TISHUG

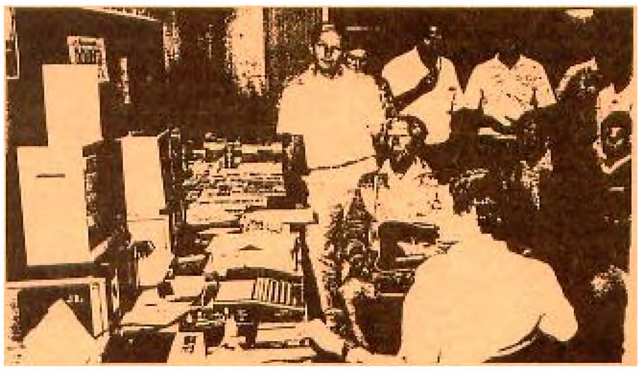
NEWS DIGEST

Focusing on the TI-99/4A Home Computer

Volume 7, Number 6 **July 1988**

Registered by Australia Post - Publication No. NBH5933

Scene at the Faire



P.O. Box 214, Redfern, New South Wales, Australia, 2016 $2

TIsHUG (Australia) Ltd.

TIsHUG News Digest

July 1988

All correspondence to:

P.O. Box 214
Redfern, NSW 2016
Australia

Membership and Subscriptions

Joining fee	$5.00
Annual Family Dues	$25.00
Overseas Airmail Dues	AUS$50.00
	or £22.00
	or US$30.00
Publications Library	$5.00
Texpac BBS	$5.00

TIsHUG Sydney Meeting

The next meeting will be at 2 pm on 2nd July at Woodstock Community Centre, Church Street, Burwood.

Printed by
The University of Wollongong
Printery

TIsHUG News Digest ISSN 0819-1984

Index

They're off

by Geoff Trott

From all accounts, both the tutorial day last month and the TI-Faire in Brisbane were great successes. Unfortunately I was unable to attend either, but I did go to the first Australian Forth Symposium instead. More of that later, but first I should apologise for the errors in last month's TND. Ross Mudie's Link-it #16 should have had a continue on page 17 at the bottom of the page. It appeared at the bottom of page 9 instead, where it was clearly not needed. Hope you worked all that out for yourself. I was sticking all those things in around midnight one night, and I guess the 9 and 6 look similar at that time of night.

****

By the way, I will be off on the family's annual trek to Adelaide for two weeks starting on 2nd July. This means that I will miss the next meeting in Sydney and the chances of the TND arriving in time for the August meeting are rather slim at this time, unless we decide to cut down on the content. I thought you might like to know before the event for a change. This week I have been struggling with a rather nasty virus, all the time keeping my eyes skinned for any sign of the snake which shed its skin in the downstairs area where my computer is set up. Since the skin is about 60 cm long, I would not like to upset the original inhabitant! So far so good.

continued on page 29

Report on the TI-Faire

by Terry Phillips

Friday afternoon and a group of interstate travellers head north from Hornsby, destination Brisbane and the awaiting TI-Faire. In the party are Les Andrews, Shane Ferrett, Peter Schubert, Ben Von Takach, Russell and Warren Welham and myself.

After an uneventfull overnight drive our group arrived in Brisbane at 6.30am, had a quick breakfast, then headed to the Faire venue where we we met by Col and Garry Christensen, a father and son combination, who are Treasurer and President, respectively of the Brisbane Group.

After setting up, people began arriving and before too long there were numbers milling around each of the exhibits, prize of these being a fully configured and operating NTSC version of the Geneve 9640. Unfortunately the guest of honour, Lou Phillips of Myarc, could not attend owing to personal business in the USA.

A run down of who we met and what was on display follows:

FROM BRISBANE - the previously mentioned Christensens plus a host of other Brisbane members. Apart from the Geneve, the Brisbane boys had taken the bull by the horns and imported some software and hardware for sale, which to me at least, was very reasonably priced. Included was the RAVE 99 speech adaptor card, essential for those contemplating buying a Geneve and wanting speech. Russell bought one of these so ask him about it if you want further information. Among the software items were TI-BASE, a new and powerful data base from Inscebot, the price being $20 which included 2 disks, a well written manual and a keyboard overlay, TI-ARTIST V2.1 for $16, ARTIST EXTRAS for $5 and DISPLAY MAKER for $5. I noticed that this software sold out fairly quickly and at those prices I do not wonder. Have a look in MICROpendium for the US equivalent prices. By the way, Garry informed me that if you are contemplating buying a Geneve, the full PAL versions should be available shortly and can be ordered through him for $750 complete. Garry can be contacted on (07)2841841 if you want further details.

Garry also presented me with V2.0 of his Disk-Aid utility, which is an assembly version and updated on the previous Forth version.

FROM THE HUNTER VALLEY 99ERS - Brian Woods, Paul Mulvaney, Peter Smith and Albert Anderson made the trip north. I think there was someone else also, but I did not write his name down. New software which was available from the HV group included Funnelweb V4.1, Cheque Book and Credit Card Manager, Genealogy Record Keeper, copies of which were obtained.

FROM ADELAIDE - Richard Earl represented the Adelaide users and presented copies of a new graphics programming language called GEE. This looks a very attractive package.

FROM MELBOURNE - Peter Gleed made the long trip to Brisbane to represent Melbourne. Peter's offering was a disk containing a program to select Lotto numbers, and a program called Flag, which I cannot get to run. Will have to contact Peter about that one.

On another table there was used hardware, software and books for sale and I think most would have been sold as the table looked bare towards the end of the day.

A couple of TIsHUG members were also noticed among the visitors. Alf Culloden, who was in Brisbane visiting Expo, and Don Gould. Alf, by the way, says if you plan on visiting Expo, take a good pair of comfortable walking shoes.

On display and for sale from our group was Peter Schubert's range of hardware, back issues of TNDs, which sold fairly well, MICROpendiums, which did not sell all that well as I guess everyone has their own arrangements for obtaining them, and a software disk containing most of the winning entries in the recently concluded software competition. One of these disks was presented to each of the visiting groups.

All too soon, it seemed it was time to pack up and face the long trip back to Sydney. But before departing a very good night of feasting and fun was had by some of the Brisbane team, the HV 99ers and our group at the Redcliffe Hotel restaurant.

Well I guess that is all there is to say about the Faire, except of course it was great to meet fellow 99ers from around the country. Would I go again? I think yes, but certainly I would explore other means of getting there.

Secretary's Notebook

by Terry Phillips

Elsewhere in this issue I have compiled a report on the Brisbane TI-Faire held during May. Following on from this two matters have been discussed which may impact on members in the future. These are:

1. The possibility of organising a joint meeting with the HV 99ers in Newcastle. It is envisaged that there would be a computer meeting during the morning and a tour of Hunter Valley vineyards during the afternoon. A social evening would round of the days activities. Watch for further news on this as arrangements get further down the track.
2. The possibility of this group and the HV 99ers organising a 1989 TI-Faire. While this thought is still very much in its infancy, members will be kept up to date as more information becomes available.

Forthcoming meeting topics

JULY - This will be a follow-up to the successful tutorial day held in June. Armed with all the knowledge you acquired on this day, and having gone home and attempted to put it into practice, you no doubt have come up with questions you would now like an answer to. Jot them down and come along and ask the experts at this meeting. Start time 2pm.

AUGUST - The big swap, buying and selling market day. Bring along your unwanted items and hopefully sell them on this day. All goods sold on an as-is basis, with the group taking no responsibility for non-working items. Meeting start 2pm.

SEPTEMBER - On this meeting day, it is hoped to be able to demonstrate the very latest in software items currently advertised in the pages of MICROpendium. Not only demonstrate, but also be able to offer to members the software for sale at realistic prices. Items likely to be available include TI-BASE, an excellent data base utility and TI-ARTIST V2.01, the renowned graphics and drawing utility. Hopefully there will also be other software packages available. If any member has any ideas on what they would like to be able to see and purchase let me know quickly, please. Again a normal 2pm start.

A big welcome is extended to 3 new members who have joined the group in recent weeks. They are:

Mark Richardson	- Blacktown
Mark Luebker	- Randwick
Keith Degraaus	- A Chief Petty Officer on HMAS Adelaide.

The latter 2 joined the group at the June meeting and seemed to be thoroughly enjoying that day's activities. Hope to see you all at future meetings.

It is also great to welcome as a member Jane Laflamme from the Ottawa TI Users Group. Jane has joined our group, as are a lot of her fellow members joining other groups throughout the world, so that more information and exchange of ideas will be possible.

A number of exchange news letters have been received over the past month. See Warren for full details on what is available.

Trouble with your clock card?

by Terry Phillips

Owing to various other commitments, members who are having construction problems with their clockcards, may not have got the assistance they needed at the June meeting. If you are in this category then all is not lost, Russell Welham will be available at the July meeting to help you out with problems you have experienced. Be sure to see Russell and explain to him the problem you are having. I feel certain he will be able to assist you.

Letters to the Editor

Dear Sir,

At around about this time of the year I am faced with the decision on which (of many) organisations I should rejoin for a further 12 months. With so many competing for my time and money it is becoming increasingly difficult to decide which ones are still worthwhile. Why should I rejoin TIsHUG? Living in the Illawarra region I am out of the mainstream of the TIsHUG community in Sydney. Perhaps there are many others in a similar situation. Receiving a monthly newsletter is not in itself worth the asking fee to rejoin. I cannot always make it to the monthly meetings. Software releases are very slow. It even costs extra to join the publications library and the bulletin board. (What happens to the money collected for these services?) I feel that clubs should be run for the benefit of the members and not to put money away for a rainy day?

This is my 5th year with the TI99/4A. I bought it as a bargain computer, as I am sure many others did. It has been a very enjoyable experience, but no one can deny that it is outshone by some of the newer computers released during the last three years. No doubt many current TI99/4A owners will change over to one of these sooner or later. Yet I believe that the TI99/4A deserves to live for a few more years yet, but how long is anyone's guess.

If TIsHUG is losing members it should do its best to find out why. I, for one, object to having to pay $25 to join then a further $10 dollars to join the publications library and the bulletin board. Surely these services can be provided free of charge now that the initial setting up costs have been paid.

Although the directors have done a good job of organising the club and its activities, they must not stop there. With each passing year there will be more and more competition for peoples time, money and interests. TIsHUG must be prepared to offer better services than previously in order to hang on to members. They must offer their members a better alternative than the "opposition"

Some of my suggestions include:

1) In order to minimise costs to users there should be no club profits on public domain software distribution, unless such profits are channeled into the aquisition of new software.
2) The latest commercial software should be purchased and evaluated for potential purchase by members. Reviews could appear in the TND. For example, new modules advertised in Micropendium.
3) Examples of the latest in hardware/firmware from the USA (or where ever) should be purchased and evaluated to determine their value.
4) All of the published literature about or related to the TI99/4A should be aquired and be available for borrowing by members AT NO EXTRA COST.
5) Bulk buying of items. For example TI joysticks, which tend to wear out but are still better than anyone elses (survey required?).
6) Hardware development costs which will directly benefit members should be bourne by the club. Certain items (for example 32K static RAM chips) should be purchased by the club in order to get bulk purchase rates.

Some of the above suggestions may seem to be contradictory in terms of my initial complaint of the cost of rejoining TIsHUG, but, as with most things, it is the perceived value that attracts people. Offer the right package and results will follow.

On a more positive note, many thanks to all those members who could and did give of themselves to provide the services that we curently enjoy. In particular, Geoff Trott and Rolf Schreiber for an excellent TND, Peter Schubert and John Paine for expanding and maintaining our computer and Ross Mudie whose unending improvements to the BBS are appreciated by all who use it (I may even become a BBS member one day).

Despite all of the above, I WILL rejoin for another year, not because of what TIsHUG has to offer, but because of a sense of loyalty to the TI99/4A. How long will this loyalty last?

Lou Amadio, Illawarra Regional Group. o

TIsHUG Software Column by Terry Phillips

The software copying at the June full day meeting proved tremendously popular. I have no idea on the number of disks copied but estimate it at upwards of 200. Some members waited in line for up to 45 minutes before their turn arrived, and maybe to be fair to all, in future a limit of 5 to 10 disks per person may have to be placed on this activity. Thanks also go to Ross for the loan of the disk duplicator which greatly speeded up the copying process.

At the July meeting, software to be distributed through the shop, will all be of local production, either from within this group or other Australian groups. Titles will be:

DISK A193 - Extended Display Package from Craig Sheehan. If you watched Craig put this through its paces at the June meeting then you will know that it is a powerful and versatile programmers utility. If you did not see it, then get a copy and you will be impressed. The package will be distributed on a flippy disk so you can use it on any disk type configuration. You will need Extended BASIC and 32K memory expansion, plus a printer for producing the lengthy documentation files.

DISK A197 - The Diskette Caretaker from Tony Imbruglia. This utility is designed to produce commented disk sleeves which can be trimmed and made into a disk jacket. It is very nicely programmed and a good addition to any software library. Requires Extended BASIC and 32K memory expansion with a printer essential. On the same disk I will add a copy of Tony's Procalc program which is a hexadecimal to decimal converter plus a lot more.

DISK A207 - Funnelweb V4.1, the very latest update of this most used utility. I will have copies in double sided and a flippy version for those with single sided drives. Extended BASIC and 32K memory expansion required.

DISK A210 - Disk-Aid, the latest assembly version of this popular utility. Extended BASIC and 32K memory expansion required.

DISK A212 - GEE, a graphics language which looks like a great piece of software to play around with. This disk also includes the Life game. Requires Extended BASIC and 32K memory expansion.

I have received, courtesy of Jane Laflamme, the latest copy of DM1000, now raised to version 4.0. Included on the disk is a version for the Geneve.

That is about it for this month, however remember, that if you require a specific program from the library either on disk or tape, or would like a hard copy or disk copy of the complete library catalog, then let me know. o

TIsHUG Shop with Bob

This column is being written before the June meeting Tutorial day. I hope you all had a worth while day. Perhaps you may have stripped the shop bare. My thanks to Steve Carr and Cyril and the other shop assistants for pulling us through this awkward time of mine. My boasts of a better column this month were empty ones. However, the HOPE listing may suffice:- Jason Scott is after a spare console. D.Dorrington wants a Mini Memory Module. (So do I. Mine was borrowed back in 1986 and has not found its way home yet.) And, Ron Kemp - I have the Dec 86 disk with a file called Archiver on it for you. Where would you like me to send it? o

For Sale

MULTIFUNCTION CARD FOR PE BOX

Trade up now to the most advanced Disk Controller for the TI with the AT Disk system and RS232 system all on one card. Adds many enhancements to your TI. Price is $350, or only $250 with your old RS232 card trade-in. Contact as above, or see Club Shop. o

The Communicators

Special Interest Group for Users of the TEXPAC BBS.
by Ross Mudie, 7th June 1988.

1. TELCO TERMINAL PROGRAM.

I recently received a phone call from Charles Earl in Canada who is the author of the TELCO Terminal Emulator. He is asking for details of the 1200/75 used in Australia and for any other comments or critisms which may help him to further improve TELCO. If any members wish to comment on the program then please leave mail to SYSOP on the BBS.

2. SENDING MAIL ON THE BBS.

Due to problems that users have been having sending TI Writer files via the BBS, small changes have been made to both the BBS and the SENDMAIL program.

The problem centered on the BBS exiting the mail or file input routine when a file was being transferred which had been created with TI Writer or one of the look alike programs. It was found that text+CR+spaces+CR in a single line could cause a false exit from the BBS editor which usually exits on a CR in the first character position of a new line. The first CR, ASCII 13, in a line followed by unprintable characters or spaces then another CR caused the BBS to exit and store the current line, then get back in time to get the second CR in the line followed by yet another CR from the sender's RS232 card under control of the SENDMAIL program. The BBS then prompted:

[C]ontinue or just ENTER >

and the SENDMAIL program saw the ">" prompt and sent the next line. If a line was sent which commenced with a "C" then the BBS editor was re-entered but any intervening lines were just lost. If no such line occurred then the rest of the file was completely lost.

To overcome these problems the SENDMAIL program has been raised to issue 5, that is, SENDMAIL5, which is available on the program download area. This program now deletes the first CR in the line and the rest of the current line. The BBS editor exit prompt has been modified to replace the ">" with a "-", that is,

[C]ontinue or just ENTER -

This will have the effect of "freezing" sendmail programs and terminal programs which detect the "line ready" prompt, ">", before transmitting the next line.

If anyone has further problems with this area of the BBS, then please let me know.

3. BBS FAILURES, the Murphy's Law Department!

There have been two system failures, the first for several months, (and just when I thought things were going well!)

On 1/6/88 at approximately 1:34 am. Failed after a download, reason unknown.

On 1/6/88, suspected to be at approximately 9.45 pm when a wide spread power glitch caused the failure of several other computer systems at the BBS site.

If the BBS fails, the watchdog timer takes the modem off line after 12 minutes. This prevents the modem from answering incoming calls when the BBS is unable to function. Callers prior to the 12 minute timeout will be answered by the modem then get no response whilst after 12 minutes the modem will not answer calls.

4. USAGE AND MEMBERSHIP.

The BBS maintains a log file to show who uses the BBS and when they signed on. The following information is derived from the logging file by a simple sort program.

TEXPAC 1988	January	February	March	April	May
CALLS	393	329	328	302	250
USERS	49	61	52	59	55
Time on line	156.26	119.39	123.40	120.13	103.37

The number of members with BBS membership is at present uncertain due to late renewals, but it looks likely that the number of registered users will drop to about 60 for this year. With less users and some changes in usage patterns, the BBS is now quite easy to get on to on most evenings.

BBS membership for TIsHUG members is only $5 per year, new members are welcome.

o

From the Bulletin Board

MAIL TO : ALL
MAIL FROM : GOWFAR

Hi to all. Can anyone help me with the INFOCOM games INFIDEL, PLANETFALL and STARCROSS, please? It is not that I need hints, it is that about at the 12th move, the program always causes a lockup in my keyboard, although the other INFOCOM games do not. If anyone can supply me with an answer or just the programs I would appreciate it. Please reply here or on SCI-FI BBS, to the SYSOP. Ta!! Regards, Greg.

MAIL TO : ALL
MAIL FROM : REQUESTS

Hello everyone,

Could the person which left mail to REQUESTS about uploading some programs please tell me the names again, as I have misplaced them. And dont forget that ANYONE can use REQUESTS.

MAIL TO : ALL
MAIL FROM : SYSOP

Looking for disk drives? Today I purchased a DSDD half height new disk drive at Sheridan Electronics in Redfern for $95. The brand is HAL (never heard of it) and it seems to work fine on my standard TI disk controller. Sheridans also have second hand full height MPI DSDD drives for $45 but they may contain faults. I am so far very happy with the HAL drive.

Regards...Ross Mudie.

MAIL TO : ALL
MAIL FROM : GOWFAR

Maybe I should take this correcting up as a lifetime pastime! I refer to the TELCO file on the BBS, here. In this file, it states that TELCO can be used to send files to BBSs by having the user set a default character (62 for TEXPAC BBS) and you can use Macros for Instruction sets, that not only allow you to logon to a BBS without remembering the Usernumber or Password but also allow you to input a set of instructions that, once on the BBS, will get you to where you want to go, quickly. The write-up also said that text files may not be over 50 lines - Have I got news for you! Any of you who have the 4A/TALK program and do not know that all the above (plus text files of any size) have not really been bothered to find out, have you? For example, to set the character that you want to wait for, from TEXPAC BBS, you go to option FCTN[6] and set the strings in option 3 under that menu. To create a Macro with 4A/TALK, load the program, hit FCTN[4] for half duplex and then type what you will, e.g. 108 (user number)

PWORD (password)

>....remembering not to use those words in brakets. Note that hitting RETURN in half duplex only causes the cursor to go back to the start of the same line that you just typed. To overcome that, instead of hitting RETURN, hit FCTN[X] which is RETURN and LINEFEED in 4A/TALK. Oh, I forgot an important part! After you have loaded the program and before the above, hit FCTN[4], to open the capture buffer. Now, having completed all the above, simply hit FCTN[5] and save to disk - use the name of the BBS it is intended for, to help you remember them without having to look them up. When you logon, open the file with FCTN[6] and when you get to a prompt, hit FCTN[D] to send a line at a time. On some of the BREADBOARD systems such as PROPHET and BLACKBOARD, you will need to use FCTN[6] before opening the file and will have to set both XON and XOFF to the same number (any number), otherwise the TBBS BBS's rip the data right out of memory without stopping. Why the big hoohaa about TELCO? All it has (that was written in the file) and more has been available for a couple of years, now with 4A/TALK. Anyone wanting to know more or where they can get a free copy of 4A/TALK, let me know!!

Regards, Greg.

----

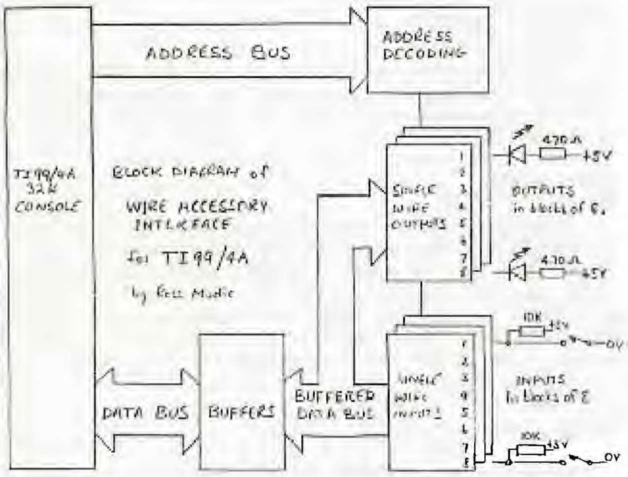
o

Wire Accessory Interface for the TI99/4A

by Ross Mudie

Have you ever thought "I would like to control the kids' train set with the computer?", or may be, "How can I connect a full music keyboard into the TI99/4A?", or "How would I control a robotic unit with the TI99/4A?". Other possible applications include: a security system; a fancy sign; or the Christmas tree lights.

Have I attracted your interest? The prototype of the Wire Accessory Interface Unit was first shown on the TIsHUG tutorial day at Burwood on 4th June 1988.



This article will show how such things are possible on a 32K memory expanded TI99/4A console. I have developed a prototype of a circuit which plugs directly into the expansion port of a 32K console which can have up to 128 single wire inputs or outputs, in blocks of 8 of either type, that is, inputs or outputs. The unit provides and accepts 5 volt logic signals on the output and input wires. Additional circuitry will be required to drive circuits requiring other conditions. My first application is for the control of a model train set, to perform the tasks of changing the points, operating the signal lamps, controlling the 12V power to the rails to stop trains and to use the information from train movement detectors to prevent train crashes when more than one train is operating on the same track. After due consultation with younger son, Peter, it was decided to go for 32 inputs and 96 outputs for the train set, used as follows:

Inputs:	29 track sections to be monitored.
	3 spare.
Outputs:	32 for the 16 points on the layout.
	29 for controlling 12 volt power to the track.
	32 for signals, 1 per track section + 3 amber.
	3 for miscellaneous building lights.

1. Construction of the demonstration prototype.

The development unit was constructed on a small piece of veroboard connected directly to a 44 way plug which plugs into the TI99/4A's expansion port. The 5 volt DC power rail from the console was used for the PCB power. Constructors should check the current drawn from the console 5V power rail and provide an external 5V power supply if the current drawn exceeds 50mA. The current drawn by the prototype was 100mA with no LEDs on and 170mA with the LEDs on.

The circuit is memory mapped from hexadecimal 8680 to 868F, which means that it uses 16 bytes. This memory area is allocated to the sound chip which is not fully decoded. This means that the sound chip will respond to values written to any memory location between hexadecimal 8400 and 87FF. Whilst this can cause some false operation of the sound chip, no other problems occur and the sound chip can be easily turned off again. It is also placed in memory immediately above John Paine's Time of Day clock.

The prototype unit was set up with 8 outputs driving 8 Light Emitting Diodes (LEDs) and 8 inputs connected from a PCB mounted switch. The outputs are at hexadecimal 8680 whilst the inputs are at hexadecimal 868C.

2. How it works.

The inputs and outputs are interfaced to the computer via 74LS373 integrated circuits which are "Tri-State octal latches". This means that they handle 8 circuits, (that is what "octal" means), and they provide a memory element for each input line, (that is the latch).

The remainder of the circuitry provides the decoding of the address bus of the computer. What this means is that when the computer addresses the appropriate memory location then the addressed input or output chip is allowed to look at the data bus and to store the value present on the data bus, if it is an output, or to place a value on the data bus if it is an input. When the extended basic statement (or command) CALL LOAD is used in the format CALL LOAD(-31104,V) the computer will place the value in the variable V on the data bus and the value hexadecimal 8680 on the address bus. After a short delay there is a pulse on the WE* line which "strobes" the output interface chip. IC25 buffers the data bus, since up to 16 74LS373 chips may need to be driven from the data bus.

This is probably a good time to look at the circuit diagram. Integrated circuits (1A), (2), (3) and (4A) decode the address bus of the computer. When the value hexadecimal 8680 (decimal -31104) is present on the address bus and the "not Write Enable" (WE*) line is active (low), then the "Latch Enable" (LE) input of IC6 receives a pulse of logic 1 and at this point IC6 stores what ever is present on the data bus. The stored value is then maintained on the output of IC6 until a new value is written into IC6 or the computer is turned off.

