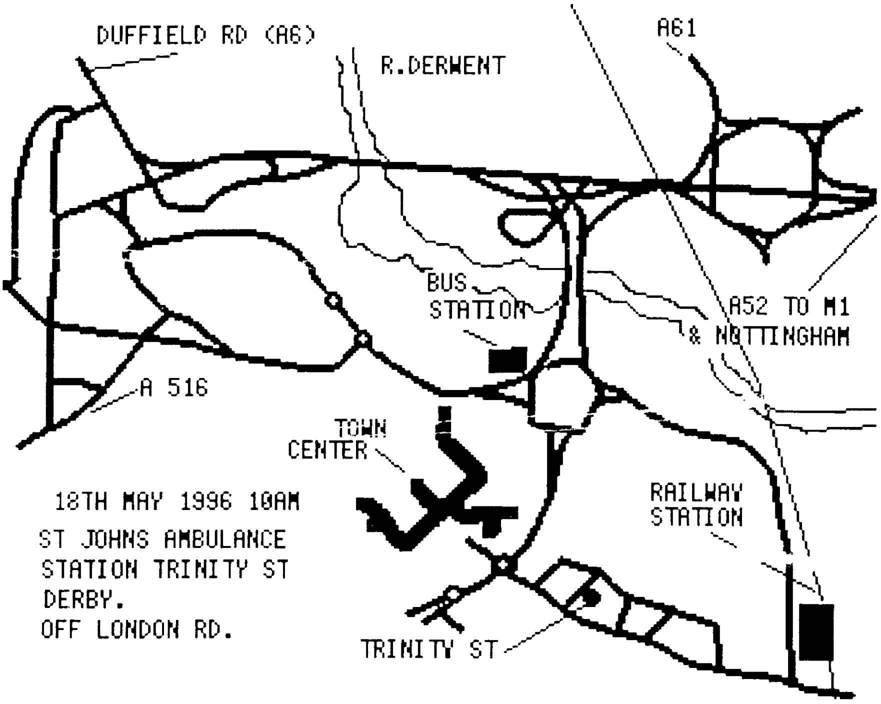
Considering the hurdles they must endure, such as lack of hardware availablilty, expensive phone calls for product support etc, their enthusiasm has not waned.
There is also a distinctive lack of negativity or superiority complexes amoungst its members who go out of their way to make new members welcome. So having said all of this I think it would be a genuine gesture of solidarity if some of you better off TI enthusiasts who would otherwise travel long distances for shows such as FW98 to come to England. It really isn't as far as you think when you consider that some folks traveled from Cleveland Ohio to Lubbock Texas. Cheap economy seats are available across the Atlantic and if booked well in enough in advance, I'm sure it could be within some of your budgets. So come on, don't let ME be the only Yank to attend!

I am still compiling, considering and re-writing my bio and I figure at my current rate of progress I should be able to submit it by oh ... this time next year.
Hopefully by that time you all will have lost interest in the idea and I will be left off the hook!

Thanks to Charles Good for the REALLY interesting PSR material. For me, the chance to peruse this stuff is a real treat and priviledge. Keep up the good work.

Type at you all later,
Dave Scriver
d.scriver@ukonline.co.uk

TI-99/4A User Group UK
Annual General Meeting



Saturday 30th May
at the
St. John's Ambulance Hall
Trinity Street, Derby

Information on V9T9 from its author Ed Swartz

> My friend Mark Wills is writing a TI Demo program in assembly >lanuage, and he was wondering if there was a memory location in V9T9 >that he could test to auto-detect whether the program is running on a 4A >or on V9T9 so he can change the message in the program accordingly.

Sure! But it's dependent upon whether various emulations are turned on,
since they modify the ROM.

For example, if the keyboard slowdown patch is on, address >0484 will be >0D40.

>02B2 =	>0C80 for keyboard emulation patch
>0962 =	>0CA0 for sprite motion patch
>065E =	>0CC0 for memory block move patch
>094C =	>0D00 for interrupt handling patch
>05AC =	>0D60 for screen filling patch

These values represent unused opcodes on the 9900, but will only function as patches at the given addresses. The user always has the option of specifying no patches, so none of these are foolproof.

Mark could try timing instructions too... I'm sure that the divide instruction, for example, is faster on V9t9 than on the 9900. Also, comparing the relative access times for the >8300->83FF range compared against the >A000->FFFF range could show which platform he's on. (On the TI, the >8300->83FF range is supposed to be much faster than the high memory, whereas on V9t9, I think it might actually be a tad slower.)
Hope this helps... unfortunately, the only other tests I can think of at the