To perform a read from an input with Extended BASIC, a CALL PEEK is used as follows: CALL PEEK(-31092,A). This will read the 8 switches by placing a pulse of logic 0 on the control pin 1 of IC7 for the duration of the Data Bus IN (DBIN) signal whilst the address hexadecimal 868C is decoded by the PCB logic. Whilst the control input of IC18 is at logic 0, its outputs become low impedance and the conditions on the inputs are passed to the outputs which are connected to the data bus of the TI99/4A.

3. Increasing the unit from 16 to 128 lines.

The prototype unit was constructed for just 16 lines to try out the idea. Each 8 inputs require another 74LS373 chip and a NOR gate for each 8 outputs and an OR gate for each 8 inputs.

The chip requirement for the 128 line unit with 96 outputs and 32 inputs is as follows.

IC no.	Qty.	Type	Function.
1	1	74HC20	Address bus decoding logic 1's.
2	1	CD4078	Address bus decoding logic 0's.
3	1	74LS154	Address bus decoding 4 LSB's.
4	1	74HC32	Control logic.
5	1	74HC32	Final address decoding for input.
6-21	16	74LS373	Output buffers and input selector.
22-24	3	74LS02	Final address decoding for output
25	1	74LS245	Bi-directional data bus buffer
26	1	74HC04	Inverters.

The prototype unit contains ICs 1, 2, 3, 4, 5, 6, 18, 22, 24 and 25.

4. How to build your own unit.

There is a lot of work to design a printed circuit board layout for 128 inputs and outputs and the cost of a small number of double sided plated through boards may not be worthwhile. A problem with committing the circuit to a PCB layout is that the PCB does not allow outputs to be converted to inputs. There may however be interest in a smaller PCB with, say, 16 inputs and 16 outputs.

If anyone is interested in getting a PCB for this design then please let me know how many I/O's you would like. If there is enough interest steps will be taken to design a PCB. Of course the simplest way is to "roll your own" on veroboard, this way you can make a unit with just the required I/O quantity and combination.

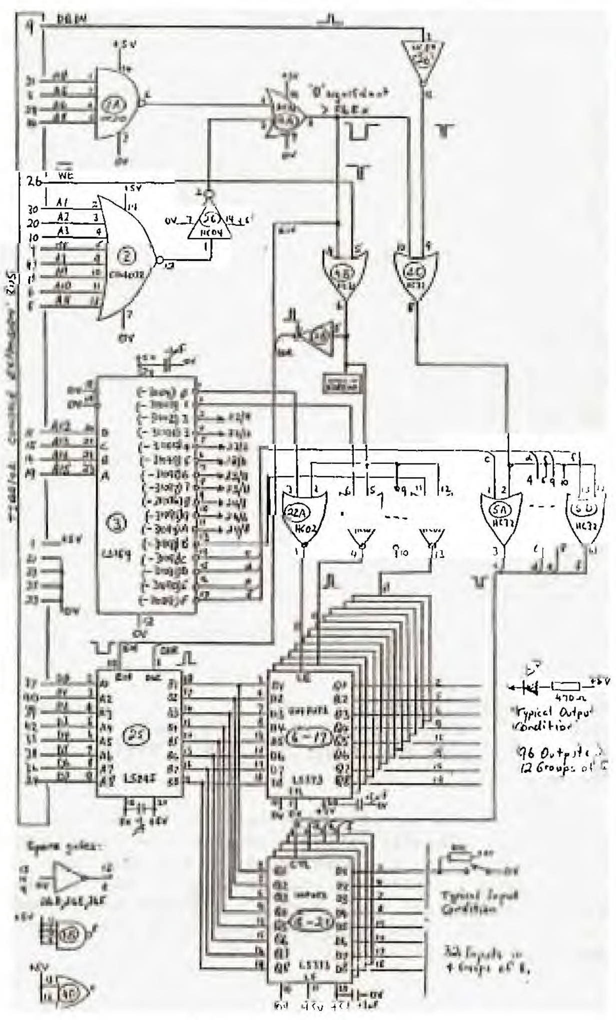
5. Assembly language programming.
The design lends itself to easy assembly language programming. Access is simple and should be quick, making processing of the inputs by an interrupt routine most probably quite feasible. As programs are developed they will be published if any interest is shown in the project.

6. Controlling a train set or other device.
The inputs and outputs are not suitable to directly control the train set. As the rest of the interface is developed copies of circuits will be available to be published if there is interest from the membership. The I/O is 5 volt logic only and is NOT protected against destruction from connecting the wrong things to it, nor is it protected against static electric discharge, apart from the normal protection provided within the integrated circuits.
The author will take no responsibility for any damage that any person may do to their computer as a result of information contained in this article.

7. Summary.
This development represents a simple way of using a "spare" TI99/4A console to perform a control task. The circuit developed so far is unsuitable for use in the expansion box since there are minor interfacing differences to operating via the side port of the console.
Additional details of hardware and software development will be made available if interest is shown in the project.

```
100 ! SAVE DSK6.CONTROL1
110 CALL INIT
120 DISPLAY AT(4,2)ERASE ALL: "DEMONSTRATION PROGRAM
    FOR": TAB(10); "PROTOTYPE": " ACCESSORY
    CONTROLLER": : TAB(8); "by Ross Mudie"
130 DISPLAY AT(10,1):"To control lamps,": "PRESS:": :
    "1.  Switches on PCB": : "2.  Binary count": : "3.
    Keys 1 to 8 control lamps"
140 DISPLAY AT(22,1):"Use BACK (Fctn 9) to return to
    this menu from any part"
150 CALL KEY(3,K,S):: IF K<49 OR K>51 THEN 150 ELSE
    K=K-48
160 ON K GOTO 190,290,410
170 !
180 !
190 ! SWITCHES TO DIRECT CONTROL LAMPS
200 DISPLAY AT(4,1)ERASE ALL:"OPERATE SWITCHES ON
    PCB.": :"Note value on screen & lamps"
210 CALL ESC
220 CALL PEEK(-31092,A)
230 DISPLAY AT(20,1):A
240 CALL LOAD(-31104,A)
250 CALL KEY(3,K,S):: IF K=15 THEN 120
260 GOTO 220
270 !
280 !
290 ! STEP THROUGH LAMPS IN BINARY SEQUENCE
300 DISPLAY AT(4,1)ERASE ALL:"Note value on screen &
    lamps"
310 CALL ESC
320 FOR V=255 TO 0 STEP -1
330 DISPLAY AT(20,1):V
340 CALL LOAD(-31104,V)
350 CALL KEY(3,K,S):: IF K=15 THEN 120
360 NEXT V
370 GOTO 120
380 !
390 !
400 ! KEYBOARD TO CONTROL LAMPS
410 CALL CLEAR :: CALL ESC
420 DISPLAY AT(4,1):"USE KEYS 1 to 8,":"note value on
    screen & lamps": :" 1  2  3  4  5  6  7  8" :: GOTO
    460
430 CALL KEY(3,K,S):: IF K=15 THEN 120 ELSE IF K<49 OR
    K>56 THEN 430
440 K=K-48
450 IF KM(K)=1 THEN KM(K)=0 ELSE KM(K)=1
460 DISPLAY AT(10,1): KM(1); KM(2); KM(3); KM(4);
    KM(5); KM(6); KM(7); KM(8)
470 V=0
480 FOR T=0 TO 7
490 IF KM(T+1)=1 THEN V=V+(2^T)
500 NEXT T
510 DISPLAY AT(14,1):V
520 CALL LOAD(-31104,255-V)
530 CALL KEY(3,K,S):: IF S<>0 THEN 530
540 GOTO 430
550 SUB ESC
560 DISPLAY AT(24,1):"Use BACK (fctn 9) to escape"
570 SUBEND
```

WIRE ACCESSORY INTERFACE
for TI99/4A, 128 I/O.
Ross Mudie 30/5/88



Address Decoding

A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
>868x	1	0	0	0	0	1	1	0	1	0	0	0	x	x	x	x

Solution to Chess Quiz

provided by George Meldrum

Key move : Rook to h6. If Black captures the Rook then the White Pawn mates. If the Black Bishop moves then the White Rook captures the pawn at h7 giving mate.

Jenny's Younger Set

Dear Jenny,
To date I have not received a single letter. It would be good if people would write in, but not to worry. Here is the address again:
Crocodile Jones
29 Palmeston Avenue
Winston Hills, NSW 2153
Please get those letters to me if you have any problems with adventures.
Here are a few clues to Mission Impossible #3.
Remember that the bomb does not go off immediately.
For the pail, the water comes in handy.
Yours faithfully
Crocodile Jones

Dear Jenny,
Is $23,000 a record at Draw Poker? I have achieved this score.
Here is a Mathematics program

```
50 REM BY V. MAKER
55 B=0
60 C=0
100 INPUT "+ - * / ? ":A$
110 IF A$="+" THEN 130
120 GOTO 160
130 INPUT "No. 1 ":B
140 INPUT "No. 2 ":C
150 PRINT B+C
155 GOTO 320
160 IF A$="*" THEN 180
170 GOTO 210
180 INPUT "No. 1 ":B
190 INPUT "No. 2 ":C
200 PRINT B*C
205 GOTO 320
210 IF A$="/" THEN 230
220 GOTO 270
230 INPUT "No. 1 ":B
240 INPUT "No. 2 ":C
250 PRINT B/C
260 GOTO 320
270 IF A$="-" THEN 290
280 GOTO 320
290 INPUT "No. 1 ":B
300 INPUT "No. 2 ":C
310 PRINT B-C
320 INPUT "ANOTHER SUM, MULTIPLICATION, DIVISION OR
    SUBTRACTION? ":B$
330 IF B$="Y" THEN 55
340 IF B$="YES" THEN 55
350 PRINT "I HOPE YOU ENJOYED THIS PROGRAM."
360 PRINT "I HOPE YOU DO WELL AT MATHS.  BYE!"
370 END
```

Well it was nice to get 2 letters this month and I must say I can readily sympathise with Crocodile Jones and his lack of mail. Never mind, we can only hope that there are still some younger members out there who are allowed to read their family's TND each month, and will burst into creativity some time if we wait long enough! o

TELCO, a New TE Program

by Chris Buttner

If you use the BBS for sending and receiving messages, you will find a program recently circulating here is a great help. It is called TELCO and was reviewed recently in MICROpendium magazine. The review version was 1.1 whereas the current version in circulation is 1.3.

TELCO is designed for use with smart modems. This does not mean it cannot be used with bread and butter modems, you just will not be able to use a number of the special features. The program allows you to emulate a number of terminals such as ANSI, VT100, ADM3A (Fasterm standard) etc. High baud rates are also supported. Because the program has so many features it is quite large. The core is 3 memory image files which call in any one of a multitude of other files as required.

Three special features which I feel worth highlighting are:
(1) the built in editor;
(2) ASCII file upload; and
(3) macro files.

The editor allows you to create your text messages from within your terminal program. Text files can be saved and reloaded for editing as often as you need. One restriction - your text file must not exceed fifty (50) lines.

The ASCII file upload is of special interest to those who send messages to the BBS. The sensible thing to do is create your message within the Editor (or have it prepared before hand. You can then send your message to the BBS from within the terminal program. In essence, there is no need to exit your terminal program to use the Sendmail program. All this is possible with the use of what is called a pace character. The terminal program sends a line of your text and then waits for the BBS to send the ">" symbol. When the program detects the ">", it then sends the next line automatically and continues in this way until all the file is sent. The pace character is set by the user as a default. For Texpac it is character 062 (ASCII).

With the macro facility you can create macro commands which can handle almost anything you do on the BBS from signing on to inserting a commonly used phrase in text or executing a sequence of commands to go to set menus within the BBS. The range is limited only by your imagination. o

Publications Library Report

with Warren Welham

New Tnd's arrived this month				
Sep 84	Apr 85	Mar 86	Mar 87	Oct 87
Oct 84	May 85	Apr 86	Apr 87	May 88
Nov 84	Jun 85	Jun 86	May 87	Jun 88
Mar 85	Oct 85	Jul 86	Sep 87	

New Books arrived this month

Code	Title	Author
00223	TMS9902A Asynchronous Communications Controller	T.Instrument
00224	The Best of 99'er Vol.1	E.Valley

New Arrivals of Overseas Publications

Group	Publications Name	Date
Northern New Jersey	-	Mar 88
Pittsburgh Users Group	The Pug Peripheral	Jan 88
Bluegrass 99 Computer Society Inc	Bytemonger	Mar 88
Hunter Valley 99ers user group	Hunter Valley News	Mar 88
Micropendium	-	Mar 88
Brisbane User Group	Bug Bytes	May 88
San Diego Computer Society	-	Feb 88
Club Information	-	Feb 88
Montreal	Cim 99	Jan 88
Club Information Montreal	Cim 99	Feb 88
Club Information Montreal	Cim 99	Mar 88
Melbourne TI Computer Enthusiasts	Melbourne Times	Apr 88

c

TI-Writer Special Character Mode

by Jim Peterson, Tigercub Software

A few newsletters recently have discussed the uses of the TI-Writer special character mode, which the manual did such a poor job of telling us about. Here are some of the things that can be done (on a Gemini 10X Printer).

(CTRL[U] FCTN[R] CTRL[U] SHIFT[M] CTRL[U] SHIFT[H] CTRL[U]) Set the left hand margin at 8. Using the TI-Writer TAB for left margin creates problems.

(CTRL[U] FCTN[R] CTRL[U] 4) Select the italic character set.

(CTRL[U] FCTN[R] CTRL[U] 5) Cancel italics, return to standard character set.

(CTRL[U] FCTN[R] CTRL[U] SHIFT[G]) Print in double-strike mode.

(CTRL[U] FCTN[R] CTRL[U] 7 CTRL[U] SHIFT[B] CTRL[U]) Select the international character set for Germany. Instead of SHIFT[B], use A for England, C for Denmark, D for France, E for Sweden, F for Italy and G for Spain.

(CTRL[U] FCTN[R] CTRL[U] SHIFT[B] CTRL[U] SHIFT[B] CTRL[U]) Set the print pitch for elite characters (12 per inch)

(CTRL[U] SHIFT[R] CTRL[U]) Restore pitch print to pica (10 cpi).

(CTRL[U] SHIFT[O] CTRL[U]) Set print pitch to condensed print.

(CTRL[U] SHIFT[R] CTRL[U]) Cancel condensed print.

(CTRL[U] SHIFT[N] CTRL[U]) Print in double width mode (on one line only); to cancel it before end of line, use DOUBLE (CTRL[U] SHIFT[T] CTRL[U]) cancelled.

(CTRL[U] FCTN[R] CTRL[U] SHIFT[H]) Cancel double strike.

(CTRL[U] FCTN[R] CTRL[U] SHIFT[E]) Emphasized mode.

(CTRL[U] FCTN[R] CTRL[U] SHIFT[F]) Cancel emphasized mode.

(CTRL[U] FCTN[R] CTRL[U] SHIFT[-] CTRL[U] SHIFT[A] CTRL[U]) Print characters with underline.

(CTRL[U] FCTN[R] CTRL[U] SHIFT[-] CTRL[U] SHIFT[A] CTRL[U]) Cancel the underlining.

(CTRL[U] FCTN[R] CTRL[U] SHIFT[S] CTRL[U] SHIFT[@] CTRL[U]) Print in superscript mode, which is always double strike and unidirectional.

(CTRL[U] FCTN[R] CTRL[U] SHIFT[T]) Cancels the superscript and unidirectional. Contrary to the manual, it obviously also cancels the double strike.

(CTRL[U] FCTN[R] CTRL[U] SHIFT[S] CTRL[U] SHIFT[A] CTRL[U]) Print in subscript mode, unidirectional and double strike.

(CTRL[U] FCTN[R] CTRL[U] SHIFT[T]) Cancels subscript, unidirectional and double strike.

Print italics double strike and underlined, return to pica without underlining but emphasized, then condensed, then double width in pica; cancel double width and go to elite type.

Any combination of codes can be used, including those which move the print head or move the paper (line feeds).

(CTRL[U] SHIFT[H] CTRL[U]) moves the print head back one space and can be used to overprint characters. (CTRL[U] SHIFT[M] CTRL[U]) placed before the end of the line will send the print head back to the beginning of the line and reprint it with the rest of the line.

I have even managed to code in customized download characters, such as my Tigercub emblem, but I cannot find any way to input CHR$(127).

Remember that these control characters are deleted by the printer and the line of print is shifted left by the number of spaces that they occupied. If you are inserting codes into text that has been prepared in columnar format, or right justified through the formatter, the best way is to use CTRL[0] to get the open square fixed mode cursor; position it over the character to the right of the point of insertion; FCTN[2] to insert characters; tap the space bar as many times as there are characters to be inserted (be careful not to shove the end of the line into oblivion!); FCTN[S] to backspace and then fill the blanks with control characters.

The text file of this article was printed through the TI-Writer editor and was also printed by a separate print program; both printouts were identical. The margin setting confused the TI-Writer formatter completely; when this was deleted, it printed out properly except that some of the foreign character sets were repeatedly overstruck and defaced!

(Printed here using a daisy wheel printer which means the effects are missing. ED)

Hidden Characters

by Steve Patterson, New Horizons

Hidden characters is a term that I have given for the use of characters in a filename that are unable to be seen on most disk catalogs. These types of characters can be used in both programs and files but there is a huge difference. With files, if you copy the program with an Extended BASIC file copier such as my 'COPIER' then you can use any character ever imagined. For program type HC I have only found one to this day, which is ASCII 127 or FCTN[V]. This is the only character that is hidden and has a key-in. You can use this character at the end of a filename to make it hard for others to run the program because they cannot figure out how to spell the filename because they do not notice the last hidden character. Of course you can use more than one and they do not have to be at the end. You could put one in the middle of the filename or you could let the entire filename be character 127. But having it at the end makes it a lot less noticeable.

Now how do you use ASCII 127 and possibly ASCII 1-29 in a filename for a file. This is where you will have to have some Extended BASIC programming knowledge. You have to use either my program mentioned above or one of your own which you can write from scratch. Basically how to get any character at the end of a filename is to open that file in Extended BASIC, then open a new file with a couple of Hidden Characters at the end of the filename. Then read each record of the file and save it to the new file. So you have actually copied the file itself and in the process you have added several invisible or Hidden Characters at the end of the filename. With the possibility of any character from 1 to 29, only you know the characters at the end. So, say you have a filename like: "DSK1.LIST". You can change that normal and easy to read file into the almost impossible to open file:

"DISK1.LIST"&CHR$(2)&CHR$(25).

This portion will place two ASCII characters (2 and 25) right after the 'T' in the name. It will not be seen on most catalogs but you will know what it is because you wrote the two characters down somewhere safe. One bad thing about this is that later, you cannot load this file into any program without changing it back to the normal way or by going in and changing the open statements in the program so that the Hidden Characters are in the filename. So you would not want to place Hidden Characters in a filename that is updated every day. Only on certain files that you only want to see and that will not be updated often, would this be of any use.

The reason I keep saying that most cataloguers will not be able to read the end characters is because I wrote one that can detect when there is a character in the filename that you would not normally see. For those who are not clear on the type of program that will add these Hidden Characters to a filename, read on.

```
100 OPEN #1:"DSK1.LIST", INPUT, DISPLAY, VARIABLE 80
110 OPEN #2:"DSK1.LIST"&CHR$(4)& CHR$(16)&CHR$(23),
    OUTPUT, DISPLAY, VARIABLE 80
120 IF EOF(1) THEN 160
130 LINPUT #1:A$
140 PRINT #2:A$
150 GOTO 120
160 CLOSE #1
170 CLOSE #2
180 END
```

The program above will give you a copy of the file LIST with three Hidden Characters at the end of the file. These characters being ASCII 4, 16 and 23. Hope you find this new and experimental process of locking up files of some use and of some help in keeping away the unwanted and un-needed.

Tips from the Tigercub #47

by Jim Peterson

Over 120 original programs in BASIC and Extended BASIC, available on cassette or disk, now reduced to just $1 each!, plus $1.50 per order for cassette or disk and postage. Minimum order of $10. Cassette programs will not be available after my present stock of blanks is exhausted. The Handy Dandy series, and Colour Programming Tutor, are no longer available on cassette. Descriptive catalogs, while they last, $1 which is deductable from your first order.

Tigercub Full Disk Collections, reduced to $5 postpaid. Each of these contains either 5 or 6 of my regular catalog programs, and the remaining disk space has been filled with some of the best public domain programs of the same category. I am NOT selling public domain programs; they are a free bonus!

TIGERCUB'S BEST, PROGRAMMING TUTOR, PROGRAMMER'S UTILITIES, BRAIN GAMES, BRAIN TEASERS, BRAIN BUSTERS!, MANEUVERING GAMES, ACTION REFLEX AND CONCENTRATION, TWO-PLAYER GAMES, KID'S GAMES, MORE GAMES, WORD GAMES, ELEMENTARY MATH, MIDDLE/HIGH SCHOOL MATH, VOCABULARY AND READING, MUSICAL EDUCATION, KALEIDOSCOPES AND DISPLAYS

NUTS & BOLTS disks

These are full disks of 100 or more utility subprograms in MERGE format, which you can merge into your own programs and use, almost like having another hundred CALLs available in Extended BASIC. Each is accompanied by printed documentation giving an example of the use of each. NUTS & BOLTS (No. 1) has 100 subprograms, a tutorial on using them, and 5 pages of documentation. NUTS & BOLTS (No. 2) has 108 subprograms, 10 pages of documentation. NUTS & BOLTS (No. 3) has 140 subprograms and 11 pp. of documentation. Now just $15 each, post paid.

TIPS FROM THE TIGERCUB

These are full disks which contain the programs and routines from the Tips from the Tigercub newsletters, in ready-to-run program format, plus text files of tips and instructions.

Tips (Vol. 1) contains 50 original programs and files from Tips newsletters #1 through #14. Tips (Vol. 2) contains over 60 programs and files from #15 to #24. Tips (Vol. 3) has another 62 from #25 to #32. Tips (Vol. 4) has 48 more from issues #33 through #41. Now just $10 each, post paid.

```
*****************************************************
* Now ready: TIPS FROM TIGERCUB (Vol. 5). Another *
* 49 programs and files from issues #42 through to *
* #50. Also $10 post paid.                         *
*****************************************************
```

TIGERCUB CARE DISKS Nos.1, 2, 3 and 4.

Full disks of text files (printer required). No. 1 contains the Tips news letters #42 thru #45, etc. Nos. 2 and 3 have articles mostly on Extended BASIC programming. No. 4 contains Tips newsletters #46 to #52. These were prepared for user group newsletter editors but are available to anyone else for $5 each postpaid.

If you bought my C11 disk, Kid's Games, please check line 100 of the Butterfly and Flowers program and, if necessary, change it to

```
100 CALL CLEAR :: CALL SCREEN(4).
```

If you bought my C12 disk, More Games, and have trouble loading Lost Plane and Andromedan Invasion, please go to line 1000 of the LOAD program and change *TC-18* to *TC-18 and *TC-23* to *TC-23. Or, return the disks to me and I will fix them.

Thanks to Ollie Hebert for this fix to the Gordian Knot in Tips #36. This will keep it from running off the edge and crashing in the automatic mode.

```
270 GOSUB 480 :: R=R-24*(R<1)+24*(R>24) ::
    C=C-28*(C<3)+28*(C>30) :: CH=128-(D=1)-(D=3) ::
    CALL GCHAR(R,C,G) :: IF G<>32 THEN IF
    INT(2*RND+1)<>1 THEN CH=G
```

The trouble with me is that, before I finish one program, I have thought of another that I want to try writing and so I do not take time time to test completed programs as well as I should. The Decompactor in Tips #35 was one that should have been tested more thoroughly. I think this version will work. It will break an Extended BASIC program into single statement lines to make it easier to modify. Then, John Dow's Compactor or a similar program will put it back together.

```
100 !DECOMPACTER V.1.1 by Jim Peterson fixed 12/87
110 DISPLAY AT(3,1) ERASE ALL: "TIGERCUB DECOMPACTER
    V.1.1": : " Program must first be -": :
    "RESequenced to greater in-": "crements than the
    number"
120 DISPLAY AT(9,1): "of statements in any one":
    "line.": : "SAVEd by": " SAVE DSK(filename), MERGE"
130 DISPLAY AT(16,1): "INPUT FILENAME?": "DSK" ::
    ACCEPT AT(17,4): IF$
140 DISPLAY AT(16,1) ERASE ALL: "OUTPUT FILENAME?":
    "DSK" :: ACCEPT AT(17,4): OF$
150 OPEN #1: "DSK"&IF$, INPUT, VARIABLE 163 :: OPEN #2:
    "DSK"&OF$, OUTPUT, VARIABLE 163
160 LINPUT #1:M$ :: LN=256*ASC(SEG$(M$,1,1))+
    ASC(SEG$(M$,2,1)) :: IF LN>LN2 THEN 180
170 DISPLAY AT(12,1) ERASE ALL BEEP: "ERROR! RESEQUENCE
    PROGRAM TO": "GREATER INCREMENTS AND TRY": "AGAIN."
    :: CLOSE #1 :: CLOSE #2 :: STOP
180 LN2=LN
190 P=POS(M$,CHR$(130),3):: IF P=0 THEN PRINT #2:M$ ::
    GOTO 260
200 A$=SEG$(M$,1,P-1) :: R=POS(A$, CHR$(132), 3) ::
    S=POS(A$, CHR$(201), 3)
210 IF R=0 THEN PRINT #2:A$&CHR$(0) :: GOTO 250
220 IF S=0 AND R<>0 THEN PRINT #2: M$ :: GOTO 260
230 IF S<>0 THEN IF S-R<3 THEN PRINT #2: A$&CHR$(0) ::
    GOTO 250
240 PRINT #2:M$ :: GOTO 260
250 LN=LN+1 :: LN2=LN :: GOSUB 270 :: M$=LN$&SEG$(M$,
    P+1, 255) :: GOTO 190
260 IF EOF(1)<>1 THEN 160 ELSE CLOSE #1 :: CLOSE #2 ::
    DISPLAY AT(12,1) ERASE ALL: "Enter NEW": : "Then
    Enter": " MERGE DSK"&OF$ :: END
270 LN$=CHR$(INT(LN/256))&CHR$(LN-256* INT(LN/256)) ::
    RETURN
```

If you have my BXB routine from Tips #40 (corrected in Tips #42) or from my TIPS disk Vol. 4 or NUTS & BOLTS (No. 3), or Genial Traveller Vol. 1 No. 6, here is a neat improvement that Barry Traver thought of. Key this in, run it to create a merge file on a disk. Then clear memory with NEW, merge in BXB, then MERGE DSK1.LINEZERO, and now save BXB again in merge format and it will CALL itself from line zero (and do something else that I am not going to tell you about!

```
100 OPEN #1: "DSK1.LINEZERO", VARIABLE 163, OUTPUT
110 M$=CHR$(0)&CHR$(0)&CHR$(157)& CHR$(200)&CHR$(3)&
    "BXB"&CHR$(130)& CHR$(157)&CHR$(200)&CHR$(4)&
    "CHAR"& CHR$(183)&CHR$(200)&CHR$(2)&"30"
120 M$=M$&CHR$(179)&CHR$(199)&CHR$(16)&
    "81C37EA58199663C"&CHR$(182)&CHR$(0) :: PRINT #1:M$
    :: PRINT #1: CHR$(255)&CHR$(255)
```

And if you have merged in BXB, the edge character (ASCII 31) can be re-identified and coloured (set 0) to give the screen an ornamental border.

```
100 CALL CHAR(31,"0"):: CALL CLEAR :: FOR J=1 TO 24 ::
    PRINT :: NEXT J :: CALL CHAR(31,
    "1824429999422418") :: CALL COLOR(0,5,16)
```

Here is an improved version of the CATWRITER program to create the Tigercub QUICKLOADER, which is intended for disks of programs which you have filled and do not plan to change. It will read the directory, display each filename, and ask you for the complete program name of each one. Then it prepares a program which displays one or more menu screens of complete program names, and auto-loads whichever one you select.

First, key in this part and save it to disk by SAVE DSK1.CAT1,MERGE. If you want, you can change the screen and character colours in line 10. Do not change the line numbers!

```
10 CALL CLEAR :: DIM M$(127) :: CALL SCREEN(5):: FOR
   S=0 TO 14 :: CALL COLOR(S,16,1) :: NEXT S :: CALL
   PEEK(8198,A) :: IF A<>170 THEN CALL INIT
11 REM (leave this in!)
12 ON WARNING NEXT :: GOSUB 21
13 X=X+1 :: READ M$(X):: IF M$(X)<>"END" THEN 13
```

```
14 R=3 :: FOR J=1 TO X-1 :: READ X$ :: DISPLAY AT(R,1):
   STR$(J);TAB(4);X$ :: R=R+1 :: IF R<23 THEN 17
15 DISPLAY AT(24,1):"Choice? or 0 to continue 0" ::
   ACCEPT AT(24,26) VALIDATE(DIGIT) SIZE(-3): N :: IF
   N>X-1 THEN 15
16 IF N<>0 THEN 19 :: R=3
17 NEXT J
18 DISPLAY AT(24,1): "Choice?" :: ACCEPT AT(24,9)
   VALIDATE(DIGIT): N :: IF N=0 OR N>X-1 THEN 18
19 CALL CHARSET :: CALL CLEAR :: CALL SCREEN(8) :: CALL
   PEEK(-31952,A,B) :: CALL PEEK(256*A+B-65534,A,B) ::
   C=256*A+B-65534 :: A$="DSK1."&M$(N) :: CALL
   LOAD(C,LEN(A$))
20 FOR J=1TO LEN(A$) :: CALL LOAD(C+J,
   ASC(SEG$(A$,J,1))) :: NEXT J :: CALL LOAD(C+J, 0)::
   GOTO 10000
21 CALL LOAD(8196,63,248)
22 CALL LOAD(16376,67,85,82,83,79,82,48,8)
23 CALL LOAD(12288,129,195,126,165,129,153,102,60)
24 CALL LOAD(12296,2,0,3,240,2,1,48,0,2,2,0,8,4,32,
   32,36,4,91)
25 CALL LINK("CURSOR") :: RETURN
10000 RUN "DSK1.1234567890"
```

Next, key in this little routine and run it to create a file called CAT2.

```
100 OPEN #1: "DSK1.CAT1", VARIABLE 163, INPUT
110 OPEN #2: "DSK1.CAT2", VARIABLE 163, OUTPUT
120 FOR J=10 TO 26:: LINPUT #1: M$ :: PRINT #2:
   CHR$(0)&CHR$(J)&CHR$(156)& CHR$(253)&CHR$(200)&
   CHR$(1)&"2"& CHR$(181)&CHR$(199)&CHR$(LEN(M$))&M$
   &CHR$(0) :: NEXT J
130 PRINT #2:CHR$(255)&CHR$(255):: CLOSE #1:: CLOSE #2
```

Finally, key in CATMATRIX. Leave the line numbers as they are, we need that space after line 9.

Then MERGE in DSK1.CAT2 to combine the two, and SAVE.

```
1 CALL CLEAR :: CALL TITLE(16, "CATWRITER") :: CALL
   CHAR(124, "3C4299A1A199423C") :: DISPLAY AT(2,10):
   "Version 1.3":;: TAB(8); "& Tigercub Software"
2 DISPLAY AT(15,1):"For free": "distribution": "but no
   price or": "copying fee": "to be charged." :: FOR
   D=1 TO 500 :: NEXT D :: CALL DELSPRITE(ALL)
3 DISPLAY AT(2,3) ERASE ALL: " TIGERCUB CATWRITER
   V.1.3":;: " Will read a disk directory,": "request
   an actual program": "name for each program-type"
4 DISPLAY AT(7,1): "filename, and create a merg-":
   "able Quickloader which dis-": "plays full program
   names and": "runs a selected program."
5 DISPLAY AT(12,1): " Place disk to be cataloged": "in
   drive 1 and press any key" :: CALL KEY(0,K,S) :: IF
   S=0 THEN 5
9 OPEN #2: "DSK1.CATMERGE", VARIABLE 163,OUTPUT
100 OPEN #1: "DSK1.", INPUT, RELATIVE, INTERNAL ::
   INPUT #1: N$,A,J,K :: LN=1000 :: FN=1 100
110 DISPLAY AT(12,1): "Disk name?":;: N$ ::
   ACCEPT AT(14,1) SIZE(-28): N$ ::
   LX$=STR$(14-LEN(N$)/2) :: LXLEN=LEN(LX$)
120 PR$=CHR$(0)&CHR$(11)&CHR$(162)& CHR$(240)
   &CHR$(183)&CHR$(200)&CHR$(1)&
   "1"&CHR$(179)&CHR$(200)& CHR$(LXLEN)&LX$
130 PR$=PR$&CHR$(182)&CHR$(181)& CHR$(199)&
   CHR$(LEN(N$))& N$&CHR$(0) :: PRINT #2: PR$
140 X=X+1 :: INPUT #1: P$,A,J,B :: IF LEN(P$)=0 THEN
   180 :: IF ABS(A)=5 OR ABS(A)=4 AND B=254 THEN 150
   ELSE X=X-1 :: GOTO 140
150 DISPLAY AT(12,1): P$;"PROGRAM NAME?" ::
   ACCEPT AT(14,1) SIZE(25):F$
160 PRINT #2: CHR$(INT(FN/256))&CHR$(FN-256*
   INT(FN/256))& CHR$(147)&CHR$(200)&CHR$(LEN(F$))&F$&
   CHR$(0) :: FN=FN+1
170 M$=M$&CHR$(200)&CHR$(LEN(P$))&P$& CHR$(179) :: IF
   X<11 THEN 140
180 IF M$="" THEN 200
190 PRINT #2: CHR$(INT(LN/256))&CHR$(LN-256*
   INT(LN/256))& CHR$(147)&SEG$(M$,1,LEN(M$)-1)&
   CHR$(0) :: LN=LN+1 :: M$="" :: X=0 :: IF LEN(P$)<>0
   THEN 140
200 PRINT #2: CHR$(INT(LN/256))&C[illegible] LN-256*
   INT(LN/256))& CHR$(147)&CHR$([illegible])&CHR$(3)& "END"&
   CHR$(0)
210 PRINT #2:CHR$(255)&CHR$(255):: CLOSE #1:: CLOSE #2
220 DISPLAY AT(8,1) ERASE ALL: "Enter -":;: "  NEW":;:
   "  MERGE DSK1.CATMERGE":;:
   "  DELETE ""DSK1.CATMERGE""":;: "  SAVE DSK1.LOAD"
230 SUB TITLE(S,T$)
240 CALL SCREEN(S):: L=LEN(T$):: CALL MAGNIFY(2)
250 FOR J=1 TO L :: CALL SPRITE(#J, ASC(SEG$(T$,J,1)),
   J+1-(J+1=S)+(J+1=S+13)+13*(J>14), J*(170/L),
   10+J*(200/L)) :: NEXT J
260 SUBEND
```

Mike Stanfill and Ed Machonis and others have been publishing some neat little single-screen "tinygram" programs, so here is my contribution. It is a one-screen one-liner!

```
1 RANDOMIZE :: PRINT : : : : : :: A=INT(RND*7) ::
   B=INT(RND*9+1) :: FOR X=1 TO 5 :: Y=A*X*X-B*X+B ::
   PRINT Y; :: NEXT X :: Y=A*X*X-B*X+B :: PRINT : : ::
   INPUT "GUESS NEXT NUMBER": N :: IF N=Y THEN PRINT :
   "RIGHT" :: GOTO 1 ELSE PRINT : "CORRECT IS": Y ::
   GOTO 1
```

Memory full! - Jim Peterson

To GOSUB or Not to GOSUB, That is the Question

by Ben Takach

The GOSUB routine is a placid, forgiving sort of an affair in BASIC. It will take a fair amount of punishment without complaint. One can call it from almost anywhere and it never forgets where to return. This last virtue of this useful tool is the downfall of the foolhardy. Its patience has defined limits. The secret of its unerring homing instinct is literally buried in the stack. The stack is just like that 9" (230 mm) long spike fastened to a heavy base plate, which Mrs. T. uses to anchor her unpaid bills to. When her stack is full, yours truly blows his stack. That is precisely what happens to your trusted TI99/4A when its stack is full. Texas Instruments, like most other computer makers, is very secretive about the stack capacity, altough one can dig it out from some of the more hi-tech litterature. Let us say it is adequate for the purpose.

Each time the program reaches a GOSUB instruction, an unexecuted return address is spiked onto the stack. The subroutines may also send the program to other subroutines, so each time a new return address is spiked on the top of the previous address. One can see, that by this simple housekeeping method (no doubt TI has copied it from Mrs. T.), the computer can not get lost, since only the topmost address is accessible, which belongs to the last GOSUB instruction. Hopefully, by the time the program is near to its end, the stack is almost empty again. It may happen that the program finishes in the middle of a subroutine, and one or more unexecuted return addresses remain in the stack. Well you will never hear about it, the junk will be discarded during the spring cleaning operation to prepare the stage for the new program. On the other hand, if one makes a habit of sending the program to a new location with a GOTO instruction from within a subroutine, then each time a return address will remain on the stack as it will not be needed (the unpaid bills syndrome). Eventually this could cause the stack to overflow. At this time you will get an error message and the program will crash. Extended BASIC is more helpful, it will tell you STACK OVERFLOW, however BASIC will let you guess it with the all encompasing MEMORY FULL error message. The rule is, if you GOSUB then RETURN, if you intend to go elswhere from a program segment then use GOTO instead. To put it in a more precise form, do not use GOTO instructions inside subroutines. Parentheses may cause a similar problem. The computer will "peel off" the parentheses layers to reach the very core of the expression. The stack will be overflowing if the expression is too complex. Again one can not find any reference to the safe limit. If the parentheses problem raises its ugly head, break the complex expression in half and let the computer execute the calculation in two stages. Extended BASIC is most helpful by reporting which line did break the camel's back.

The third possible cause of a stack overflow may be due to a user defined function. This may become very volatile if a complex user defined function refers to another user defined function. The remedy like before: simplify all or part of the defined function by converting it to a (numeric or string) variable.

Subroutines and Sub-programs

from SUBFILE99, USA

Word Wrap and Fill Routines

Below is a short program that illustrates two very useful routines for handling displays of string data. The first routine automatically right justifies each line of text to give a neat appearance to things like instructions etc., without having to "count out the spaces."

The second routine will automatically calculate the longest possible portion of the string that can be presented on one line and breaks the line up accordingly. This prevents that annoying break up of words that sometimes occurs when printing long strings.

The actual routines themselves appear first in both BASIC and Extended BASIC. A short TI BASIC demo program follows.

Fill Subroutines

```
10298 REM *FILL/B*
10299 REM
10300 FOR XL=1 TO LEN(M$)
10301 IF LEN(M$)=28-XI THEN 10307
10302 IF SEG$(M$,XL,1)<>" " THEN 10305
10303 M$=SEG$(M$,1,XL)& " "& SEG$(M$, XL+1, LEN(M$)-XL)
10304 XL=XL+1
10305 NEXT XL
10306 GOTO 10300
10307 RETURN
10308 REM
10798 ! *FILL/X*
10799 !
10800 SUB FILL(R,I,M$)
10801 FOR X=1 TO LEN(M$)
10802 IF LEN(M$)=28-I THEN 10804 ELSE IF SEGS(M$,X,1)=
      " " THEN M$=SEG$(M$,1,X)& " "& SEG$(M$, X+1,
      LEN(M$)-X) :: X=X+1
10803 NEXT X :: GOTO 10801
10804 DISPLAY AT(R,29-LEN(M$)):M$
10805 SUBEND
```

Wrap Subroutines

```
12598 REM *WRAP/B*
12599 REM
12600 X1=0
12601 M$=M$&" "
12602 X2=POS(M$," ",X1+1)
12603 PRINT SEG$(M$,X1+1,X2-X1); 12604 IF
      X2=LEN(M$)THEN 12607
12605 X1=X2
12606 GOTO 12602
12607 RETURN
12608 REM
12598 ! *WRAP/X*
12599 !
12600 SUB WRAP(M$)
12601 M$=M$&" " :: X1=0
12602 X2=POS(M$," ",X1+1)
12603 PRINT SEG$(M$,X1+1,X2-X1); :: IF X2=LEN(M$) THEN
      SUBEXIT
12604 X1=X2 :: GOTO 12602
12605 SUBEND
```

Demonstration Program

```
100 REM ***************
110 REM *             *
120 REM * FILL & WRAP *
130 REM *             *
140 REM *  SUB DEMOS  *
150 REM *             *
160 REM ***************
170 REM
180 REM SUBFILE99
190 REM 04/85
200 REM
210 REM *HOUSEKEEPING*
220 REM
230 L$="----------------------------"
240 CALL CLEAR
250 RESTORE 970
260 FOR L=1 TO 17
270 READ M$
280 PRINT TAB(7);M$
290 NEXT L
300 REM
310 INPUT " PRESS ENTER TO START":A$
320 REM
330 REM *SELECT DEMO*
340 REM
350 CALL CLEAR
360 PRINT "SELECT DEMO:"
370 PRINT "------------"
380 PRINT
390 PRINT
400 INPUT "<W>RAP, <F>ILL OR <Q>UIT? ":A$
410 IF (A$<>"W")*(A$<>"F")THEN 920
420 IF A$="W" THEN 690
430 REM
440 REM *FILL DEMO*
450 REM
460 CALL CLEAR
470 PRINT "FILL DEMO"
480 PRINT "---------"
490 PRINT
500 PRINT
510 PRINT L$
520 PRINT
530 RESTORE 1100
540 FOR L=1 TO 10
550 READ M$
560 GOSUB 1290
570 PRINT M$
580 NEXT L
590 REM
600 PRINT
610 PRINT L$
620 PRINT
630 PRINT
640 INPUT "HIT <CR> KEY":A$
650 GOTO 350
660 REM
670 REM *WRAP DEMO*
680 REM
690 CALL CLEAR
700 PRINT "WRAP DEMO"
710 PRINT "---------"
720 PRINT
730 PRINT
740 PRINT L$
750 PRINT
760 RESTORE 1240
770 FOR L=1 TO 2
780 READ M$
790 GOSUB 1400
800 NEXT L
810 REM
820 PRINT
830 PRINT
840 PRINT L$
850 PRINT
860 PRINT
870 INPUT "HIT <CR> KEY":A$
880 GOTO 350
890 REM
900 REM *QUIT PROGRAM*
910 REM
920 CALL CLEAR
930 END
940 REM
950 REM *TITLE DATA*
960 REM
970 DATA "***************"
980 DATA "*             *"
990 DATA "* FILL & WRAP *"
1000 DATA "*             *"
1010 DATA "*  SUB DEMOS  *"
1020 DATA "*             *"
1030 DATA "***************"
1040 DATA ,,," SUBFILE99"
1050 DATA " 04/85",,,,,
1060 REM
1070 REM
1080 REM *FILL DATA*
1090 REM
1100 DATA THIS IS AN EXAMPLE OF THE
1110 DATA FILL ROUTINE IN TI-BASIC.
1120 DATA AS YOU CAN SEE - IT IS NOT
1130 DATA THE FASTEST ROUTINE AROUND
1140 DATA BUT IT GETS THE JOB DONE!
1150 DATA IT COMES IN VERY HANDY WHEN
1160 DATA YOU WANT TO CREATE A NEAT
1170 DATA LOOKING SCREEN LAYOUT W/O
1180 DATA ALL THE HASSLE OF COUNTING
1190 DATA SPACES WHEN ENTERING DATA!
```

on page 26

A Review of PRbase

by Peter Smith

Recently a very extensive and genuinely "dinky di" review of the major data base programs was published in that great supporter of the TI99/4A, MICROpendium. I would suggest that anyone who has read this far should be interested enough to read page 32 of the October issue of that great magazine.

I use PRbase to keep track of the children in the primary (soon also the infants) department of my school, (approximately 197 children).

I used to use Multiplan, as this allowed me to use numerous mathematical operations on the information which I kept, however the memory limitations meant that each class had to be kept separately and EXTERNAL COPIES of only a few items from each class had to be used if a departmental printout was required (for example, a list of names, houses and ages for the whole department).

PRbase overcomes the memory problem by keeping details (records) on disk. Very large amounts of data can be kept in one record: each small piece of data is called a field; lots of fields make a record; and a collection of records make a file. For instance, you can have up to 32 fields in a record, up to 700 records per disk and each field can be up to 246 characters long. This sort of power allows, "something useful" to be done with the TI.

Using the program

There are 2 main sections to the program:
1. setting up the disk, the screen for entry of data, the output for reports and mailing lists; and
2. entering and manipulating data.

Once the difficult part (in terms of organising what you want) is completed part 2 is a breeze and a delight. A help screen is easily available from the data entry screen. The instructions are clear, in retrospect. Once the steps have been deciphered and gone through, and a fair bit of planning undertaken and executed, the program acts in a way which makes it convenient and efficient. (I have noticed that virtually any worthwhile activity needs planning and not just intuitive "hunches".)

Setting up the main data screen is a busy and thoughtful task which requires planning, but so it should. A certain amount of predefined graphic characters can be incorporated into this screen to add some "panache" and "colour" to the proceedings.

Reports require very careful attention and experimentation. Up to 5 separate reports may be configured and called up. To set them up, you need to know about your printer codes etc.

Disk access time, when writing and reading to disk is certainly not time consuming. The ease of editing screens and searching for and altering data is truly a joy.

At this stage I feel that I am warning people away from attempting to use the program. Please give it a number of tries, because when it is finally under way, you will have learnt a lot about your computer and printer, and about the data you wish to use.

One of the main uses of data bases is to present specific data sorted in special ways. For example, all those children in class 5C sorted into age groups in houses alphabetically. Unfortunately P.R.B will find this difficult, unless some of the special utilities are used as, you see, PRbase only sorts on 2 fields. (Once I have said this, someone will shoot me down. Please do, but show me how it can be done, simply, time after time.) Individual screens can be printed simply.

I guess I am never happy (completely that is) but I do have a wish list of things which I would love to be able to do with PRbase, I think that it would make it perfect for my use. (**Note, this is a very subjective point of view.)

1. Mathematical functions being added would allow a variety of uses.
2. A combined index and sort routine. Although fast, the sort routine is still rather time consuming.
3. A method of varying the size of the sort field. As it is based on 10 characters and can often involve more than 1 field in the sorting process, which can lead to errors in data output. For example, I have 2 fields side by side, HOME CLASS and READING CLASS. If I sort "HOME CLASS" on say "4D", and have a "4D" in my "READING CLASS", often the child, with "4D" in "READING CLASS" will show up in the sort. This is easily fixed by careful naming of classes, but could be eliminated if the sort field length was "definable?".
4. Global alterations. Global searches are available, but boy, it would be nice to be able to:
   (a) do global maths activities. For example, last year my childrens' ages were one year less than this year on 1/1/88. It would be great to be able to add a given quantity to a field in all records. This would allow me to change childrens' ages at the beginning of the year.
   (b) Do the equivalent of TI-Writer's "Replace String" that would allow easier correction of errors.
5. The ability to sort on more than 2 fields is often needed and would be an advantage to have. (For example, all 8 year old boys in Newman House in class 3V)

I feel I have not done this wonderful program justice with my comments about my "wish list", as even without these functions it is fantastic and I am more than happy to recommend it to anyone. I congratulate the author for such a fine job and the support he gives to users having problems, even if they live half way round the world from him and have simply had a glitch in printing out the comprehensive instructions.

I have a couple of hints which may help you.
1. When setting up the printer, I have found that PIO.EC works. I tried just about everything else.
2. Remember that all data is kept as strings and this means that if we do a sort on "ages" we can end up with some funny results. For example, 7.2, 9.6, 7.1, 10.4, 7.11 sorted from smallest to largest becomes 10.4, 7.1, 7.11, 7.2, 9.6. I have found it necessary to use 4 digits as follows; 07.01, 07.02, 07.11, 09.06, 10.04.
3. When using the Option facility to redirect the output to disk, make sure that you use spaces to fill the field when prompted for the output device. For example: DSK3.FILE1 (and spaces until cursor stops)

I would encourage you, if you have stuck with me this far, to try the program, as it is good; but, ah well, I would love an Alpha GTV 2000 too!! o

Purchasing Software Overseas

by Lou Amadio

Have you ever looked through a copy of Micropendium and marvelled at the prices of software (and hardware) in the advertisements? The prices, although listed in $US, were sufficiently low to tempt me to send away for some module software that I had not seen before. I mentioned this to some of my friends, and, before I knew it, I had enough requests for a sizeable order.

The order consisted of:

22 modules	$US130.95
5 joysticks	$US 39.75
Postage (airmail)	$US 87.00
US State Tax	7.73
	65.43
Convert to $A at 73.9 =	$A 359.17
Aust Import Duty	$A 55.54
Total	$A 414.71

The final landed prices were considerably higher than what one would expect from a simple conversion from $US to $A. In fact the ratio turned out to be approximately 2.43.

Note:

1) The above items were purchased at a time when the $A was worth approximately 74c US. Current exchange rates (81c, May 88) will help to offset the direct purchases by approximately 9.5%.

2) Surface mail rates would be considerably cheaper than airmail, but you will have to wait approximately 3 months. o

US Tennis

by Robert Brown and Stephen Judd

This tennis program features most of the actions of a real tennis match, including:
- serves;
- forehand and backhand shots;
- lobs, volleys;
- balls out or in the net;
- defensive or offensive play;
- tie-breaker; and
- scores announced by the referee via the speech synthesizer.

And even better, you can play against a wonderful partner: the TI99/4A computer; or against a second player. Three different levels allow you to select a perfect partner, really adapted to your training and skill, from beginner to professional.

Furthermore, a live demonstration game between two computer players will show you how realistic the action is, and perfectly illustrate all the capabilities of this program.

Users' instructions

1- Loading the program

Required configuration:
- TI99/4A;
- peripheral expansion system with disk drive and memory expansion (it is also possible to load it from cassette using Extended BASIC with just 32K memory expansion, ED);
- joysticks;
- Speech Synthesizer (optional); and
- Editor/Assembler cartridge (or Extended BASIC, ED).

Select the Editor/Assembler option 3 (Load and run)

-File Name: DSK1.TENNIS
-Program Name: TENNIS

The program is now ready for use.

2- Selecting game options

The introduction screen appears, announcing the program. After a few seconds, a demonstration game starts automatically, showing live action. Press BACK, then any key to get the option selection screen.

```
                 TENNIS
            OPTION SELECTION


              1        2
            PLAYER   PLAYERS    DEMO

  NOVICE     --0
  AMATEUR
  PRO
```

Select the level and the number of players (or a demonstration game) by moving the small racket shown in the chart by using the joystick or the arrow keys (S,D,E,X). Press ENTER or FIRE once your selection is made.

You are then prompted for the name of the players. You can also give a name to the computer champion. If you do not enter a name, the computer will just assign a standard one to allow the two players to be distinguished on the score board. Note that a coloured player indicates the colour of the player to which the name is assigned.

Then the following message appears at the bottom of the screen: "REMOVE THE ALPHA-LOCK THEN PRESS ENTER"

You are now ready to start your tennis match.

3- Playing a tennis match

Move the players with the joysticks. Press the FIRE button to swing the racket in order to hit the ball. You can position the player to receive the ball either on the forehand or on the backhand. When you press the FIRE button, the racket starts moving. The direction of the shot is determined by the relative position ball and racket when the coincidence is detected.

Serving

When it is your turn to serve, use your joystick to give the direction of the ball, relatively to the serve area (left, center or right) but also the strength of your serve (up or down for fast serve, center position for normal serve). Then press FIRE while keeping the joystick in the selected position. If your first serve is out, you are naturally given a second chance. the probability of success is related to the direction and strength you selected as in a real tennis game.

Positioning the player to return the ball

Moving your player to the right (left) results automatically in a positioning of his racket for a forehand shot (backhand shot). However, in order to allow a fine positioning of players, they can move a few steps left or right before the racket gets actually positioned. In any case, hitting the FIRE button results in moving the racket from backhand to forehand and vice versa.

Returning the ball

The ball speed control can be achieved by the players' motion when they hit the ball.
- If the player moves towards the net, the ball will be accelerated.
- If the player moves backward, the shot will be a lob if the opponent is close to the net.
- If the player does not move vertically, the ball will be hit at normal speed.

The ball direction is also affected by the movement of the player.

Scoring

All the tennis rules are respected. the players change sides after every odd game. The referee announces the score. The match takes place in five sets. A tie-break game takes place when necessary.

Levels

The three levels are characterized by the pace of the action and by the increasing aggressiveness of the computer champion. At novice level, the computer champion returns the ball in your direction and is not aggressive. At professional level, the champion becomes merciless; he alternates fast and normal shots, executes lobs and volleys. He will not let you breathe a second. At amateur level, the computer champion plays at an intermediate level, but be careful; sometimes he will play as a real professional. At the end of a match, the level and the number of players is displayed, showing the level of a performance.

Special options

- Pause: pressing SPACE results in stopping temporarily the action. Press any other key to resume the game.
- Speed: the keys + and - allow to increase or decrease by step the pace of the game.
- Colour: the colour of the court can be changed by pressing the function key followed by 1, 2 or 3. This simulates various kind of tennis courts (grass, clay or decoturf).
- Redo: the key sequence FCTN[8], allows a restart of a match from the beginning.
- Back: the key sequence FCTN[9] takes you back to the options selection screen.

o

RAMdisk Menu Tip

by Lou Amadio

I do not remember reading about the following in the Menu documentation, so I am presenting it for what it is worth.

Whilst using the SHOW DIRECTORY option from the primary Menu screen I discovered that it is not necessary to revert back to the primary menu in order to do a show directory on another drive. All that needs to be done is to press the number of the next drive that you want to access. For example, to show the directory of drives 2, 5, 8 simply enter these numbers in any order (after first selecting "SHOW DIRECTORY" from the primary menu).

If, on the other hand, you wish to list the directory of a number of floppy discs in drive 1, simply press 1 after replacing each disk with the next one.

continued on page 30

```
10 ! CRAYON PROGRAM--DISK
20 ! DATE 10/23/81
30 ! BY JEFF COOPER
40 CALL CLEAR
50 CALL SCREEN(3)
60 PRINT "GOING TO USE SPEEC
H?  Y";
70 ACCEPT AT(24,23)SIZE(-2)V
ALIDATE("YN")BEEP:I$
80 IF I$="Y" THEN 100
90 IF I$="N" THEN SPCH=1 ELS
E 60
100 CALL CLEAR
110 PRINT "PROGRAMS:": :"1 C
REATE A DESIGN.": :"2 DISPLA
Y OLD DESIGNS (DISK) ": :
120 INPUT "ENTER PROGRAM #":
N
130 IF N=2 THEN 1570
140 IF N<>1 THEN 120
150 CALL CLEAR
160 PRINT "COLOR CRAYON PROG
RAM.": : :
170 INPUT "WOULD YOU LIKE IN
STRUCTIONS Y OR N: ":I$
180 IF I$="N" THEN 310
190 IF I$<>"Y" THEN 170
200 PRINT : : :"THIS PROGRAM
 LETS YOU DRAW":"DESIGNS ON
THE SCREEN.": :"YOU HAVE A M
OVABLE SQUARE"
210 PRINT "WHICH CAN CHANGE
COLOR.":"YOU CAN ALSO CHANGE
 THE":"SCREEN COLOR.": : :
220 INPUT "HIT ENTER TO CONT
INUE ":I$
230 CALL CLEAR
240 PRINT "TO CHANGE THE COL
OR OF THE":"SQUARE, PRESS TH
E SHIFT KEY":"('B'*) THEN A
COLOR CODE    (1-16) THEN PR
ESS ENTER."
250 PRINT : :"TO CHANGE THE
SCREEN COLOR":"PRESS ENTER('
.'*) THEN A    COLOR CODE,TH
EN PRESS ENTER.": : :
255 PRINT "* ON 99/4A": :
260 INPUT "HIT ENTER TO CONT
INUE ":I$
270 CALL CLEAR
280 PRINT "MAKE A DESIGN BY
LEAVING A":"TRAIL: PUSH S TO
 GO <,":"D TO GO >, E ^, X V
,":"AND W R Z C FOR DIAGONAL
S."
290 PRINT : : :"TO SKIP OVER
 A LINE PUSH:":"J<, K>, I^,
MV, AND U O N .":"FOR DIAGON
ALS.": : :
300 PRINT "TO END PROGRAM PU
SH CLEAR.": :
310 PRINT : :"WILL YOU WANT
TO SAVE THE"
320 INPUT "DESIGN ON DISK Y
OR N: ":I$
330 IF I$="N" THEN 380
340 IF I$<>"Y" THEN 320
350 FLAG=1
360 PRINT : :"TO SAVE YOUR D
ESIGN PRESS P":"(PERMINATE)
AND THEN ENTER.":"THE SCREEN
 WILL BE PUT ON":"DISK."
370 OPEN #1:"DSK1.2180/D",AP
PEND,VARIABLE 254,INTERNAL
379 INPUT "HIT ENTER ":I$
380 CALL CLEAR
390 COL=15
400 PRINT "HERE ARE THE COLO
R CODES.": :"TRANSPARENT 1",
"BLACK        2","MED. GREEN
 3","LT. GREEN   4","DR. BLU
E    5",
410 PRINT "LT. BLUE    6","D
R. RED     7","CYAN        8
","MED. RED    9","LT. RED
  10","DR. YELLOW 11",
420 PRINT "LT. YELLOW 12","D
R. GREEN  13","MAGENTA    14
","GRAY        15","WHITE
  16": :
430 ROW=15
440 OCHAR=32
450 OROW=1
460 IF SPCH THEN 480
470 CALL SPGET("ENTER",ENT$)
:: CALL SPGET("SQUARE",SQR$)
:: CALL SPGET("SCREEN",SCR$)
:: CALL SPGET("COLOR",COL$)
480 OCOL=1
490 PRINT "ENTER SCREEN COLO
R CODE: 8";
500 IF SPCH THEN 520
510 CALL SAY(,ENT$,,SCR$,,CO
L$)
520 ACCEPT AT(24,26)SIZE(-3)
BEEP VALIDATE(DIGIT):SCR_COD
E
530 IF (SCR_CODE>16)OR(SCR_C
ODE=0)THEN 510
540 PRINT "ENTER SQUARE COLO
R CODE: 7";
550 IF SPCH THEN 570
560 CALL SAY(,ENT$,,SQR$,,CO
L$)
570 ACCEPT AT(24,26)SIZE(-3)
BEEP VALIDATE(DIGIT):K
580 IF (K>16)OR(K=0)THEN 560
590 SC=K*4+76
600 CALL CLEAR
610 CALL SCREEN(SCR_CODE)
620 CALL SETCOLOR
630 CALL KEY(1,K,S)
640 CALL GCHAR(ROW,COL,FLASH
)
650 CALL HCHAR(ROW,COL,30)
660 CALL HCHAR(ROW,COL,FLASH
)
670 IF S=0 THEN 940
680 IF K=16 THEN 790
690 ADDROW=(K=5)+(K=6)+(K=4)
-(K=14)-(K=0)-(K=15)
700 ADDCOL=(K=15)+(K=2)+(K=4
)-(K=6)-(K=3)-(K=14)
710 ROW=ROW+ADDROW
720 COL=COL+ADDCOL
730 ROW=(ROW>0)*(ROW<25)*ROW
-(ROW<1)-(ROW>24)*24
740 COL=(COL>0)*(COL<33)*COL
-(COL<1)-(COL>32)*32
750 CALL HCHAR(ROW,COL,SC)
760 OROW=ROW
770 OCOL=COL
780 GOTO 630
790 IF SPCH THEN 810
800 CALL SAY(,ENT$,,SQR$,,CO
L$)
810 CALL GCHAR(24,16,CGC)
820 CALL GCHAR(24,17,CG2)
830 CALL GCHAR(24,18,CG3)
840 ACCEPT AT(24,14)SIZE(3)V
ALIDATE(DIGIT):K$
850 IF K$="" THEN 840
860 K=VAL(K$)
870 IF (K>16)OR(K=0)THEN 840
880 SC=K*4+76
890 CALL HCHAR(ROW,COL,SC)
900 CALL HCHAR(24,16,CGC)
910 CALL HCHAR(24,17,CG2)
920 CALL HCHAR(24,18,CG3)
930 GOTO 630
940 CALL KEY(2,K,S)
950 IF S=0 THEN 630
960 IF K=11 THEN 1220
970 IF K=13 THEN 1100
980 ADDCOL=(K=15)+(K=2)+(K=4
)-(K=6)-(K=3)-(K=14)
990 ADDROW=(K=5)+(K=6)+(K=4)
-(K=14)-(K=0)-(K=15)
1000 ROW=ADDROW+ROW
1010 COL=COL+ADDCOL
1020 ROW=(ROW>0)*(ROW<25)*RO
W-(ROW<1)-(ROW>24)*24
1030 COL=(COL>0)*(COL<33)*CO
L-(COL<1)-(COL>32)*32
1040 CALL GCHAR(ROW,COL,NCHA
R)
1050 CALL HCHAR(OROW,OCOL,FL
ASH)
1060 FLASH=NCHAR
1070 OROW=ROW
1080 OCOL=COL
1090 GOTO 630
1100 IF SPCH THEN 1120
1110 CALL SAY(,ENT$,,SCR$,,C
OL$)
1120 CALL GCHAR(24,16,CGC)
1130 CALL GCHAR(24,17,CG2)
1140 CALL GCHAR(24,18,CG3)
1150 ACCEPT AT(24,14)SIZE(3)
VALIDATE(DIGIT):K1$
1160 IF K1$="" THEN 1150
1170 SCR_CODE=VAL(K1$)
1180 IF (SCR_CODE>16)OR(SCR_
CODE=0)THEN 1150
1190 CALL SCREEN(SCR_CODE)
1200 IF SCR_CODE<3 THEN CALL
 COLOR(0,8,1)ELSE CALL COLOR
(0,2,1)
1210 GOTO 900
1220 IF FLAG=0 THEN 630
1230 IF SPCH THEN 1250
1240 CALL SAY("TO SAVE+SCREE
N+ON+DISKETTE, PRESS ENTER")
1250 CALL KEY(0,K,S)
1260 IF (S=0)+(S=-1)THEN 125
0
1270 IF K<>13 THEN 630
1280 IF SCR_CODE>9 THEN 1310
1290 BS$="0"&STR$(SCR_CODE)
1300 GOTO 134D
1310 BS$=STR$(SCR_CODE)
1320 CC=0
1330 NLINES=9
1340 FOR R=1 TO 3
1350 B$=""
1360 IF R=3 THEN NLINES=6
1370 FOR COUNT=1 TO NLINES
1380 CC=CC+1
1390 FOR C=3 TO 30
1400 CALL GCHAR(CC,C,CGC)
1410 B$=B$&CHR$(CGC):: NEXT
C :: NEXT COUNT
1420 IF R=3 THEN 1450
1430 PRINT #1:B$
1440 NEXT R
1450 FOR C=1 TO 2
1460 GOSUB 1540 :: NEXT C
1470 FOR C=31 TO 32
1480 GOSUB 1540 :: NEXT C
1490 O$=B$&SEG$(B1$,1,84)
1500 B1$=SEG$(B1$,85,12)
1510 PRINT #1:O$
1520 PRINT #1:B1$&BS$
1530 GOTO 630
1540 FOR R=1 TO 24
1550 CALL GCHAR(R,C,CGC):: B
1$=B1$&CHR$(CGC):: NEXT R
1560 RETURN
1570 CALL CLEAR
1580 PRINT "DISPLAY DESIGNS
PROGRAM."
1590 PRINT : : :"YOUR DESIGN
S WILL BE":"RETRIEVED FROM D
ISK AND":"DISPLAYED ON THE S
CREEN.": :
1600 PRINT "IF YOU HAVE MORE
 THAN ONE":"DESIGN, HIT THE
SPACE BAR;":"YOUR NEXT DESIG
N WILL BE":"DISPLAYED."
1610 PRINT : :"END PROGRAM B
Y PUSHING CLEAR": : : :
```

continued on page 17

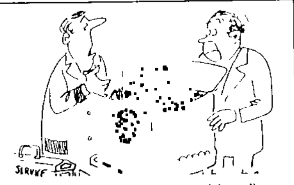
```
100 REM ************
110 REM *DRAW POKER*
120 REM ************
130 REM TISHUG LIBRARY 1&2
140 REM TI BASIC
150 CALL CLEAR
160 PRINT TAB(11);"DRAW POKE
R"
170 PRINT "AUTHOR..": :TAB(6
):"MANUEL CONSTANTINIDIS"
180 DIM UC$(13)
190 DIM RR(9)
200 UC$(1)="003E040810221C"
210 UC$(2)="001C222018221C"
220 UC$(3)="00103E12141810"
230 UC$(4)="001C22201E021E"
240 UC$(5)="001C22221E021C"
250 UC$(6)="0004040810203E"
260 UC$(7)="001C22221C221C"
270 UC$(8)="001E203C22221C"
280 UC$(9)="00324A4A4A4B32"
290 UC$(10)="001C2220202020"
300 UC$(11)="003C322A22221C"
310 UC$(12)="0022120A060A12"
320 UC$(13)="0022223E22221C"
330 US$(1)="081C3E7F7F3E1C08
"
340 US$(2)="081C3E3E7F7F7736
"
350 US$(3)="1C083E7F2A1C1C08
"
360 US$(4)="1C08367F7F3E1C08
"
370 DIM S(2,5)
380 DIM H(13)
390 DIM D$(5)
400 FOR DE=1 TO 1500
410 NEXT DE
420 CALL CLEAR
430 PRINT TAB(10);"INSTRUCTI
ONS": : :
440 PRINT "TO DISCARD A CARD
 PRESS A":"NUMBER FROM 1 TO
5.WITH THE":"TOP CARD BEING
NUMBER 1":
450 PRINT "DO THE SAME FOR A
LL CARDS TO":"BE DISCARDED T
HEN PRESS THE":"SPACE BAR.IF
 YOU CHANGE";
460 PRINT "YOUR MIND ON A CA
RD SIMPLY":"PRESS THAT NUMBE
R AGAIN.":"CARDS TO BE DISCA
RDED ARE":"INDICATED BY AN";
470 PRINT " ARROW.": : :
480 PRINT "TO CHANGE THE VAL
UE OF THE":"CHIPS DURING THE
 GAME PRESS":"0 WHEN ASKED T
O MAKE A BET."
490 PRINT "PRESSING THE SPAC
E BAR FOR":"A BET WILL TAKE
THE PREVIOUS":"BET VALUE.":
: :
500 INPUT "VALUE OF EACH CHI
P?":CVAL
510 IF CVAL<=50 THEN 540
520 PRINT "HOUSE LIMIT $50"
530 GOTO 500
540 CALL CLEAR
550 RANDOMIZE
560 INPUT "HOW MUCH TO START
 WITH?":BANK
570 IF BANK<=1000 THEN 600
580 PRINT "DON'T BE GREEDY":
"LESS THAN $1000 PLEASE"
590 GOTO 560
600 CALL CLEAR
610 PRINT "WILL YOU USE A  1
. 7 UP DECK":"
  2. FULL DECK"
620 CALL KEY(0,KEY,STATUS)
630 IF STATUS<=0 THEN 620
640 IF KEY=49 THEN 680
650 IF KEY<>50 THEN 620
660 CID=52
670 GOTO 690
680 CID=32
690 CALL CLEAR
700 DIM CRD(52)
710 IF CID=52 THEN 740 ELSE
720
720 GOSUB 4380
730 GOTO 770
740 FOR I=1 TO 52
750 CRD(I)=I
760 NEXT I
770 XP=2
780 YP=2
790 A$="BET:   CHIPS   BANK:
 $"&STR$(BANK)
800 GOSUB 3020
810 YP=4
820 XP=8
830 A$="CHIP VALUE: $"&STR$(
CVAL)
840 GOSUB 3020
850 FOR C=144 TO 156
860 READ C$
870 CALL CHAR(C,C$)
880 NEXT C
890 RESTORE
900 FOR C=128 TO 140
910 READ C$
920 CALL CHAR(C,C$)
930 NEXT C
940 READ C$
950 CALL CHAR(157,C$)
960 READ C$
970 CALL CHAR(158,C$)
980 CALL CHAR(159,"")
990 READ C$
1000 CALL CHAR(141,C$)
1010 READ C$
1020 CALL CHAR(142,C$)
1030 FOR C=120 TO 125
1040 READ C$
1050 CALL CHAR(C,C$)
1060 NEXT C
1070 READ C$
1080 CALL CHAR(96,C$)
1090 READ C$
1100 CALL CHAR(97,C$)
1110 READ C$
1120 CALL CHAR(98,C$)
1130 READ C$
1140 CALL CHAR(99,C$)
1150 CALL CHAR(35,"010204089
0A0C0F0")
1160 DATA "0038440810207C"
1170 DATA "00384418044438"
1180 DATA "00081828487C08"
1190 DATA "00784078044438"
1200 DATA "00384078444438"
1210 DATA "007C0408102020"
1220 DATA "00384438444438"
1230 DATA "003844443C0478"
1240 DATA "004CD25252524C"
1250 DATA "00040404044438"
1260 DATA "0038444544C3C"
1270 DATA "00485060504844"
1280 DATA "003844447C4444"
1290 DATA "10387CFEFE7C3810"
1300 DATA "6CEEFEFE7C7C3810"
1310 DATA "10383854FE7C1038"
1320 DATA "10387CFEFE6C1038"
1330 DATA "FF"
1340 DATA "0101010101010101"
1350 DATA "80808080808080FF"
1360 DATA "00000000000000FF"
1370 DATA "8080808080808080"
1380 DATA "FF01010101010101"
1390 DATA "0707070707070707"
1400 DATA "00000000000000FF"
1410 DATA "0101010707070707"
1420 DATA "0000000000000001"
1430 FOR Z=1 TO 8
1440 CALL COLOR(Z,2,13)
1450 NEXT Z
1460 K=16
1470 CALL COLOR(9,2,K)
1480 CALL COLOR(13,2,K)
1490 CALL COLOR(14,2,K)
1500 CALL COLOR(12,2,13)
1510 CALL COLOR(15,14,K)
1520 CALL COLOR(16,14,K)
1530 CALL COLOR(10,14,K)
1540 CALL COLOR(11,2,K)
1550 CALL SCREEN(2)
1560 REM  *******PROGRAM****
**
1570 FOR F=10 TO 23
1580 CALL VCHAR(7,F,32,12)
1590 NEXT F
1600 CALL HCHAR(23,1,32,32)
1610 CALL HCHAR(24,1,32,32)
1620 GOSUB 2930
1630 GOSUB 2180
1640 GOSUB 4010
1650 CRDNUM=1
1660 X=10
1670 Y=14
1680 GOSUB 2310
1690 S(1,CRDNUM)=SUIT
1700 S(2,CRDNUM)=CARD
1710 GOSUB 2550
1720 FOR CRDNUM=2 TO 5
1730 X=X+2
1740 Y=Y+1
1750 GOSUB 2310
1760 S(1,CRDNUM)=SUIT
1770 S(2,CRDNUM)=CARD
1780 GOSUB 2620
1790 NEXT CRDNUM
1800 CRDNUM=CRDNUM-1
1810 XP=7
1820 YP=20
1830 A$="DISCARD ANY CARDS?"
1840 GOSUB 3020
1850 CALL KEY(0,KEY,STATUS)
1860 IF STATUS<=0 THEN 1850
1870 IF (KEY=49)+(KEY=50)+(K
EY=51)+(KEY=52)+(KEY=53)+(KE
Y=32)THEN 1900
1880 CALL SOUND(1,110,1)
1890 GOTO 1850
1900 IF KEY=32 THEN 1980
1910 KEY=KEY-48
1920 IF D$(KEY)=" " THEN 195
0
1930 D$(KEY)=" "
1940 GOTO 1960
1950 D$(KEY)="#"
1960 CALL HCHAR(13+KEY,8+2*K
EY,ASC(D$(KEY)))
1970 GOTO 1850
1980 X=10
1990 Y=14
2000 CALL HCHAR(20,1,32,32)
2010 FOR II=1 TO 5
2020 IF D$(II)<>"#" THEN 209
0
2030 CRDNUM=CRDNUM+1
2040 GOSUB 2310
2050 GOSUB 2790
2060 H(S(2,II))=H(S(2,II))-1
2070 S(1,II)=SUIT
2080 CALL HCHAR(Y,X,32)
2090 X=X+2
2100 Y=Y+1
2110 NEXT II
2120 GOSUB 3070
2130 IF RANK<>0 THEN 2150
2140 GOSUB 3200
2150 GOSUB 3460
2160 CALL KEY(0,KEY,STATUS)
2170 IF STATUS<=0 THEN 2160
ELSE 1570
2180 A$="            SHUFFLING
"
2190 XP=1
2200 YP=12
2210 GOSUB 3020
2220 FOR I=1 TO CID
2230 RANDOMIZE
2240 N=INT(CID*RND)+1
2250 P=CRD(I)
```

```
2260 CRD(I)=CRD(N)
2270 CRD(N)=P
2280 NEXT I
2290 CALL HCHAR(12,1,32,32)
2300 RETURN
2310 IF (CRD(CRDNUM)>=1)*(CR
D(CRDNUM)<=13)THEN 2320 ELSE
 2370
2320 SUITCHR=157
2330 SUIT=1
2340 CARD=CRD(CRDNUM)-13*(SU
IT-1)
2350 CRDCHR=CRD(CRDNUM)+143
2360 GOTO 2530
2370 IF (CRD(CRDNUM)>=14)*(C
RD(CRDNUM)<=26)THEN 2380 ELS
E 2430
2380 SUITCHR=158
2390 SUIT=2
2400 CARD=CRD(CRDNUM)-13*(SU
IT-1)
2410 CRDCHR=CRD(CRDNUM)+130
2420 GOTO 2530
2430 IF (CRD(CRDNUM)>=27)*(C
RD(CRDNUM)<=39)THEN 2440 ELS
E 2490
2440 SUITCHR=141
2450 SUIT=3
2460 CARD=CRD(CRDNUM)-13*(SU
IT-1)
2470 CRDCHR=CRD(CRDNUM)+101
2480 GOTO 2530
2490 SUITCHR=142
2500 SUIT=4
2510 CARD=CRD(CRDNUM)-13*(SU
IT-1)
2520 CRDCHR=CRD(CRDNUM)+88
2530 H(CARD)=H(CARD)+1
2540 RETURN
2550 CALL HCHAR(Y-7,X+1,123,
4)
2560 CALL VCHAR(Y-6,X+5,124,
6)
2570 CALL VCHAR(Y-6,X,121,6)
2580 CALL HCHAR(Y,X+1,120,4)
2590 GOSUB 2750
2600 GOSUB 2790
2610 RETURN
2620 CALL HCHAR(Y-7,X+1,97,2
)
2630 CALL HCHAR(Y-7,X+3,122)
2640 CALL HCHAR(Y-7,X+4,123)
2650 CALL VCHAR(Y-6,X+5,124,
6)
2660 CALL VCHAR(Y-7,X,99)
2670 CALL VCHAR(Y-6,X,98)
2680 CALL VCHAR(Y-5,X,96,4)
2690 CALL VCHAR(Y-1,X,125)
2700 CALL HCHAR(Y,X+1,120,4)
2710 GOSUB 2750
2720 CALL HCHAR(Y-5,X+1,SUIT
CHR)
2730 GOSUB 2790
2740 RETURN
2750 FOR L=(Y-6)TO (Y-1)
2760 CALL HCHAR(L,X+1,159,4)
2770 NEXT L
2780 RETURN
2790 CALL HCHAR(Y-6,X+1,CRDC
HR)
2800 CALL HCHAR(Y-5,X+1,SUIT
CHR)
2810 IF (CRDNUM=5)+(II=5)THE
N 2820 ELSE 2920
2820 IF (SUIT=1)+(SUIT=2)THE
N 2830 ELSE 2880
2830 CALL CHAR(104,UC$(CARD)
)
2840 CALL CHAR(105,US$(SUIT)
)
2850 CALL HCHAR(Y-2,X+4,105)
2860 CALL HCHAR(Y-1,X+4,104)
2870 RETURN
2880 CALL CHAR(112,UC$(CARD)
)
2890 CALL CHAR(113,US$(SUIT)
)
2900 CALL HCHAR(Y-2,X+4,113)
2910 CALL HCHAR(Y-1,X+4,112)
2920 RETURN
2930 FOR I=1 TO 13
2940 H(I)=0
2950 NEXT I
2960 FOR I=1 TO 5
2970 S(1,I)=0
2980 S(2,I)=0
2990 D$(I)=" "
3000 NEXT I
3010 RETURN
3020 REM
3030 FOR I=1 TO LEN(A$)
3040 CALL HCHAR(YP,I+XP,ASC(
SEG$(A$,I,I)))
3050 NEXT I
3060 RETURN
3070 RANK=0
3080 IF (S(1,1)=S(1,2))*(S(1
,1)=S(1,3))*(S(1,1)=S(1,4))*
(S(1,1)=S(1,5))THEN 3090 ELS
E 3100
3090 RANK=4
3100 I=0
3110 I=I+1
3120 IF I>=10 THEN 3190
3130 IF H(I)=0 THEN 3110
3140 IF (H(I)=H(I+1))*(H(I)=
H(I+2))*(H(I)=H(I+3))*(H(I)=
H(I+4))THEN 3150 ELSE 3190
3150 IF RANK<>4 THEN 3180
3160 RANK=1
3170 RETURN
3180 RANK=5
3190 RETURN
3200 I=0
3210 I=I+1
3220 IF I=14 THEN 3380
3230 IF H(I)=0 THEN 3210
3240 ON H(I)GOTO 3210,3250,3
340,3440
3250 I=I+1
3260 IF I=14 THEN 3400
3270 IF H(I)=2 THEN 3300
3280 IF H(I)=3 THEN 3320
3290 GOTO 3250
3300 RANK=7
3310 RETURN
3320 RANK=3
3330 RETURN
3340 I=I+1
3350 IF I=14 THEN 3420
3360 IF H(I)=2 THEN 3320
3370 GOTO 3340
3380 RANK=9
3390 RETURN
3400 RANK=8
3410 RETURN
3420 RANK=6
3430 RETURN
3440 RANK=2
3450 RETURN
3460 ON RANK GOTO 3470,3510,
3550,3590,3630,3670,3710,375
0,3790
3470 A$="    A STRAIGHT FLUS
H 100/1"
3480 CALL SOUND(4000,440,2,6
59,2,880,2)
3490 BON=100
3500 GOTO 3820
3510 A$="        FOUR OF A KIN
D 30/1"
3520 CALL SOUND(3600,440,2,6
59,2,880,2)
3530 BON=30
3540 GOTO 3820
3550 A$="         FULL HOUSE
20/1"
3560 CALL SOUND(3200,440,2,6
59,2,880,2)
3570 BON=20
3580 GOTO 3820
3590 A$="          FLUSH 15/
1"
3600 CALL SOUND(2200,440,2,6
59,2,880,2)
3610 BON=15
3620 GOTO 3820
3630 A$="        STRAIGHT 10
/1"
3640 CALL SOUND(1500,440,2,6
59,2,880,2)
3650 BON=10
3660 GOTO 3820
3670 A$="     THREE OF A KIN
D 5/1"
3680 CALL SOUND(1000,440,2,6
59,2,880,2)
3690 BON=5
3700 GOTO 3820
3710 A$="         TWO PAIRS 3
/1"
3720 CALL SOUND(1000,440,2,6
59,2,880,2)
3730 BON=3
3740 GOTO 3820
3750 A$="          ONE PAIR 1
/1"
3760 CALL SOUND(500,440,2,65
9,2,880,2)
3770 BON=1
3780 GOTO 3820
3790 A$="            NOTHING"
3800 CALL SOUND(500,110,1)
3810 BON=0
3820 RR(RANK)=RR(RANK)+1
3830 XP=1
3840 YP=23
3850 GOSUB 3020
3860 WIN=BET*CVAL*BON
3870 IF BON=0 THEN 4000
3880 A$="YOU WIN $"&STR$(BET
*CVAL)&" X "&STR$(BON)&" = $
"&STR$(WIN)
3890 XP=INT((32-LEN(A$))/2)
3900 YP=24
3910 GOSUB 3020
3920 FOR P=1 TO 20
3930 CALL SOUND(50,RANK*100+
1000,2)
3940 NEXT P
3950 BANK=BANK+WIN
3960 A$=STR$(BANK)&"  "
3970 XP=24
3980 YP=2
3990 GOSUB 3020
4000 RETURN
4010 A$="BETTING? 1_5 CHIPS"
4020 XP=6
4030 YP=12
4040 GOSUB 3020
4050 CALL KEY(0,KEY,STATUS)
4060 IF STATUS<=0 THEN 4050
4070 IF KEY=32 THEN 4110
4080 IF KEY=48 THEN 4210
4090 IF ((KEY<=48)+(KEY>=54)
)THEN 4050
4100 BET=(KEY-48)
4110 BANK=BANK-(BET*CVAL)
4120 CALL HCHAR(12,1,32,32)
4130 XP=7
4140 YP=2
4150 A$=STR$(BET)
4160 GOSUB 3020
4170 XP=24
4180 A$=STR$(BANK)&"    "
4190 GOSUB 3020
4200 RETURN
4210 CALL CLEAR
4220 CALL SCREEN(13)
4230 INPUT "VALUE OF EACH CH
IP?":CVAL
4240 IF CVAL<=50 THEN 4270
4250 PRINT "HOUSE LIMIT $50"
4260 GOTO 4230
4270 CALL CLEAR
```

continued from page 14

```
4280 A$="BET:   CHIPS   BANK
: $"&STR$(BANK)
4290 XP=2
4300 YP=2
4310 GOSUB 3020
4320 A$="CHIP VALUE: $"&STR$
(CVAL)
4330 XP=8
4340 YP=4
4350 GOSUB 3020
4360 CALL SCREEN(2)
4370 GOTO 4010
4380 CC=0
4390 FOR I=6 TO 13
4400 CC=CC+1
4410 CRD(CC)=I
4420 NEXT I
4430 FOR I=19 TO 26
4440 CC=CC+1
4450 CRD(CC)=I
4460 NEXT I
4470 FOR I=32 TO 39
4480 CC=CC+1
4490 CRD(CC)=I
4500 NEXT I
4510 FOR I=45 TO 52
4520 CC=CC+1
4530 CRD(CC)=I
4540 NEXT I
4550 RETURN
```

```
1620 INPUT "HIT ENTER TO CON
TINUE. ":I$
1630 OPEN #2:"DSK1.2180/D",I
NPUT ,VARIABLE 254,INTERNAL
1640 CALL CLEAR
1650 CALL SETCOLOR
1660 COUNT=0
1670 IF EOF(2)THEN 1830
1680 INPUT #2:A$
1690 INPUT #2:B$
1700 INPUT #2:C$
1710 INPUT #2:I$
1720 D$=SEG$(C$,1,168)
1730 I$=SEG$(C$,169,84)&I$
1740 PRINT A$;B$;D$;
1750 FOR C=1 TO 2
1760 GOSUB 1870 :: NEXT C
1770 FOR C=31 TO 32
1780 GOSUB 1870 :: NEXT C
1790 CALL SCREEN(VAL(SEG$(I$
,97,2)))
1800 CALL KEY(0,K,S)
1810 IF S=0 THEN 1800
1820 IF K<>32 THEN 1800 ELSE
 1660
1830 CLOSE #2
1840 IF SPCH THEN CALL CLEAR
 :: STOP
1850 CALL SAY("END+OF+DATA,
PRESS+CLEAR TO+STOP")
1860 GOTO 1860
1870 FOR R=1 TO 24
1880 COUNT=COUNT+1
1890 CALL VCHAR(R,C,ASC(SEG$
(I$,COUNT,1))):: NEXT R
1900 RETURN
1910 SUB SETCOLOR
1920 FOR I=1 TO 15 STEP 2
1930 CALL CHAR(I*4+76,"FFFFF
FFFFFFFFFFF")
1940 CALL CHAR(I*4+80,""):: 
NEXT I
1950 C=1
1960 FOR I=7 TO 14
1970 CALL COLOR(I,C,C+1):: C
=C+2 :: NEXT I
1980 SUBEND
```


"Well, there's *your* problem."

Sneggit

by Robert Brown and Stephen Judd

Sneggit is a game requiring fast reactions, quick planning, and a small dose of caution. Those evil forces of chaos have struck the henhouse, scattering all the eggs around where some hungry snakes are ready to make a quick meal if you do not stop them. But watch it! These snakes are not about to just slink around while you rescue all the eggs; they think you taste pretty good, too.

You control the chicken who was left to guard the henhouse. By moving next to an egg and pressing the fire button, you can pick up the egg onto your back, where you can carry it to a nest and drop it off safely. Defend the nest long enough, and your egg hatches into a baby chick, who runs off the screen and leaves you another egg to save. The snakes will leave eggs in the nest as long as you are there to guard the nest, but slip away to get another egg and your nest egg is fair game. If you can put 16 eggs into nests, the chicken in the sky will give you a helper to take over in case of snakebite. There are a lot of weeds and rocks scattered around the barnyard which just get in your way and make it hard to get around. The snakes are smaller than you, so they can sometimes go places you cannot. Snakes can also go down snake holes when they feel like it, and they come up where you least expect them. Watch your nests! You can face down a snake, but turn your back on one and you are his next meal.

The chicken is controlled by either a joystick or the standard keyboard keys for directional control (S/J, D/K, E/I, and X/M for left, right, up, down, and Q/Y for fire). Since Sneggit can be played by two players, only the active player's keys are enabled. During play, the REDO key will take you back to the secondary title screen where you can restart another game or go back to the main title screen. Pressing the BACK key during play takes you back to the main title screen.

A round ends when all eggs have been eaten or after the snake has you for lunch. Three rounds make a game. Your highest score is kept as long as Sneggit is running. The scores of the last game played are shown for comparison.

Sneggit may be played by one or two players. There are three levels of play; Novice, Advanced, and Expert. The Novice level is set up for easy learning and a slower playing speed. After you master the Novice level, you can move up to the Advanced level. Advanced gives twice as many points per egg, but is quite a bit faster to play. When even the Advanced level is too easy, you can move on to the Expert level. This scores four times the points as the Novice level, but there are two snakes, and they move much faster.

Each colour egg gives a different number of points when it is picked up; see the help screen for a picture which shows the scoring. The help screen also shows you where you can pick up eggs and where you must stand to drop them into the nest. You may call up the help screen from the main title screen by pressing the AID key. If you are not close enough to the nest, or if you hold the fire button too long, you will drop the egg and smash it. The FIRE button needs a lighter touch for each advanced level of play, so you need a very quick touch at the Expert level.

Eggs are scored once when you pick them up, again when you successfully put them into the nest, and one You must protect the eggs until they hatch.

A nest can only hold eight eggs at one time; if you try to put more than eight into a nest, the new eggs will fall out onto the ground. It is not a problem but you can prevent it from happening by not putting all your eggs in one basket. Sometimes eggs can be very hard for you to see, but the snake still knows where they are.

I cannot tell you any more. If you want to learn more, you will have to play the game! Happy egging!

continued from page 20

Table 4 - TMS9902 ACC Control Register Format

Address	Name	Description
7	SBS1	Stop Bit Selection. ([0,0]: 1.5 bits;
6	SBS2	[0,1]: 2 bits; [1,x]: 1 bit)
5	PENB	Parity Enable
4	PODD	Odd Parity Select
3	CLK4M	Input Divide Select
2	-	Not used
1	RCL1	Character Length Select ([0,0]: 5 bits;
0	RCL0	[0,1]: 6 bits; [1,0]: 7 bits; [1,1]: 8 bits.

The Power of Relational Expressions

by Jim Peterson, Tigercub Software

What the heck are those, you say? You may well ask. The "blue book" that came with your computer says nothing about them, and most of the programming tutorial books on the subject are equally silent. If you waded through the computerese and mathematese text of the User's Reference Guide, you found them discussed on page II-14 under Relational Expressions and on page II-51 under IF THEN ELSE, but you probably did not realize their potential. Then, you graduated to Extended BASIC and found those easy to use, in the clear logical expressions AND, OR, NOT and XOR, and you looked no further.

So, what can a relational expression do? Nothing that cannot be done without it. But it can often do the job so much more compactly, so much more efficiently, and therefore so much faster!

So, let us learn to use them. And let us learn in plain English, not computerese. The following may not be technically correct, but it is the way it all works out.

First, every expression has a true/false value, which is entirely different and separate from the value of the variables or numbers or strings it contains. On the TI99/4A, a false statement has a value of 0, which is easy to remember. A falsehood is worth nothing. Unfortunately, a true statement has a value of -1, which does not fit in too well! On some other computer you may have learned that a true expression has a value of +1, but on the TI99/4A it is -1.

So, in ...F=7 :: IF F=8 THEN...., F=7 has a value of -1 because obviously F does equal 7, and F=8 has a value of 0 because it is not true.

Second, when an IF refers to a variable without an "=" sign, it means "<>0". For instance, IF X THEN 1000 means "if X is more or less than 0, if it is not 0, if it is anything other than 0, then go to 1000".

Third, the computer will try to use the expression mathematically before it tries to interpret its true/false value. Remember that everything within parentheses is worked first. For instance ...X=1 :: Y=2 :: IF (X=1)+(Y=2) THEN 1000... Since both are true, this works out to IF (-1)+(-1)<>0 THEN 1000, and since -1 plus -1 is not 0, we go to 1000. On the other hand, X=1 :: Y=2 :: IF X=1+Y=2 THEN 1000 will first be calculated as X=1+Y, which comes out as X=3, and then as X=3=2, which has a true/false value of 0 (false) because X=3 has a true/false value of 0 (false), not 2!

Finally, always remember that a variable keeps its previous value until the calculation of an entire equation is completed. X=3 :: X=X+(X+3)*X-X/X^X+(X=0) is worked as X=3+(3+3)*3-3/3^3+(3=0).

Now that you have assimilated this vast knowledge, how can it be used? The most common way is in the expression IF (X=1)+(Y=2) THEN 200. In this case, if it is true that X=1 but Y does not equal 2, then -1+0 is <>0 so you go to 200. If X is not 1 but Y=2, then 0+-1 is still <>0, and if X=1 and Y=2 then -1 plus -1 is still <>0, so you still go to 200, but if X is not 1 and Y is not 2 then 0+0 is not <>0 so you do not. Of course, in Extended BASIC, you could simply write IF X=1 OR Y=2 THEN 200.

If you want to go to 200 only if X=1 or if Y=2 but not if both are true, then you can write IF (X=1)+(Y=2)=-1 because either -1 plus 0 or 0 plus -1 will equal -1. In Extended BASIC, this is the "exclusive OR", IF X=1 XOR Y=2.

And if you want to go to 200 only if both are true, you can write IF (X=1)+(Y=2)=-2, or more commonly IF (X=1)*(Y=2) because if either or both are not true the multiplication by 0 will give 0. In Extended BASIC, this is IF X=1 AND Y=2 .

And you can write more complicated versions, carefully watching your parentheses, such as IF (X=1)+((Y=2)*(Z=3)) which translates to IF X=1 OR Y=2 AND Z=3.

So, if you are programming in Extended BASIC, why bother with all those parentheses? Why not just use OR and AND? In the above cases, that is true. But you have not yet begun to see the power of relational expressions!

Since the true/false value is a numeric value, it can be used in calculations, and it does not have to be used with an IF statement.

For instance, this is a statement that I have used within a loop to alternate control of the two joysticks between two players....X=X+1+(X=2)*2 :: CALL JOYSTICK(X,Y,Z) . In this, the first time around, X has not been given a value, so the equation is read X=0+1+(0=2)*2 and, since 0 does not equal 2, 0+1+(0*2)=1 and joystick #1 is activated. Next time around, X=1 and X=1+1+(1=2)*2 gives X a value of 2, since 1=2 has a true/false value of 0. The 3rd time around, X now has a value of 2, and X=2+1+(X=2)*2 which is worked as X=2+1+(-1)*2 and then X=2+1+(-2) which is X=2+1-2 and X=1 again!

If you think that is neat, look at this one from the Airport Area UG newsletter, credited to Robert Cooley. X=X=0 :: CALL JOYST(X+2,Y,Z). Here, the first time around, X does equal 0 so the statement X=0 has a true/false value of -1 so X=-1 and X+2 activates joystick #1. Then X=-1 so X=0 has a true/false value of 0 so X=0 so X+2 activates joystick #2, and so on! Of course, you could also write IF X=1 THEN X=2 ELSE X=1 if you prefer.

Another example: A=INT(10*RND):: B=INT(10*RND):: FOR J=A TO B.. Now, if the random B happens to be smaller than the random A, the loop falls through with nothing happening. You could add a line IF A>B THEN T=1 ELSE T=-1 and FOR J=A TO B STEP T . But why not just FOR A TO B STEP (B<=A)+ABS(A<=B) . If B<A then -1+ABS(0) gives a STEP -1 to count backwards, but if A<B then 0+ABS(-1) gives STEP 1, and if A+B then 0+ABS(0) equals STEP 0! Here is another example - 100 INPUT "SCREEN COLOR? ":S :: FOR SET=1 TO 14 :: X=SET+1-(SET>=S):: CALL COLOR(SET,X,X):: NEXT SET.. That changes the character sets to colors 2 to 16 in sequence, skipping over whatever color has been selected for the screen.

Strings can also be manipulated.

```
100 P$(1)="S"
110 INPUT "HOW MANY? ":N :: PRINT "THE PRICE IS "&
    STR$(n)& " DOLLAR" & P$(ABS(N>1)) :: GOTO 110
```

Or, more efficiently:

```
100 INPUT "HOW MANY? ":N :: PRINT "THE PRICE IS "&
    STR$(N)& SEG$("DOLLARS",1,7-(N>1)) :: GOTO 100
```

However, it is also possible to overdo it. The following routine will read key input to move the cursor around the screen in all 8 directions, stopping at the borders or travelling along them if struck diagonally. However, it requires so many calculations for each key input that it is not the fastest method for accomplishing this.

```
100 CALL CLEAR :: R=1 :: C=3
110 CALL KEY(3,K,ST):: IF ST=0 THEN 110
120 C=C+((K=82)+(K=68)+(K=67))*
    (C<32)-((K=87)+(K=83)+(K=90))*(C>2)
130 R=R+((K=90)+(K=88)+(K=67))*
    (R<24)-((K=87)+(K=69)+(K=82))*(R>1)
140 CALL HCHAR(R,C,42):: GOTO 110
```

So, for compact, efficient, programming, learn to use the relational expressions! But also learn when not to use them!

For Sale

TI99/4A console complete with power supply and modulator
TI Peripheral Expansion Box
RS232 card
32K bytes memory expansion card
TI disk controller card
Single sided drive
TI joysticks
5 modules
Completed RAMdisk
Epson RX80 printer
Manuals and lots of software
Take the lot for $1400. Ring Jack on (02)546 2234 after hours. Will negotiate either way.

The Asynchronous Communications Controller (ACC)

by Mack McCormick, USA

Many people have asked me to explain, in more detail, how they can directly access the RS232 ports on their TI99/4A for such things as terminal emulators, BBS Software, protocol file transfer software, and so on. Many of these people have made some attempt at trying to do single byte I/O using the standard DSR entries, but soon get frustrated with the hassle that is involved, not to mention the lack of ability to tell when a character has arrived at the port, detecting carrier and ringing signals and other such useful features. In this article we will give you an overview of the TMS9902 Asynchronous Communications Controller (ACC) chip used in the TI99/4A RS232 card and define the input CRU testing bits. The final article will explain the details of programming the TMS9902 directly with some code examples.

TMS9902 Overview

Figure 1 shows the pinout of the TMS9902 ACC chip.

```
             ---------
  INT(L)-  | 1    18 | -VCC
 XOUT(H)-  | 2    17 | -CE(L)
  RIN(H)-  | 3    16 | -SYSCLK(L)
CRUIN(H)-  | 4    15 | -CRUCLK(H)
  RTS(L)-  | 5    14 | -S0(H)
  CTS(L)-  | 6    13 | -S1(H)
  DSR(L)-  | 7    12 | -S2(H)
CRUOUT(H)- | 8    11 | -S3(H)
     VSS-  | 9    10 | -S4(H)
             ---------
```

Figure 1. TMS9902 pinout

The TMS9902 is fabricated using NMOS and uses standard TTL level signals on all inputs and outputs. The device is capable of handling asynchronous communications, with word lengths from 5 to 8 bits with 1, 1.5 and 2 stop bits, full parity checking (even/odd/none), with built in data rate generation (no external bit rate generators required) up to 19,200 bits per second. The TMS9902 also includes an interval timer with resolution from 64 to 16,320 microseconds. Interfacing between the TMS9902 and the TMS9900 host environment is done through the Communications Register Unit, the CRU. As you look at Figure 1 you can see S0 to S4, which are the CRU address lines that are used to interface the TMS9902 to any TMS9900 series microprocessor. When the chip enable (pin 17) is asserted the TMS9902 is placed on the bus and the function requested is determined by S0 to S4 and the CRU control logic. CRUCLK (pin 15) provides the CRU timing while the system clock (pin 16) provides the timing for the rest of the 9902. CRUIN (pin 4) and CRUOUT (pin 8) are the CRU input and output bits, respectively. In addition to the CRU logic, additional lines are provided for Request To Send (pin 5), Clear To Send (pin 6) and Data Set Ready (pin 7). The received data is on pin 3, the transmitted data is sent out on pin 2, and the interrupt line for the TMS9902 is on pin 1. Finally, +5 volts goes on pin 18 and logic zero (ground) is on pin 9. For RS232 applications, the widely used 75188/75189 buffer chips are used for bringing out the transmit/receive signals, as well as DSR, RTS and CTS. In the TI99/4A, RTS and CTS are tied together.

The interrupt pin will go to a low logic level when certain conditions occur during the operation of the TMS9902. These will be explained shortly.

TMS9902 Input Assignments

Accessing the T̈ ··02 in your program requires the use of assembly language with the CRU instructions TB, SBO, SBZ, LDCR and STCR. Check your Assembly Language reference manual for the format of the instructions.

Table 1 - TMS9902 ACC Input CRU Bit Assignments

Address					Base	Name	Description
S0	S1	S2	S3	S4	10		
1	1	1	1	1	31	INT	Interrupt
1	1	1	1	0	30	FLAG	Register load control flag
1	1	1	0	1	29	DSCH	Data set status change
1	1	1	0	0	28	CTS	Clear to send
1	1	0	1	1	27	DSR	Data set ready
1	1	0	1	0	26	RTS	Request to send
1	1	0	0	1	25	TIMELP	Timer elapsed
1	1	0	0	0	24	TIMERR	Timer error
1	0	1	1	1	23	XSRE	Transmit shift reg empty
1	0	1	1	0	22	XBRE	Transmit buffer reg empty
1	0	1	0	1	21	RBRL	Receive buffer reg loaded
1	0	1	0	0	20	DSCINT	Data set status change intr
1	0	0	1	1	19	TIMINT	Timer interrupt
1	0	0	1	0	18	-	Not used, always 0
1	0	0	0	1	17	XBINT	Transmitter interrupt
1	0	0	0	0	16	RBINT	Receiver interrupt
0	1	1	1	1	15	RIN	Receive input
0	1	1	1	0	14	RSBD	Receive start bit detect
0	1	1	0	1	13	RFBD	Receive full bit detect
0	1	1	0	0	12	RFER	Receive framing error
0	1	0	1	1	11	ROVER	Receive overrun error
0	1	0	1	0	10	RPER	Receive parity error
0	1	0	0	1	9	RCVERR	Receive error
0	1	0	0	0	8	-	Not used, always 0
0	0	x	x	x	7-0	RBR7 -RBR0	Receive buffer register, received data

Table 1 shows the input bit assignments which go as follows.

Bit 31: Interrupt (INTR). When programmed, any of the following conditions, if met, will set this bit and also cause the interrupt pin on the TMS9902 to be asserted: DSCINT, TIMINT, XBINT, RBINT. In addition, the respective bit that raised this condition will also be set.

Bit 30: Flag (FLAG). Set when any of the load register commands are issued or when the break bit is set (See Table 2.)

Bit 29: Data Set Status Change (DSSC). Set when DSR or CTS changes logic states and is stable for 2 clock cycles. This bit is cleared when an output to bit 21 (DSCENB) is made.

Bit 28: Clear To Send (CTS). Indicates the logic level on the Clear To Send pin. Note that the bit present will be the inverse of the logic level on the pin.

Bit 27: Data Set Ready (DSR). Indicates the logic level on the Data Set Ready pin. The bit will be inverse of the input.

Bit 26: Request To Send (RTS). Indicates the logic level of the Request To Send pin. The bit will be the inverse of the pin.

Bit 25: Timer Elapsed (TIMELP). Set when the timer register equals zero. Cleared by an output to bit 20 (TIMENB).

Bit 24: Timer Error (TERR). This bit is set when the timer has reached zero and when TIMELP is set. This is raised when the software fails to recognize TIMELP and reset it before another interval passed. This is also cleared by an output to bit 20.

Bit 23: Transmitter Shift Register Empty (XSRE). Set when the Transmitter shift register is not sending data (also indicates that XOUT is at a high logic level.) When this bit is at logic zero, character transmission is in progress.

Bit 22: Transmit Buffer Register Empty (XBRE). When set, this indicates that there is not a character waiting to be transmitted. It is set to a logic zero when a character is written to the transmit buffer register.

Bit 21: Receive Buffer Register Loaded (RBRL). When set, indicates that a complete character has been assembled in the receive buffer register. It is cleared by an output to bit 18 (RIENB).

Bit 20: Data Set Status Change Interrupt (DSCINT). Set when either DSR or CTS changes state and the Data Set Status Interrupt has been enabled.

Bit 19: Timer Interrupt (TIMINT). Set when the timer has elapsed and the Timer Interrupt has been enabled.

Bit 17: Transmitter Interrupt (XBINT). Set when the Transmit buffer register is empty and the Transmitter Interrupt has been enabled.
Bit 16: Receive Interrupt (RBINT). Set when the Receive Buffer register is loaded and the receive interrupt has been enabled.
Bit 15: Receive Input (RIN). The logic level of the RIN line at the time that the input was sampled.
Bit 14: Receive Start Bit Detect (RSBD). Set during transition of the first bit of a word. Normally not used.
Bit 13: Receive Full Bit Detect (RFBD). Set when the first bit of the character (excluding the start bit) is received, and is cleared when the character is completed. Not normally used.
Bit 12: Receive Framing Error (RFER). Set when the stop bit of the received character is a logic zero (should be a logic one), indicating a transmission error. Reset when a character with a correct stop bit is received. It is recommended that this bit only be sampled when RBRL is set.
Bit 11: Receive Overrun Error (ROVER). Set when another character is being assembled in the TMS9902 and the previous characeter has not been acquired from the TMS9902 by the software and the RBRL flag reset.
Bit 10: Receive Parity Error (RPER). Set when the parity of the incoming character is incorrect, indicating transmission error. Reset when a character with correct parity is processed.
Bit 9: Receive Error (RERR). Set when RFER, ROVER, or RPER is set.
Bits 7-0: Receive Buffer Register (RBR). This is the register where the received character is stored. For data lengths shorter than eight bits the data is right justified in the register.

Table 2 - TMS9902 ACC Output CRU Bit Assignments

Address					Base	Name	Description
S0	S1	S2	S3	S4	10		
1	1	1	1	1	31	RESET	Reset
					30-22	-	Not used
1	0	1	0	1	21	DSCENB	Data Set Status Change Interrupt Enable
1	0	1	0	0	20	TIMENB	Timer Interrupt Enable
1	0	0	1	1	19	XBIENB	Transmitter Interrupt Enable
1	0	0	1	0	18	RIENB	Receiver Interrupt Enable
1	0	0	0	1	17	BRKON	Break On
1	0	0	0	0	16	RTSON	Request To Send On
0	1	1	1	1	15	TSTMD	Test Mode
0	1	1	1	0	14	LDCTRL	Load Control Register
0	1	1	0	1	13	LDIR	Load Interval Register
0	1	1	0	0	12	LRDR	Load Receiver Data Rate Register
0	1	0	1	1	11	LXDR	Load Transmit Data Rate Register
					10-0		Control, Interval, Receive and Transmit Data Rate, and Transmit Buffer Register

CRU Output Bit Assignments

(Also see Table 2.) Described below are the Output CRU Bit assignments and their function:
Bit 31: Reset (RESET). Setting this bit high causes the TMS9902 to reset the entire chip. All interrupts are disabled, RTS is set high, all register flags (LDCTRL, LDIR, LRDR, LXDR) are set high and the BREAK flag is reset. No operation to the TMS9902 should be performed for 11 clock cycles after setting this bit.
Bits 30-22: Not used.
Bit 21: Data Set Change Interrupt Enable (DSCENB). Setting this bit causes the TMS9902 to generate an interrupt when input bit 29 (Data Set Status Change) is set. Resetting this bit disables the DSCINT interrupt. Writing any value to this bit will reset input bit 29.
Bit 20: Timer Interrupt Enable. Setting this bit causes the TMS9902 to generate an interrupt when TIMELP (Input bit 25) is set. Resetting this bit disables the TIMELP interrupts. Writing any value to this bit will reset input bit 25.
Bit 19: Transmit Buffer Interrupt Enable (TBIENB). Setting this bit causes the TMS9902 to generate an interrupt when XBRE (input bit 22) is set. Resetting this bit disables XBRE interrupts. This bit dos not affect the current value of XBRE.
Bit 18: Receiver Interrupt Enable (RIENB). Setting this bit causes the TMS9902 to generate an interrupt when RBRL (input bit 21) is set. Resetting this bit disables RBRL interrupts. Writing any value to this bit causes RBRL to reset.
Bit 17: Break On (BREAKON). Setting this bit causes the XOUT line to go to logic zero whenever the transmitter is active and XBR and XSR (both transmitter buffer and shift registers) are empty and inhibits loading characters into the transmit buffer register. Resetting this bit causes the transmitter to return to normal operation.
Bit 16: RTS On (RTSON). Setting this bit causes the RTS handshake signal to be active (low). Resetting this bit causes RTS to go high after XSR and XBR are empty and BRKON is reset. (RTS will not go inactive until a character has completed transmission.)
Bit 15: Test Mode (TEST). Setting this bit causes RTS to be internally connected to CTS, XOUT to RIN, DSR held internally low, and the interval timer to operate at 32 times its normal rate. This is useful for debugging programs that are transmitting and receiving data and for tsting to insure that the internal TMS9902 functions are working properly. Resetting this bit causes the TMS99D2 to return to normal operating mode.
Bits 14-11: Register Load Control Flags. These bits are described below.
Bit 14: Load Control Register (LDCTRL). This bit when set (either explicitly or after the RESET bit is set) causes the control register in bits 0-7 to be loaded into the TMS9902. Resetting this bit (or upon reception of the last control register bit) causes the control register to be inhibited from loading.
Bit 13: Load Interval Register (LDIR). Setting this bit causes the next 8 bits (bits 0-7) to be loaded into the TMS9902's interval timer register. The bit is reset when the last data bit is loaded or a logic zeo is written to LDIR. To load this regiter, LDCTRL must be reset (See Table 3.)
Bit 12: Load Receive Data Rate Register (LRDR). Setting this bit causes the next 11 bits (0-10) to be loaded into the TMS9902's Receive data rate register. The bit is reset when the last data bit is loaded or a logic zero is written to LRDR. To load this register LDCTRL and LDIR must be zero.
Bit 11:Load Transmit Data Rate Register (LXDR). Setting this bit causes the next 11 bits (0-10) to be loaded into the TMS9902's Transmit data rate register. The bit is reset when the last data bit is loaded or a logic zero is written to LXDR. To load this register LDCTRL and LDIR must be zero. Note: If both the transmit and receive data rates are the same both registers can be loaded simultaneously by setting both LRDR and LXDR.
Bits 10 through 0: Data. Depending on the control bits set, either data rate registers, the interval register, the control register or data to be transmitted are written to these bits.

Table 3 - TMS9902 ACC Register Load Selection

LDCTRL	LDIR	LRDR	LXDR	Register Enabled
1	x	x	x	Control Register
0	1	x	x	Interval Register
0	0	1	x	Receive Data Rate Register*
0	0	x	1	Transmit Data Rate Register*
0	0	0	0	Transmit Buffer Register

The Control Register

The control register is used to define the character size, the parity of the word, the number of stop bits and the operating frequency of the TMS9902. Table 4 outlines the bit positions and their settings. Of importance is bit 3. For the TI99/4A environment this bit should always be zero. The PENB bit determines if parity checking is done, if set, parity checking is done, if reset no parity checking is done. The PODD bit determines if the TMS9902 will check for and generate odd parity. If set, odd parity is processed, if reset even parity is processed.

continued on page 17

Exploring TI DOS

by Terry Atkinson, Canada

With thanks to: SUBFILE 99 (the SOURCE), with the permission of Mike Amundsen, Rich Stanford and others too numerous to mention, who have contributed their two cents worth.

Buzzwords

DF: Diskfixer
AU: Allocatable Units (sectors)
VIB: Volume Information Block (AU0)
<: Less than
>: Denotes hexadecimal number (or greater than)
DM2: Disk Manager II
FDC: Floppy Disk Controller
FDIR: File Descriptor Index Record
FDR/B: File Descriptor Record/Block (AU2 to AU23)

General

To have a look at a disk, sector by sector, you will have to use one of the many Disk Fixer (DF) programs. The TINS library has the TI in house Disk Fixer (DISKO) available for a small price. Others, such as the Navarone V1, Navarone V2 (Potts) and a couple of Forth versions are also kicking around. The TI version has a capability of toggling between hexadecimal and ASCII, but has no printer dump routine. I prefer the Potts program. A great deal can be learned from experimenting with a DF program. Practically, you can also repair a blown directory, or restore files which have been accidentally erased. Yes, when you delete a file, it is still on the disk! Only the pointers are deleted. You will understand this better as you practice what you will (hopefully) learn from this article.

General physical layout

The TI disk is formatted with 360 Allocatable Units (AUs). These AUs are more commonly called sectors, and can hold 256 bytes each. Of course, with TI Dos and a Teac drive, there can be 360x2=720 AUs, and with the new CORCOMP DSDD controller, each disk is capable of handling 1440 AUs. For simplicity sake, and because this is the more common configuration, we will stick with the standard TI SSSD set up. Unfortunately, TINS newsletter formatting precludes giving actual examples of sectors, but I am sure the information contained here will be sufficient to get you started. Practice is the key. Now, initialize a disk and put a file, any file, on it. Run the DF, and load sector 0. Let us have a look at what is there.

Sector 0: Volume Information Block (VIB)

This AU contains information about the disk and how it is formatted. Here is a complete breakdown of all addresses on sector 0.

Address	Contents
0 0009	Disk Name (up to 10 characters). If the disk name is less than 10 characters, it is padded with space characters (>20).
000A-000B	Number of sectors on disk. (>0168=360, >02D0=720, >05A0=1440)
000C	Number of sectors per track (>09=9, >10=16, >12=18)
000D-000F	Characters "DSK" (>44534B). When a disk is initialized, these characters are placed on the disk. If, during a disk access for example, the FDC does not find these characters, a "disk not initialized" error is issued. This may be important in fixing a blown disk. Remember it!
0010	If this address contains >20, the disk is not proprietory. The disk is flagged as proprietory (not copiable by DM2) if it contains >50.
0011	Number of tracks per side (>28=40, >23=35)
0012	Number of sides (01=single sided, 02=double sided)
0013	Density (01=single density, 02=double density)
0014-0037	Unknown. Set to zero. Anyone have any information?
0038-end	Sectors used/available bit map flags (1=used, 0=unused). The first byte (at >0038) is for sectors 0 to 7, the next byte (at >0039) would be for sectors 8 to 15, the third byte for sectors 16 to 23 and so on. You would decode the value into binary, and read from right to left. For example, let us say >0038 contained the value >8B. Convert to binary: 1000 1011. Read from right to left flags sectors 0, 1, 3 and 7 as used, and sectors 2, 4, 5 and 6 as available. You could continue in this manner and find out all the actual sectors that are in use on the disk. The flags are updated whenever files are added to or deleted from the disk. Information for the second of a DSSD formatted disk starts at address >0065 and ends at >0091.

OK, here are a few things to test out. See if you can read the sector as explained above. Does it contain what you would expect? Now change the contents of address >0010 to >50. See if you can copy the disk with DM2. Reload the DF and change >0010 back to >20. Now put any hexadecimal code in address >D, >E, >F. Now see if you can access the disk. Experiment! When you are satisfied with AU0 restore it to normal and read in sector 1 (AU1).

Sector 1: File Descriptor Index Record (FDIR)

This sector (AU1) contains the alphabetical index of all files on the disk. It is fairly straight foward, and simple to figure out. Each 16 bit word contains the sector number of the File Descriptor Record (FDR) for a file. The list is terminated by a (word) >0000. Since these pointers are in words, and an AU has 256 bytes, this means that the disk has a potential of 256/2=128 files. However, remember the (word) >0000 to terminate the list is required. This leaves 128-1=127 possible files per disk. When a new file is created, the FDRs (2 to 33) are scanned, sorted and sector pointers are set into AU1 in the new alphabetical order. Should the alphabetical order be corrupted, the binary search method used to locate files could be affected, and some files may become inaccessible.

By way of an example, let us say that address 0000/01 contain the value 0005. That means that the first file FDB (alphabetically) is located in sector 5. Easy enough?

That is all there is to sector one (AU1). Load a few files onto a blank disk and get into your DF. Examine AU1. See if the above makes sense, then change a few numbers around to see the results. Try a few experiments of your own.

Sectors 2 to 33 inclusive: File Descriptor Block (FDB)

The FDBs contain all the information specific to a file. Each file has its own FDB for the purposes keeping track of file descriptions, begining and ending AUs, and where the sectors belonging to fractured files are, etc. Below is a break down of each address of an FDB.

Address	Contents
0 0009	Filename (up to 10 characters). Shorter names are padded with "space characters. (>20).
000A-000B	Unknown
000C	Bit map file type. (bit0: 0=data file, 1=memory image (program) file; bit1: 0=display, 1=internal format; bit3: 0=unprotected, 1=protected; bit7: 0=fixed, 1=variable record length; bits2,4,5,6:? (always 0). Or, (for those that hate binary) >01=Program or memory image, >00=display fixed >02=internal fixed >80=display variable >82=internal variable. If the file is protected, a value of >08 is added to the above, so if you find an >88 in this address, it means a write protected DIS/VAR file.
000D	Number of records per sector.
000E-000F	Number of sectors allocated to the file. The value does not include the FDB, therefore, add 1 (as DM2 does) to get the total sectors used by the particular file.
0010	For variable length data (and memory image) files, this byte contains the number of bytes used in the last sector for that file. Simply stated, it is an EOF pointer or offset.

```
0011        Maximum record size of data file.
0012-0013   File record count. The value in >0013 is
            the most significant byte of the count, and
            the value at >0012 is the least significant
            byte of the count.
0014-001B   Unknown. Any information would be
            appreciated.
001C-end    Block Link Pointers.
```

Files are put on the disk in a first come first served basis. The first file written will be placed at >0022, and each subsequent file will be placed immediately following the first. If the first file is deleted, a newer file will occupy the sectors formerly used by the deleted file. If this space is not large enough, the new file is "fractured" and the remainder is placed following the last sector used. The block link pointers keep track of this "fracturing". It is also the most complicated to understand. To quote one source: "Each block link is 3 bytes long. The value of the second digit of the second byte followed by the 2 digits of the first byte is the address of the first sector of the fracture. The value of the third byte followed by the first digit of the second byte is the number of sectors the fracture occupies". Let us see if we can simplify the above with an example.

Let us say addresses >001C to 1F contain the values 0B F1 03 xx (xx=does not matter):

```
Address: 00-> 1C 1D 1E 1F
Value  :      0B F1 03 xx
Nybble#:      12 34 56 78
```

Rearranging the above to conform with the above statement produces:

10B..which is the first sector of the fracture (sector 267 dec);

03F..which is the number of sectors in the fracture (63 dec).

At this point, I should mention that disks copied with DM2 are copied file by file (as happens with file copy by any disk manager), thereby eliminating fractures entirely. Since disk accesses to a program or data with many fractures slows down the retrieval process considerably, I recommend you occasionally copy the disk over to a new disk to remove the fractures for the sake of maximum speed, and for backup purposes. Not because you will overextend the fracturing capability, you can have up to 76 different fractures of the same file!

OK, so now something for you to do. Initialize a disk and put a short file on it. Call it "A". Then save the same file as "B". Now, delete "A" and place a larger file called "C" on the disk. Boot your DF and take a look at sector 2. This should contain the file "C", and it should be fractured. See if you can locate the fractured portion. An interesting idea here: can you figure out a way to read a program file like a data file? This has some very interesting possibilities. After some experimentation, you should be able to restore "blown directories" or retrieve improperly closed files.

Sectors 34 to 359: File storage space.

This is the space where all data and programs are held. Since it is impossible to give specific examples, I leave it to you to explore. Just remember that BASIC programs are stored on disk in memory image format (just as they sit in memory), so do not expect to see your line numbers in chronological sequence all the time.

A hint

Since the maximum record size is 255 bytes (fixed) or 254 bytes (variable) and a record is never fractured, plan your record size such that it will fit the maximum number of records into each sector. For example:

DF80: 3 records per sector (80x3=240 bytes used; 16 unused).

IF10: 25 records per sector (10x25=250 bytes used; 6 unused).

DF130: 1 record per sector (126 bytes unused, what a waste!).

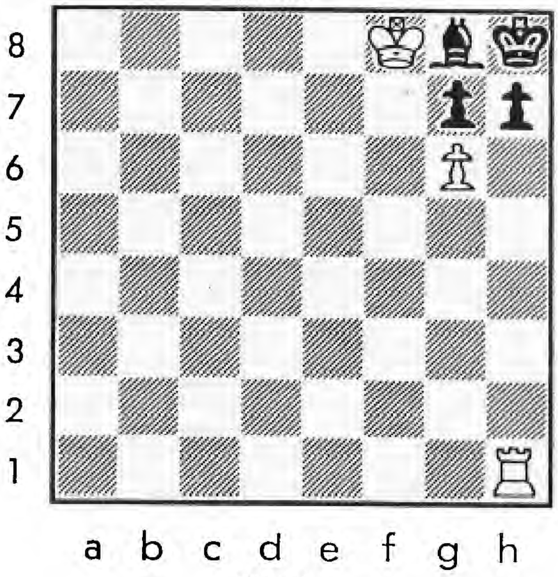
(Note also that Display type file length is limited to around 148 to 155 characters.) (By Extended BASIC? ED)

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

Chess Quiz

by George Meldrum

(Black)



(White)

White to play and mate in two moves

In a book designed to test your chess, and appropriately titled "Test Your Chess", the above diagram appears. The problem is to find a move for White, called the "Key Move", to which any Black response is unsuccessful in preventing checkmate by White's next move. The story goes that it took an electronic brain a little over 15 minutes to solve this teaser back in about 1950.

Time had come to power up the TI99/4A and plug in the old TI Video Chess Cartridge. The Video Chess Cartridge has a "SET UP A PROBLEM" selection on the menu. So set it up and 83 seconds later, bingo, there is the answer. But what of those dedicated computers that devote their entire electronic lives to the eight by eight matrix? Drag out the SciSys Chess Computer and, whiz, 2 seconds later, there is the answer.

The TI Video Chess Module, although slow, does have one most endearing feature for problem solving. That feature is that it does not stop after finding a solution to the problem but rather goes through every possible move so that all solutions are found. The solutions are then displayed at the end of the search. Chess problems are meant to have only one solution, however sometimes a second, unintended solution creeps in, this in chess terms is refered to as a "cook". If you are into composing chess problems then the TI Video Chess Cartridge is an invaluable aid for testing for "cooks".

By the way, I did manage to personally solve the problem above in a time that somehow escapes my (convenient) memory. See how you fare.

Viatel Screen dump

by Peter Schubert

Those who have the AT RS232 system with PIO port, either on the Mini-PE System or the Multifunction Card, can now use our favourite Viatel program to dump screens to printer thanks to the efforts of Shane Ferret, otherwise known as WEASEL. The original version only dumped to the TI or Corcomp RS232 Cards due to its unusual method of accessing the printer. Contact Shane or myself for a copy of the program.

A Graphics Programming Language (GPL) Primer

Routine to Read/Edit Text from KSCAN
by Mack McCormick 74206,1522

I have always avoided delving into the study of GPL because I felt it was too difficult, cumbersome, executed too slowly, and had little to offer. Boy, was I wrong. It makes writing routines used by BASIC a snap in assembler. For example, I recently needed a routine to read text from the screen which would allow full editing including erase, insert, delete, quit, bonk tone at right margin, and enter, up arrow, down arrow. I also wanted the neat auto-repeat feature used by TI, where there is a slight pause before the key takes off repeating. I began to consider writing the routine but then remembered that an identical routine resided in GPL in GROM (Graphics Read Only Memory) in the console. I first considered using the routine from GROM but then remembered that it added the screen offset of >60 to each character and I did not need that. I could have done some fancy trick to make it work but decided to convert GPL to 9900 assembler code.

You will find two programs here. One to link you to the routine in console GROM with a CALL LOAD from E/A BASIC and the identical (almost) routine in 9900 assembler code ready to link into any program you may have that needs this utility. I have included in the 9900 routine the actual GPL code used by the Pre-Scan routine of the monitor so you may see what conversions were necessary. Try reading the GPL instructions (marked with three *) to get a flavour for GPL. If the GPL gets in the way of using the routine in your program, you may delete the GPL statements, although they will have no effect if they are allowed to remain.

It has really become obvious to me why TI invented GPL, although I used to condemn them for it. The major reason is that it uses about 59% of the code that straight assembler would use. GROM, as you may know, is only used by TI, and is a chip which supplies a byte at a time to a memory mapped address and auto-increments to the next byte (like VDP RAM) unless you change the address to be read from. It is a great way to save memory. TI calls it medium speed memory. It is 6K bytes big and resides on 8K boundaries. It is an ideal medium to hold the console BASIC routines, because the TMS9900 CPU chip in the console can only directly address 64K. The GPL actually does not execute any code. GPL is interpreted in console ROM beginning at >0024 and extending to >D18. This interpreter is straight assembler code which acts as directed by the GPL bytes coming from the GROM. Hence you see one reason TI BASIC is slow. It is an interpreted by GPL and GPL is interpreted by assembler. Two interpretations! Instructions in GPL usually have two operands and most instructions can access RAM, GROM, or VDP RAM. Most instructions are single byte operands unless the operand is preceded by a D for double operand. GPL uses two stacks, a data stack at >83A0 and a subroutine address stack at >8380 (this allows arbitrary nesting of subroutines). Here are a few types of instructions:

<u>Data transfer</u> - Single/Double Byte
- Block to Block
- Formatted Block Moves

<u>Arithmetic</u> - Add, subtract, multiply, divide, negate, absolute value.
<u>Logical</u> - AND, XOR, shifts.
<u>Condition</u> - Arithmetic and logical.
<u>Branching</u> - Conditional and unconditional.
<u>Bit manipulation</u> - Set, reset, test.
<u>Subroutine</u> - Call, return.
<u>Stack operations</u> - Push and pop.
<u>Miscellaneous</u> - Random number, KSCAN, coincidence detection, sound, input and output.

The closest language to GPL is assembler and any experienced assembler programmer should have little difficulty learning GPL. One major difference is the use of MACRO instructions by the GPL assembler, such as REPEAT.. ..UNTIL and IF.. ..THEN.. to 99000 assembly language.

A few words about how memory is addressed. Here are a few of the most common ways and their syntax.
5 represents the decimal byte 5.
>33 represents hexadecimal 33.
&110011 represents binary 110011.
#5506 represents the decimal number 5506.
:A: is the ASCII equivalent >41.

Well this has been a very general overview of GPL. Let us look at some actual GPL source code and my interpretation of the 9900 assembler equivalent. This routine could have been shortened but I tried to keep it as close to GPL as possible. Hope you enjoy it. If you have questions just ask. My 6 GPL manuals cover thousands of pages and we have just skimmed the surface here. I plan to write a GPL disassembler and interpreter to convert GPL to 9900 object code within the next six months if my schedule permits. That should make the job easy!

```
       DEF  START
       REF  KSCAN,VSBW,VSBR,GPLLNK
* just a little routine to test subroutine *
START  LWPI WS
       MOVB @H00,@KEYVAL scan entire keyboard
LOOP   BL   @READLN
       DATA >002,>2FE start, end positions
       JMP  LOOP
****************************************************
* This is the console GPL READLN routine at (>2A42 in
* GROM 1) converted to 9900 assembler.  Interprets
* backspace, insert, delete, and forward.  Uses
* scratch pad RAM.  Total number of characters may be
* limited by changing the start value of ARG+2 (upper
* limit) and entering at READL1.  VARW is the start
* address of the field.  VARA is the current highest
* write address.  Entering at READL1 allows us to
* pre-specify the minimum number of characters to be
* read for default creation.  Entering at READ00
* allows specification of the initial cursor position.
* In this case ARG+6 has to be set to the cursor
* position and ARG+4<>0.  Programmer responsibility to
* ensure that VARW <= ARG+6 <= VARA <= ARG+2.  ARG+4
* indicates if the line has been changed.  If so,
* ARG+4=0.
* This is a possible call:
* BL    @READLN
* DATA >1DF,>35D lower,upper screen limits
****************************************************
* Equates *
WS     EQU  >8300                         my workspace
ARG    EQU  >835C
VARW   EQU  >8320                      abs lower limit
VARA   EQU  >832A                  current end of line
TEMP   EQU  0                   R0 used for temp storage
TEMP1  EQU  1              R1 used for addl temp storage
R1LB   EQU  WS+3
TEMP2  EQU  2
TEMP3  EQU  3
TIMER  EQU  >8379        VDP timer inc every 1/50 sec.
KEYVAL EQU  >8374                   keyboard to scan
RKEY   EQU  >8375                           key code
STATUS EQU  >837C                    GPL status byte
* Constants
* (Should be EQU with byte values in code to save
* memory.)
H00    BYTE 0
H01    BYTE 1
HFF    BYTE >FF
H508   DATA 508
H60    DATA 60
H14    BYTE 14
H766   DATA 766
BREAK  BYTE >02
DLETE  BYTE >03
INSRT  BYTE >04
CLRLN  BYTE >07
BACK   BYTE >08
FORW   BYTE >09
DOWN   BYTE >0A
MVUP   BYTE >0B
CHRTN  BYTE >0D
CURSOR BYTE >1E
SPACE  BYTE >20
VARV   BYTE 0           (this is at >8301 in GPL but I
*                             used >8300 for workspace)
VAR1   DATA 0             auto repeat counter (this is
*                                1 byte at >830D in GPL)
```

```
       EVEN
READLN
* The GPL code stores >35D at ARG+2 but to give more
* utility replaced with the next two lines of code.
***    [illegible] >35D,@ARG+2              GPL double store
       [illegible]  *R11+,@VARW   start address of the field
       MOV  *R11+,@ARG+2                    upper limit
***    DST  @VARW,@VARA
       MOV  @VARW,@VARA                 nothing entered yet
* VARA should point to a space location or end of
* field
READL1
***    ST   1,@ARG+4              store byte=1 to ARG+4
       MOVB @H01,@ARG+4          means no change in line
READL2
***    DST  @VARW,@ARG+5     had to use ARG+6 because
***                            of word boundary problems
       MOV  @VARW,@ARG+6          position cursor at
*                                      start of field
READ00
***    CLR  @VAR1      clear byte. I had to use word
***                    because 9900 is so much faster
       CLR  @VAR1               counter for auto repeat
* This is where we return to exit insert mode.
READ01
***    CLR  @ARG+7      used ARG+8 because had to use
***                          ARG+6 & ARG+7 already
       MOVB @H00,@ARG+8 normal operation mode
***    ST   C[illegible]R,@VARV
       MOVB @[illegible]SOR,@VARV               VARV used for
*                                    cursor/character
READ$1
* Input 1 char and alternate cursor and character for
* blink
***    EX   @VARV,RAM(@ARG+5)  exchange @VARV with
***                      what is at location ARG+5 in VDP
       MOV  @ARG+6,TEMP            exchange VARV,ARG+6
       BLWP @VSBR
       SWPB TEMP1
       MOVB @VARV,TEMP1
       BLWP @VSBW
       MOVB @R1LB,@VARV
***    [illegible] @TIMER
       [illegible] @H00,@TIMER        set vdp timer to zero
***    $REPEAT                 macro. repeat code until
***                                     $until is true
L00001 LIMI 2              enable interrupts so the VDP
*                          timer (>8379) can increment
       LIMI 0             disable interrupts so the VDP
*                                will not get messed up
***    SCAN                       scan the keyboard
       BLWP @KSCAN                scan for a character
***    BS   [illegible]        branch on cond bit (eq) set
       MOVB [illegible] ,@STATUS            equal bit set?
       JNE  READS2                 found a new character
***    INC  [illegible]   increment the byte @VAR1 by one
       INC  [illegible]       increment auto-repeat counter
***    $IF  @RKEY .NE. >FF THEN macro    if RKEY not
***         EQ >FF then execute the following code
***       otherwise skip to the $END IF terminator
       CB   @RKEY,@HFF                        old key?
       JEQ  L00002                               yes
***    $IF  [illegible]1 [illegible]. 254 THEN higher or equal
       C    [illegible]1,[illegible]       hold old key for a while
*                               had to double 254 to slow
*                                    down assembly code
       JLT  L00002              before starting repeat
***    SUB  30,@[illegible]                     subtract byte
       S    @H60.[illegible]1          control repeat rate
***    B    REA[illegible]             unconditional branch
       JMP  REA[illegible]
***    [illegible] IF
***    [illegible] IF
***    [illegible]L @TIMER .H. 14   terminator for repeat
***                             until higher than 14
L00002 CB   @TIMER,@H14
       JLE  L00001          time next character switch
***    BR   READ$1         branch cond bit reset, used
***                        to save one byte of memory
       JMP  READ$1             restart char blink cycle
READ$2
***    CLR  @[illegible]
       CLR  @[illegible]          clear auto repeat counter
READ$3
***    $IF  @VARV .NE. @CURSOR THEN
       CB   @VARV,@CURSOR     if NE exchange again
       JEQ  L00003
***    EX   @VARV,[illegible]RG+5)
       MOV  @ARG+6.[illegible]          exchange VARV,ARG+6
       BLWP @VSBR
       SWPB TEMP1
       MOVB @VARV,TEMP1
       BLWP @VSBW
       MOVB @R1LB,@VARV
***    $END IF
***    $IF  @RKEY .L. : : THEN     if RKEY less than
***                          space then execute code
L00003 CB   @RKEY,@SPACE         if .LT. space then
*                                       control char
       JLT  L00004
       B    @L0000C
* This is where you would trap all control codes
* Handle break char first
*      CB   @RKEY,@BREAK
*      JNE  LABLE
* back arrow - space back one position
***    $END IF
***    $IF  @RKEY .EQ. BACK GOTO RBACK       GOTOs do
***                        not require an END IF term
L00004 CB   @RKEY,@BACK                  back arrow?
       JNE  B00002            to fix out of range error
       B    @RBACK
*                           had to double 254 to slow
*                                  down assembly code
       JLT  L00002            before starting repeat
***    SUB  30,@VAR1                     subtract byte
B00003 CB   @RKEY,@INSRT
       JNE  L00005
***    ST   1.[illegible]RG+8
       MOVB @[illegible].@ARG+8         set insert mode flag
***    $END IF
* delete - delete the current char
***    $IF  @RKEY .EQ. DLETE THEN
L00005 CB   @RKEY,@DLETE
       JNE  L00006
***    CLR  @ARG+4
       MOVB @H00,@ARG+4   indicate a change in line
***    $IF  @VARA .DNE. @ARG+6 THEN       the d means
***                  double or word of memory compare
       C    @VARA,@ARG+6                   empty line?
       JEQ  L0001F                                yes
***    [illegible] @VARA,@ARG
       [illegible]  @VARA,@ARG            move everything from
*                                           the right
***    DSUB @ARG+5,@ARG double byte (word) subtract
       S    @ARG+6,@ARG   of the cursor to the left
***    MOVE @ARG FROM RAM(1(ARG+6)) TO RAM(@ARG+6)
***      this is a block move of @ARG bytes of VDP
***          RAM from what is at addr ARG+6 plus 1
***          to what is at address ARG+6. In short
***        move everything on screen one byte lower.
       MOV  @ARG,TEMP2                         counter
       MOV  @ARG+6,TEMP
       INC  TEMP          move @ARG from RAM(1(ARG+6))
*                                       to RAM(@ARG+6)
L00008 BLWP @VSBR
       DEC  TEMP
       BLWP @VSBW
       INCT TEMP
       DEC  TEMP2
       JNE  L00008
***    DDEC @VARA          decrement the word (double)
***                                       at VARA
       DEC  @VARA             pre-update end of string
***    $IF  RAM(@VARA) .EQ. : :+OFFSET GOTO READ01
***                       OFFSET is screen offset >60
       MOV  @VARA,TEMP
       BLWP @VSBR
       CB   @TEMP1,@SPACE
       JNE  B00001            to resolve out of rg error
       B    @READ01
***    DINC @VARA               increment the word of
***                                  memory at VARA
B00001 INC  @VARA
***    $END IF
***    ST   : :+OFFSET,RAM(@VARA)
L0001F MOV  @VARA,TEMP
       LI   TEMP1,>2000
       BLWP @VSBW
***    BR   READ01
```

```
       B    @READ01
* clear - clear the entire input line
***    $IF  @RKEY .EQ. CLRLN THEN
L00006 CB   @RKEY,@CLRLN
       JNE  L00009
***    $REPEAT
***    ST   : :+OFFSET,RAM(@VARA)
       MOVB @SPACE,TEMP1
CLRLIN
       MOV  @VARA,TEMP          so we can fiddle with
*                                         value
       BLWP @VSBW
       DEC  @VARA              pre-update end of line
***    $UNTIL @VARA .DL. @VARW     double less than
       C    @VARA,@VARW    up to and incl first pos
       JHE  CLRLIN
***    [illegible] [illegible]A
       [illegible]     [illegible]A         undo last subtraction
       CLR  @ARG+4
       MOVB @H00,@ARG+4             indicate change
***    BR   [illegible]
       B    [illegible]             restart everything
***    $END IF
* general exit point
***    $IF  [illegible] .[illegible]. CHRTN THEN
L00009 CB   [illegible]   [illegible]TN   only react on cr/up/down
       JEQ  L0000A
***    $IF  @RKEY [illegible]. MVUP THEN
       CB   @RKEY, [illegible]
       JEQ  L0000A
***    $IF  [illegible] .[illegible]. DOWN GOTO READ$1
       CB   [illegible]
       JEQ  L0000A
       B    @READ$1
***    $END IF
***    $END IF
***    $IF  @V[illegible] .DEQ. @ARG+2 THEN    double equal
L0000A C    @V[illegible] @ARG+2    check for block on last
*                                            position
       JNE  L0000B
***    [illegible] [illegible](@V[illegible] .NE. : :+OFFSET THEN
       [illegible]  [illegible]RA,[illegible]
       BLWP @VSBR
       CB   TEMP1,@SPACE                     blocked?
       JEQ  L0000B
***    DINC @VARA
       INC  @VARA     point beyond last char in line
***    $END IF
***    $END IF
L0000B RT                       enter the current line
***    $END IF                (this is from the $IF that
***                            checked for CTRL codes)
* insert routine *
***    $IF  @[illegible] 8 .[illegible]. 0 THEN                 insert
L0000C CB   @[illegible] 8, [illegible]              insert mode
       JEQ  L0000D
READ$4
***    [illegible] @[illegible],@ARG
       [illegible] @[illegible],@ARG   use ARG as temp for insert
***    $WHILE @ARG .DH. @ARG+6
L0000F C    @ARG,@ARG+6        move everything up to
*                                     cursor location
       JLE  L0000E
***    DDEC @ARG
       DEC  @ARG               copy lower location to
*                                         higher one
***    ST   RAM(@ARG),RAM(1(ARG))    go from high to
***                                  low in VDP RAM
       MOV  @ARG,TEMP
       BLWP @VSBR
       INC  TEMP
       BLWP @VSBW
       JMP  L0000F
***    $END WHILE                terminator for while
***    $IF  [illegible]RA .[illegible]. @ARG+2 THEN
L0000E C    @VARA,@[illegible]-2  only update VARA as upper
       JHE  L0000D
***    DINC @VARA
       INC  @VARA             has not been reached yet
***    $END IF
***    $END IF
***    ST   [illegible] [illegible]ARG+6)
L0000D MOVB [illegible]              display the character
       MOV  @ARG+6,TEMP
```

```
       BLWP @VSBW
***    CLR  @ARG+4
       MOVB @H00,@ARG+4    indicate change in line
READ05
***    $IF  @[illegible]-5 .DEQ. @ARG+2 THEN
       C    @[illegible]-6,@ARG+2          hit right margin?
       JNE  L0002F
***    CALL TONE2         call another GPL routine in
***                                 this case bonk
       MOVB @H00,@STATUS      clear the status byte
*                               before accessing GPL
       BLWP @GPLLNK         give a bad response tone
       DATA >0036
***    BR   [illegible]D$1
       B    [illegible]D$1              stay in current mode
***    $END IF
***    DINC @[illegible]6
L0002F INC  @[illegible]6            update current address
***    $IF  @ARG+6 [illegible] @VARA THEN
       C    @ARG+6,@[illegible]A   check for last new high
*                                            limit
       JLE  L00010
***    DST  @ARG+5,[illegible]
       MOV  @ARG+6,[illegible]        update new high limit
***    [illegible]D IF
***    [illegible] @[illegible] .DL. >2FE GOTO READ$1
L00010 C    @[illegible]A,@H766
       JHE  L00011          to fix out of range problem
       B    @READ$1              still some space to go
L00011
* this is where we could scroll the screen if needed
* update pointers if you scroll *
***    [illegible]  SCROLL               scroll the screen
***    [illegible]  28,[illegible]
*      S    @H28,@VARA         back to start of line
***    DSUB 32,@VARW
       BLWP @VSBR
       INC  TEMP
       BLWP @VSBW
       JMP  L0000F
***    $END WHILE               terminator for while
***    $IF  @[illegible]A .DL. @ARG+2 THEN
L0000E C    @[illegible]A,@ARG+2  only update VARA as upper
       JHE  L0000D
*      S    @H32,@ARG+6 current cursor position also
***    BR   [illegible]$1
       B    [illegible]D$1          start with something else
* forward cursor move
RFORW
***    CLR  @ARG+8
       MOVB @H00,@ARG+8             leave insert mode
***    BR   [illegible]05
       B    [illegible]A05                use rest of logic
* back cursor move
RBACK
***    $IF  @ARG+5 [illegible] @VARW
       C    @ARG+6,[illegible]W         check bottom range
       JLE  L00012
***    [illegible] @ARG+6
       [illegible] @ARG+6
***    $END IF
***    BR   [illegible]1
L00012 B    @READ01
       END
**************************************
* This is a routine to directly access *
* the GROM READLN routine. Use CALL    *
* LOAD("DSK1.filename") and CALL LINK  *
* ("DSK1.start") from E/A BASIC to see *
* it because of screen offset.         *
**************************************
       DEF  START
GPLWS  EQU  >83E0          address for GPL work space
H00    BYTE 0
WS     BSS  >20                          my workspace
       EVEN
START
       LWPI WS                point to my workspace
       LI   R0,>2
       MOV  R0,@>8320  start screen address for scan
       MOVB @H00,@>837C      clear the status byte
*                          so we do not get an error
       BLWP @GPLLNK      link to the routine in GROM
       DATA >2A42
       MOVB @H00,@>837C       return to the calling
*                                program on enter
```

continued on page 26

How to Disassemble Programs

by Phil R. Storey

One of the ways to learn how to program in assembler is to look at what other people have written. To do this you must first disassemble a program and have it printed out on paper. There are a number of disassembler programs on the market, including those in commercial, freeware and public domain. You should have no trouble finding one. The most favoured one seems to be DISkASSEMBLER, marketed by Millers Graphics. This program is capable of disassembling programs directly from disk, as well as those in memory. There are a number of features built into this program that aid re-assembly of the disassembled programs. The manual is fairly large and comprehensive.

For your first disassembly project I would suggest a program that is small, (less than 8K), such as the SAVE utility on the Editor/Assembler disk. Before you create a written disassembly of the program, I would suggest going through the program looking for, and noting, blocks of text and/or data. (Text and data produce about 5 times as much code when they are disassembled as mnemonics rather than data). Text blocks are easy to recognise, however, data blocks can be more elusive. They become easier to detect once you get a feel for program flow.

Data blocks can usually be found by looking at the mnemonics. They are usually preceded by either a B, RTWP, or a jump instruction. The mnemonics themselves will seem to have no direction. Dead give aways are code sections that contain DATA or INVALID MNEMONIC statements or strings of similar statements such as compare instructions. A compare is usually followed by a conditional jump. (The exception here is when the compare is used to add 4 to a register, e.g. C *R1+,*R1+ could be used to add 4 to the contents of R1).

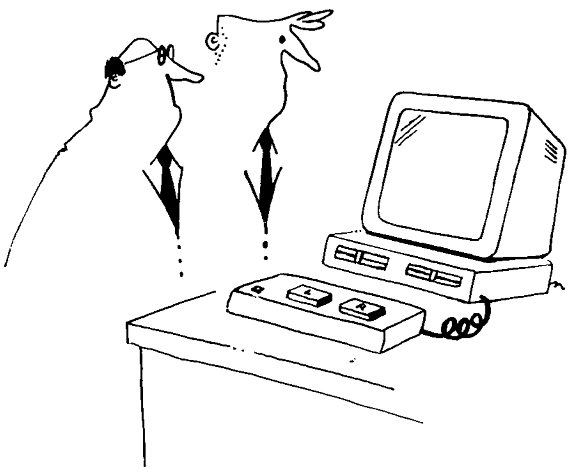
Once you think you have written down where all of these data and text blocks are, you can proceed to disassemble the program. Disassemble the text and data blocks as such and the remaining code as mnemonics. At this stage I usually take two pieces of paper and label one BLWPs and the other BLs. Since we are starting with a relatively small program, use a single piece of paper divided into two sections. While looking through the program code you will usually find many references to each BLWP and BL subroutine. As you look through the code write down each BLWP and BL address (and label, if there is one) you come across. When you figure out what each routine does, write a short description of it on your paper. You can refer to this paper when you come across further links to the same routine. This technique can save you a lot of time when working on larger programs.

Your job now is to go through your program listing making written comments as to what each instruction, or group of instructions, is trying to accomplish. At first you may have to look up each instruction in the Editor/Assembler manual. As you progress you will find that you need to reference it less and less. You will probably find that it will be necessary to go through the program several times. Each time you go through the code you should discover the purpose of at least one routine. Continue this process until you have commented all of the code.

At some point you may find yourself going through the code and not resolving a single routine. You have probably reached your level of comprehension and will either need help or further experience before you can proceed any further.

The Editor/Assembler Quick Reference Card has several errors in the way it shows the composition of the instruction formats on page four. See the Editor/Assembler manual (pages 65 to 73) for the correct breakdown. Quick access to the ASCII tables and the like is an advantage. I have several of those taped to the wall around the desk where my computer sits. I should warn you that, if you continue this work, your Editor/Assembler manual will start to show signs of considerable wear. When I take mine down from the shelf, several pages fall out, and a few have gone missing. Tape is holding it together because the cover is torn.

Do not be suprised to find yourself waking up in the middle of the night with a solution to some routine that you were working on earlier. Good luck, and pleasant dreams.


'... and this system was designed for the two finger typist'.

continued from page 11

```
1200 REM
1210 REM
1220 REM *WRAP DATA*
1230 REM
1240 DATA THIS LINE WAS ORIGINALLY VERY LONG AND IT HAS
   BEEN SHORTENED SO THAT IT WILL APPEAR ON THE SCREEN
   WITHOUT CUTTING ANY WORDS OFF.
1250 DATA IT'S REALLY AMAZING WHAT CAN BE DONE WITH A
   LITTLE PATIENCE AND PERSERVERANCE!
1260 REM
1270 REM *FILL/B*
1280 REM
1290 FOR XL=1 TO LEN(M$)
1300 IF LEN(M$)=28-XI THEN 1360
1310 IF SEG$(M$,XL,1)<>" " THEN 1340
1320 M$=SEG$(M$,1,XL)& " "& SEG$(M$,XL+1,LEN(M$)-XL)
1330 XL=XL+1
1340 NEXT XL
1350 GOTO 1290
1360 RETURN
1370 REM
1380 REM *WRAP/B*
1390 REM
1400 X1=0
1410 M$=M$&" "
1420 X2=POS(M$," ",X1+1)
1430 PRINT SEG$(M$,X1+1,X2-X1);
1440 IF X2=LEN(M$)THEN 1470
1450 X1=X2
1460 GOTO 1420
1470 RETURN
```

continued from page 25

```
       LWPI GPLWS
       B    @>0070
* You could have placed an end statement here and
* REF'd GPLLNK instead of using this routine.
* GPLLNK routine *
GPLLNK DATA UTILWS,XGPL    vector for the GPLLNK BLWP
UTILWS EQU  >2094
SUBSTK EQU  >8373
FLAG2  EQU  >8349
SVGPRT EQU  >2030
H20    BYTE >20
       EVEN
XGPL
       MOVB @SUBSTK,R1
       SRL  R1,8
       MOV  *R14+,@>8304(R1)
       SOCB @H20,@FLAG2
       LWPI GPLWS
       MOV  @SVGPRT,R11
       RT
       END
```

The Forth Column

Forth Forum <3>
by George L. Smyth

This month I will conclude the discussion of mathematical operations, and begin to talk about stack manipulation. There are more mathematical operators than I have included in this session, but most of those are floating point routines, and may be discussed sometime in the future. To tell you the truth, I have not worked much with the floating point words because they do not offer any speed advantage over BASIC, and BASIC being easier and quicker to program, I tend to write those routines in that language.

Also this month I am presenting a program I downloaded from Compuserve. At first I had difficulties understanding how the program worked, but after figuring it out, I found that this program was going to be invaluable as far as converting Forth programs that, having been downloaded, are in a DIS/VAR 80 format, to the Forth format required by that system. I did not find the 'HELP' files provided very useful, so I went ahead and rewrote them, along with a few other things, to make the program a bit more user friendly. Of course, I may not have succeeded as far as my 'HELP' files are concerned, but the program does work, and you can always contact me if you have problems. So, on with this month's tutorial.

More Mathematical Operations

- (nl n2 --- nl-n2)

The minus sign subtracts the value on the top of the stack from the value just below it, and places the result on the stack. This is straightforward, in that to perform the operation '5-3=?', you need merely enter '5 3 - .' to have the answer printed to your screen.

/ (nl n2 --- quot)

The slash divides the integer on the top of the stack into the integer just below it, and places the result on the stack. I purposely used the term "integer" in this definition because this word, more than the other three arithmetic operators, shows the exclusive use of integer, or whole number, values. TI Forth does include a floating point extension, but that will not be covered for quite a while. The routine '50 5 / .' will print "10", the result, to the monitor. Because we are involved only with integer values, the routine '50 6 / .' will print "8" to the monitor. This is because 50 can be divided by "6" 8 times. The remainder is dropped. I think of this routine in the same respect as I would decide how many dimes I can get for my quarter. What if we need to know the remainder? We use 'MOD'.

MOD (nl n2 --- rem)

This word divides the number on the top of the stack into the number just below it and returns the remainder to the stack. For instance, '50 6 MOD .' would print a "2" on the monitor, as that is the remainder of the division. If the number is evenly divisible, "0" is returned to the stack. If we want to find both the quotient and the remainder of a division, instead of using both / and MOD, we use /MOD.

/MOD (nl n2 --- rem quot)

/MOD divides the value at the top of the stack into the number underneath it and returns to the stack both the remainder and the quotient. If we entered '50 6 /MOD . .' we would find "8 2" printed, the quotient and the remainder.

Summarizing:

```
50 6 / .        prints 8
50 6 MOD .      prints 2
50 6 /MOD . .   prints 8 2
```

It would be difficult to overemphasize the fact that the more you experiment with the system and try things out, the easier working with Forth (or any language for that matter) will become. One problem you may have come across while playing with these words, is that of manipulating large numbers. What happens when you try to multiply 5,000 by 25? This can be done, but because your Forth system assumes your values are single word in size, special instructions must be implemented to accomodate these conditions. Because I am getting tired of discussing only the mathematical aspects of Forth, this subject will be explored more fully some other time. For now we will allow the limitation of using single word length numbers, that is integers between -32768 and 32767. I explored this aspect extensively because the words and their meaning, as far as stack manipulation is concerned, must be fully understood before you can go any further, so be sure you have a good understanding up to this point.

Stack Manipulation

Unfortunately, the stack is not always arranged in the exact order we would like. For instance, in a BASIC statement, certain mathematical operations are performed before others because of a hierarchy, as explained on page 41 of your Extended BASIC manual. In Forth, this hierarchy does not exist. Therefore, the values you wish to manipulate must be on the stack in the order you wish to manipulate them. To find the value of the expression "(10/2)+7" you would enter '7 10 2 / + .'. By the time your system reads the slash, the top two numbers on the stack are 2 and 10. These numbers are divided and the result, 5, is placed on the stack. The "+" word is next executed, which takes the two numbers on the top of the stack, 5 and 7, adds them, and places the result, 12, on the stack. The period takes the "12" and prints it to the screen. This is all fine, but how do you evaluate the expression "(14+1)/5"? Without manipulating the position of the values on the stack, there is no way to perform this operation with one line of code. Here are some words which will do just that.

DUP (n --- n n)

This word copies and returns the value at the top of the stack. If all you had on the stack was "4", after entering 'DUP' the stack would hold "4 4". An application that comes to mind immediately is that of finding the square of a number. Since the square of a number is that number times itself, the word could be defined thus:

: SQUARED DUP * . ;

Now if we enter '4 SQUARED', the number "4", after being placed on the stack, would be copied by the word DUP. Now the top two values on the stack are 4. The asterisk multiplies these two numbers and the period prints the result, 16.

DROP (n ---)

The DROP word simply removes the value at the top of the stack. If your stack values are '5 3 1' and you execute 'DROP', your stack now contains '5 3'. Often, these two words are used in conjunction with timing loops. Because Forth is so fast, executing the BASIC equivalent of "FOR X=1 TO 1000 :: NEXT X"" in Forth takes almost no time. Therefore, by copying the top of the stack and dropping the value, all you are doing is taking up time. DUP DROP results in no change to your program (note: of course there must be some value on the stack before 'DUP' is executed).

SWAP (nl n2 --- n2 nl)

True to its name, this word exchanges the position of the two top numbers on the stack. The use of this word enables us to find the result of the operation (14+1)/5. Here is how to do it : '5 14 1 + SWAP / .'. The plus sign adds the two numbers and places the result, 15, on the stack. The stack containing | 15 5 (the symbol "|" is used to indicate the bottom of the stack. In this case, "5" is the top of the stack value and "15", just below, is on the bottom), SWAP switches the stack to | 15 5. Now the slash word can divide the numbers and the period word prints a "3" to the monitor.

ROT (nl n2 n3 --- n2 n3 nl)

This word moves the third item on the stack to the top. If your stack contained | 4 3 2 1 and you executed ROT, the resultant stack values would be | 4 2 1 3.

I mentioned it last month after the tutorial, but I will explain again if that section was skipped. The stack values can be read by executing the word ".", however this destroys the stack values. What is needed is a way of looking at the stack without affecting the values. If you configured your system in accordance with the directions in my first article, you will have access to the word '.S'. If not, load -DUMP into your system. '.S' (---) displays the stack without affecting the values or position. The "|" signifies the bottom of the stack, with the rightmost value representing the top of the stack.

At this point we can configure the order of the three numbers at the top of the stack in any position, that is abc, acb, bca, cba, bac, cab. Before you continue, try to follow the operations leading to each of these configurations.

```
           | A B C          | A B C          | A B C
Do nothing | A B C     SWAP | A C B      ROT | B C A
             - - -            - - -            - - -

           | A B C          | A B C          | A B C
     SWAP  | A C B      ROT | B C A      ROT | B C A
      ROT  | C B A     SWAP | B A C      ROT | C A B
             - - -            - - -            - - -
```

OVER (n1 n2 --- n1 n2 n1)

The word "OVER" copies the second number down and returns it to the top the stack. If the stack contained | 0 1 2 and 'OVER' was executed, the stack would then hold the values | 0 1 2 1. This is a standard word found in all systems, but can be defined with words we have already learned. Try to write this word on your own, and if you get stuck, I will include the answer in next month's column.

That about does it for this month. After all this, you may be wondering just where you are in terms of writing programs. Well, we still have quite a way to go. The more power you are given, the longer it takes to learn how to efficiently and wisely use that power. Anyway, next month I will cover the Editor used with the TI Forth system. The following month I will introduce looping structures and finally a few simple programs that will use much of the words we have learned up to now. Try playing around with these new words and do not forget to use '.S' to look at your stack values.

The Forum

Thought for the month:

"Computers do not do what you want them to do,
they do what you tell them to do!"

Becoming a new member of Compuserve allowed me the opportunity to engage in some heavy duty downloading of everything I could find relating to TI Forth. One of the biggest problems was the fact that the download file was in a DIS/VAR 80 format, not conducive to the Forth operating system. One of the offerings was a program which converted DIS/VAR 80 files into Forth screens and vice versa. I had previously entered a program proclaiming the same function but unfortunately it had a bug and I did not have time to work it out. This program worked the first time, but because of the lack of understandable documentation, I did not realize it for quite some time. Therefore, I decided to change a few things so that the program could be used by an individual who did not even understand Forth. I feel that it is the duty of the programmer to cater to the lowest common denominator. In this respect, a wider range of individuals will be able to utilize the realization of your programming skills.

Here are a couple of the changes I made:

1.) I changed the method of calling the 'HELP' screens, so that this program can be placed on any series of screens without the necessity of changing the screen number to be loaded.
2.) I used vectored execution to determine the proper routine to be run. The word 'RUN' is universal to both routines.
3.) I rewrote the field descriptor word, so that the knowledge and understanding of this routine need not be necessary.
4.) I rewrote the 'HELP' files so that, hopefully, they are more useful.
5.) I included the word 'IT', so that when the user is finished with the program, all of the words can be erased from the dictionary by entering the phrase 'FORGET IT'.

This routine was written by Jeff Young of Brooklyn, NY, who otherwise did an excellent job documenting the functions of the program. One can see the implementation of the PABS clearly with his explanations.

Type this program onto a series of Forth screens. Good luck and, as always, if you have any problems, give me a call.

George L. Smyth

```
( File -> Screen -> File converter written by Jeff Young )
BASE->R  HEX            ( Enhancements by George L. Smyth )
: IT ;                                              ( 4/85 )

0 VARIABLE BLKSV
0 VARIABLE VEX

: CS 10 SYSTEM 0 0 GOTOXY ; CS  CR CR CR
."     Just a sec ... loading program !"

BLK @ 6 + BLKSV !

: HELP BLKSV @ LOAD ;
: RUN VEX @ EXECUTE ;

-->

( -FTRAN2 - FORTH TO DIS/VAR 80 TRANSFER ROUTINES MCA 19JUL84 )

: BEP 34 GPLLNK ;
: FLNAME 20 PAB-ADDR @ 0A + SWAP
         WORD HERE COUNT >R SWAP
         R VMBW R> N-LEN! ;
( SET UP PERIPHERAL ACCESS BLOCK )
0 VARIABLE    FILBUF  50 ALLOT ( CREATE AN 80 [>50] CHR BUFFER)
              PABS  @  A +     ( DATA IS 10 [>A]BYTES INTO PAB)
              FILBUF     1900  ( PLACE THE BUFFER AT VDP >1900)
              FILE    FILTRAN  ( REFER TO THIS PAB AS FILTRAN )
( SET DISK FILE PARAMETERS       )
: SETFILE     FILTRAN  SET-PAB  ( CALL UP PAB AREA       )
                  , DSPLY       ( SEQUENTIAL, DISPLAY,   )
                    50 REC-LEN ; ( VARIABLE 80 [>50]     )
-->

( -FTRAN2 - FORTH TO DIS/VAR 80 TRANSFER ROUTINES     16AUG84 )
( DEFINE FILE OUTPUT ROUTINE )                        ( Page 3)
DECIMAL

: PUTFILE  CR ." Insert data file disk." CR KEY DROP
           APPND FILTRAN OPN          ( OPEN THE PAB FOR OUTPUT)
1+ SWAP DO I DUP CR ." Copying screen:" . CR
BLOCK 16 0 DO DUP FILBUF 80 BLANKS  (  LOOP FOR 16 RECORDS    )
                  FILBUF   64 CMOVE   (  COPY 64 CHAR FROM BUFR)
                           64 WRT     (  WRITE REC# I [64 CHRS])
                           64 +       (  INC I      ADDR [1 LN] )
              LOOP DROP    LOOP       (  BACK       NEXT RECORD  )
           CLSE                       (  CLOSE THE FILE          )
      BEP  ." Disk file completed. "   CR CR CR QUIT ;

-->

( -FTRAN2 - FORTH TO DIS/VAR 80 TRANSFER ROUTINES     16AUG84 )

( DEFINE FILE INPUT ROUTINE )                         ( Page 4)
: GETFILE   CR ." Insert data file disk." CR KEY DROP
            INPT FILTRAN OPN          ( OPEN THE PAB FOR INPUT)
    1+ SWAP DO I DUP CR ." Loading screen:" . CR
            DUP BLOCK                 ( GET SCREEN ADDRESS    )
            16 0 DO FILBUF 80 BLANKS  ( LOOP FOR 16 RECORDS   )
            RD DROP                   ( READ REC FROM DISK )
        DUP FILBUF SWAP 64 CMOVE      ( MOVE LINE INTO BUFFER )
            64 +                      ( INC BUFFER CHAR COUNT )
            LOOP DROP                 ( BACK FOR NEXT RECORD  )
      BLOCK DROP UPDATE LOOP          ( UPDATE THE DICTIONARY )
            CLSE                      ( CLOSE THE FILE  )
      BEP ." All screens loaded." CR CR CR QUIT ;
-->


( -FTRAN2 - FORTH TO DIS/VAR 80 TRANSFER ROUTINES     16AUG84 )

( FORTH SCREEN TO DIS/VAR 80 FILE ROUTINE)            ( Page 5)

: SCR-FILE ( scr#[s] scr#[e] --- )
    ' PUTFILE CFA VEX !
       CS 9 3         ." Screen to File Transfer"
          9 4         ." ======================="
          CR CR CR CR CR
          SETFILE BEP
   ." Enter 'FLNAME' followed by  " CR
   ." a space and the filename    " CR
   ." Example:  'FLNAME DSK1.FILE'" CR CR
   ." Then type 'RUN'.            " CR
      QUIT ;
-->

( -FTRAN2 - FORTH TO DIS/VAR 80 TRANSFER ROUTINES     16AUG84 )
( DIS/VAR 80 FILE TO FORTH SCREEN ROUTINE)            ( Page 6)
: FILE-SCR ( scr#[s] scr#[e] --- )
    ' GETFILE CFA VEX !
       CS 9 3 GOTOXY ." File to Screen Transfer"
          9 4 GOTOXY ." ======================="
          CR CR CR CR CR
          SETFILE BEP
   ." Enter 'FLNAME' followed by  " CR
   ." a space and the filename    " CR
   ." Example:  'FLNAME DSK1.FILE'" CR CR
   ." Then type 'RUN'.            " CR
      QUIT ;
: INTRO CS 6 6 GOTOXY
      ." Enter 'HELP'    " CR CR ."       'FILE-SCR'" CR CR
      ."    or 'SCR-FILE'" CR CR ; INTRO   R->BASE
```

```
( Help screen )  CS  CR                                        ."
 This program will convert FORTH screens"                      ."
into DIS/VAR 80 files and vice versa,    "                     ."
thus allowing you to download FORTH      "                     ."
programs via modem and convert them to   "                     ."
run on your FORTH system without re-     "                     ."
typing them.                             "                     ."
 FILE-SCR is the word which invokes the  "                     ."
conversion of DIS/VAR files into FORTH   "                     ."
screens. The starting screen you wish    "                     ."
to convert the file to, and the ending   "                     ."
screen the conversion will produce must  "                     ."
be entered before the word is executed.  "                     ."
For instance, a 112 line file to be      "                     ."
converted can be placed on screens 41-47"                      ."
by entering:                             "         —>

( Help screen-page #2 )                                        ."
              '41 47 FILE-SCR'           "                     ."
Care must be taken to assure that the    "                     ."
file is a multiple of 16 lines or an     "                     ."
I/O error will result. Also, it is sug-  "                     ."
gested that the original file be placed  "                     ."
on a new disk and that the screen desti-"                      ."
nation be relatively high, as otherwise  "                     ."
the file could be overwritten if care    "                     ."
is not taken.     * Press any key *      "  KEY  DROP  CS  CR  ."
 SCR-FILE converts a series of FORTH     "                     ."
screens into a DIS/VAR 80 file. The pro-"                      ."
cess is the reverse of the previously    "                     ."
discussed routine. The sta[illegible]     screen   "                     ."
and ending screen of the F[illegible]     program  "                     ."
must be entered before the routine name."      -->


( Help screen-page #3 )                                        ."
For example  '41 47 SCR-FILE'  will in-  "                     ."
duce a routine that will convert screens"                      ."
41-47 into a DIS/VAR 80 file. One note   "                     ."
I made was that if an existing filename  "                     ."
is referenced, the new file does not     "                     ."
destroy the old file, but concatenates   "                     ."
following the old file.                  "                     ."
 Both routines contain self-explanatory  "                     ."
input requirements during the running of"                      ."
the program, but should pose no problems"                      ."
to the user. Upon program completion,    "                     ."
type 'FORGET IT' to remove the words     "                     ."
from the dictionary. If difficulties are"                      ."
encountered, give me a call." CR ." George L. Smyth (703)533-871
0" CR CR ."              * Press any key *    "  KEY  DROP  INTRO
```

o

continued from page 1

The Forth Symposium was held on May 19 and 20 at the University of Technology Sydney, and attracted more than 180 participants from all over Australia and the world. Included in the visitors from the USA were Charles Moore and Elizabeth Rather, who were the first two people to write programs in Forth. They were also the founders of Forth Inc., and shared reminiscences of the early days with the delegates who attended the dinner at the end of the first day. The Symposium was organised by a group of people led by Roy Hill and Paul Wilson, and was probably triggered off by the appearance of the Novix chip, a Forth processor designed by Charles Moore and a hardware design team. This chip runs Forth code extremely quickly, and has led to an increase in interest in Forth as a language to solve control problems in particular. Charles Moore gave two talks which were illustrated with "slides" on a large colour video monitor. Producing the pictures was a Forth computer with a Novix chip on a board smaller than the Mini-PE board. This board had a 3 button keyboard for data entry and produced all the signals for the monitor at 12 Mhz. As well as that it could communicate with a 3.5 inch floppy disk if required. Charles Moore has a rather different philosphy to most computer people. He is not too impressed with hardware and believes in smaller being better, but he is also not very impressed with the conventional approach to software either. Of course the normal approach to software does not include Forth, so I guess that is reasonable from his point of view. He claimed that his company (Forth Inc., pronounced for think) was used as a safety net, in that when others could not get complicated software of any type to work in the specified time, they called in Forth Inc., who then produced a working system in a few weeks. The fact that they are still in business would seem to give credence to his claims. I met a few people at the symposium who have been producing software locally using Forth for several years in a competitive environment, and say that it is the only way they can produce the product in a reasonable time. Also of interest were the number of software products which have a Forth base or flavour, such as ASYST, a data acquisition and data processing package for the PC, and PostScript, a programming language for programming printers like laser printers to generate graphic and text output. There was also an announcement at the symposium by Harris Corporation of a new Forth processor like the Novix chip but enhanced in some directions. It appears that Novix have become rather hard to deal with. Thereis a local company, Maestro, (at Kincumber, NSW) which is making boards with the Novix chip on them for insertion in a PC or as a stand alone unit. What if we could get one which would plug into the PEB, or Mini-PE system? It certainly is a very fast processor, and if you agree with Charles Moore that if you need real numbers you have not scaled your problem correctly, then it could provide the only thing that we do not have at the moment with our beloved little computer, speed.

****

Ross Mudie was telling me that there were a lot of members who came to his tutorial day presentation on interfacing to the TI99/4A who did not know even the first thing about hardware, gates and so on. So for all of you who are in that position and would like to learn something, Ross is preparing a tutorial on logic for beginners, and the first one of these should make these pages next month.

****

While reading the news letters, I came across some rather disturbing comments. First there was this one from Richard Earl in the April 1988 issue of News from the ATICC.

'Now on to our interstate "friends" who complained about the fact that we are reprinting so much interstate material, firstly this newsletter is primarily for the members of ATICC. secondly if you feel the material you publish is not up to our standard then let me know and you wont see any more in our newsletter.'

Richard admits that he was not in a good mood when he wrote the editorial of which this was a paragraph, and he clearly did not have time to proof the editorial to correct errors. I feel that he has allowed his emotions to show in a rather unfortunate way. Who is he talking about? There might be 4 or 5 news letters who could be "interstate", but it seemed to me it must be my comments that have caused this outburst. In an attempt to provide a service to the members of TIsHUG I have been giving a very brief resume of the contents of news letters which come across my console. In the March TND I wrote:

'December 1987 issue of ATICC from Adelaide is rather thick but contains mainly articles copied from MICROpendium and TND.'

I guess the trouble with the written word is that you can read what you want into a sentence. These words would seem to me to be very bland, but in the next paragraph about the Melbourne effort, I comment that TIMES must be all typed in or come from disk as it appeared on fanfold paper straight from the printer. I certainly was not trying to judge anyone's efforts at producing a news letter, as I have enough trouble myself one way and another. I feel that it is a pity that the worst connotation is taken of some rather simple statements, whose purpose was simply to give information. Surely the benefit of doubt should go to the best of intentions rather than the worst of possible intentions? As to whether the contents of the TND can be copied for other news letters, I am all in favour of that. In fact it gives me a thrill to see one of our articles in another place as I assume that is a pat on the back that we are doing a good job in editing and layout. If other editors think that our efforts are good enough to be copied directly, then I am very pleased. In fact we have no statement about giving credit and so on for copies, as we assume that people in the TI community are all good and honourable people by now and will do the right thing by our authors. Of course we print articles from overseas and other sources to which we try to attach accurate credit and we are fortunate in having volunteers to type these in for us if needed.

What I have much more difficulty in understanding, is the attitude of the Hunter Valley group in taking up this editorial comment from Adelaide and endorsing it. Perhaps they will tell us some day how it helps the TI community to carry on like this and encourage bad feeling between user groups. How did they know which group Adelaide was talking about? I still do not know for sure. I shall try to talk to someone in Adelaide when I am there.

Finally, I do not want these words to be interpreted just as criticism of peoples' actions and words. If two or more people have misinterpreted my words (assuming I am the culprit) and have been upset by them then there are probably many more also making the same interpretations. I hope that this will cool everyone off rather than further inflame their passions. There is a lesson for us all in this I think. I hope that everyone will think kindly of everyone else and not assume the worst. If there appears to be a problem then a letter or telephone call to the person involved would probably solve it with less acrimony than a broadside in the media.

****

Now to the newsletters. This is an information service for TIsHUG members to enable them to decide which of the news letters they may wish to read in more detail.

The 99'er Online from Edmonton February 1988 issue has an article by Yves Chevalier on fixing disks, on Form letters by John Harbour and programs on printing listings from files in multiple formats, based on an idea from TIsHUG by Bob Pass and Disk Labeler by G. Kivell and enhanced by Gordon Bradlee. In March 1988 there is an article on arrays by Bob Pass and another on formatting by John Harbour. There is also a good example of a flowchart for those who have trouble finding their way around them.

The April and May issues of CIM from Montreal are virtually all in french which makes it difficult for me to tell you what information may be in them of interest to you. All I can say is that there are articles on assembly (8th part), TI-Runner, Multiplan and PRbase.

The LeHigh 99'er computer group news for February 1988 has a menu flow chart for Funnelweb V4.0 from the Lima Ohio user group, an article on error trapping after RUN from CIN-DAY user group and an article on strings by Jack Shattuck of Delaware Valley users group. Jerry Boyer has a continuing series on the Geneve in these news letters. In the March issue he gives the commands for My-Word. Also in March is an article by Jack Sughrue of Maryland on 'Good old Days'. There are also a number of short programs for producing interesting titles from the Pudget Sound 99er, to dump a line to a printer by John Witham from MICROpendium, inverse video character list from MSP newsletter, quick catalog program or subprogram also from MSP and from Mike Slattery of TIsHUG a program to TRACE to a printer. In the April issue there is an article on PRINT USING, with an program example by Brad Snyder, along with Jerry Boyer's article on the Geneve which looks at DOS.

The Northern NJ 99er's Users Group of April 1988 has a hint for people with lines in the picture. It is suggested that a ground wire connected to the RF box on the TV would help. Chick De Marti (LA 99ers) has a column for beginners and there is a PEB speech synthesizer interface by Joe Spiegel as well as some information on power supplies (from Charlotte TI99/4A user group).

The Ottawa Newsletter of May 1988 has some interesting articles on their local news together with Lucie Dorias on Extended BASIC (DISPLAY AT and ACCEPT AT) and Stephen Shaw on BASIC. There is also part 6 of David Caron's articles on expansion port interfacing; and games hints in code by Henri Monat.

The ROM newsletter from Orange County has a letter from Tony McGovern, an article on Forth by Earl Raquse and some useful general information by Newt.

Network, from the Sacramento 99er users group (SNUG) has an article about Steve Shakleford and his portable. He has put a RS232 card, disk controller card, 32K memory, 2 DSSD disk drives, power supply, Extended BASIC and Editor/Assembler cartridges, 22 cm monochrome monitor, speaker, and keyboard into a portable case. Telco is reviewed by Jeff Braden and [illegible] about FORTRAN.

Spirit of 99 (Central Ohio) April edition has columns by Mickey Schmitt on getting the most from your cassette system, TI-Writer by Stan Katzman, Tips from the Tigercub #48 by Jim Peterson and Impact-99 by Jack Sughrue. There is also an article by Dallas Phillips on building a switch to allow your TI99/4A to use two printers, one at a time. Jean Hall has a review of PLUS! from Jack Sughrue with word processing aids and a short bit on Load interrupt, Hold and Reset switches by Curt Borders. In May there are the columns by Mickey Schmitt on a high speed cassette loader, TI-Writer by Stan Katzman, Tips from the Tigercub #49 by Jim Peterson and Impact-99 by Jack Sughrue, in which he talks about the early days and Jim Peterson's influence. There is also an interesting story from Jim Peterson, an introduction to a new data base program FIRSTBASE by Warren Agee, to a new genealogical program STIRPSLINE by Allan Cox and to a graphics program called CLASS by Bill Harms. The article on power supplies (from Charlotte TI99/4A user group) mentioned before also appears.

The Tacoma Informer has some advice from Joe Nollan that the 6264 memory chips from Hyundai do not seem to work on the TI99/4A bus. He also talks about program timing and delays. There is an article about the TI-XPO-88 in Las Vegas and some programming hints on RUNning programs from other programs.

The March TI-SIG newsletter from San Diego also has an article on the TI-XPO-88, another on the Octopus display by Waldo Hamilton and a status report on the United 99/4 Data Base. The April issue has Arto Heino and John Paine's Poor man's disk system with good comments, and they liked our bumper TI-Writer issue. Unfortunately they also printed the incorrect version of the tip for entering Extended BASIC from a RAMdisk without accessing drive 1.

April TopIcs from Los Angeles has an editorial deploring the over use of archiving. They are also worrying about the postal costs involved in exchange newsletters and suggest that perhaps individual members should become members on behalf of groups. There is a mini-mod for super extended BASIC by Tom Freeman, a review of Graphics Expander by D. R. Fudge, Tips from the Tigercub #52 by Jim Peterson, some tips including a bug in Funnelweb 4.? by Chick De Marti who also contributes a BASIC tutorial, a review of EZ-Keys by Bill Gaskill and another attempt at creating a data base of all TI99/4A information by Bill Gaskill.

The April edition of ATICC from Adelaide has a long article on the G Graphics Language developed by Gene Krawczyk and written about by Bob Warren. It also promises some future hardware from Colin Cartwright such as RGB interfaces, very large memory expansion cards and the ultimate in digitized graphics boards.

In May the Hunter Valley group newsletter has two articles by Bob Carmany, the first of which looks at the way you can use Mini-mem to store programs from BASIC in 4 different places in memory, and the second looks at using files from BASIC. There is also the announcement of a cheque/credit card management program from Richard Terry, with a review by Al Lawrence. There is an article on Strings by Jack Shattuck from the LeHigh user group, another on communication protocols by L.C. Twiss from the Perth user group, release information on PLUS! from Jack Sughrue and descending characters for Extended BASIC from Richard Terry.

There were two editions from Brisbane. The April one has an article on M-DOS fro the Geneve and another on building an analog to TTL circuit for a monitor for the Geneve. There is also an article on Graphics compatibility by Don MacClellan of the Bluegrass Computer Society and an article by Regena on DEFs from MICROpendium. The May issue is the TI-Faire edition and is extra thick and recommended reading for all.

continued from page 13

The menu program (V7.3 on my system) provides for a default filename for option 2 (Display a File) and for option 3 (Run a Program). If you wish to change these defaults, simply select either 2 or 3 from the primary menu screen, overtype the default filename that appears on the bottom left of the screen with the one that you want, then escape (FCTN[9]), then press FCTN[5] (for edit mode), escape (FCTN[9]) again, then hit any key (except escape) to save the menu program back to your RAMdisk. Note that you need Extended BASIC and that the MENU program must not be protected.

Regional Group Reports

Meeting summary.

Banana Coast	10/7/88	Sawtell
Carlingford	20/7/88	Carlingford
Central Coast	9/7/88	Toukley
Glebe	7/7/88	Glebe
Illawarra	18/7/88	Keiraville
Liverpool	8/7/88	Moorebank
Northern Suburbs	28/7/88	Mona Vale
Sutherland	15/7/88	???

BANANA COAST Regional Group
(Coffs Harbour area)

Regular meetings are held in the Sawtell Tennis Club on the second Sunday of the month. For information on meetings of the Banana Coast group, contact Keir Wells at 9 Tamarind Drive, Bellingen, telephone (066) 55 1487.

CARLINGFORD Regional Group.

Regular meetings are usually on the third Wednesday of each month at 7.30pm. Contact Chris Buttner, 79 Jenkins Rd, Carlingford, (02) 871 7753, for more information.

CENTRAL COAST Regional Group.

Meetings are normally held on the second Saturday of each month, 6.30pm at the Toukley Tennis Club hall, Header St, Toukley. Contact Russell Welham (043) 92 4000

GLEBE Regional Group.

Regular meetings are normally on the Thursday evening following the first Saturday of the month, at 8pm at 43 Boyce St, Glebe. Contact Mike Slattery, (02) 692 0559.

Last meeting: (May, attendance 14) We had a limited demonstration of the SPAD II Mark 2 flight simulator which was appreciated by all. (Thank you Peter for bringing it along.) Our TELCO demonstration involved logging on to 2000&Beyond, Texpac, Prophet and ISO. The program ran faultlessly. As well as these there was the Debug tutorial.

ILLAWARRA Regional Group.

Regular meetings are normally on the third Monday of each month, except January, at 7.30pm, Keiraville Public School, Gipps Rd, Keiraville, opposite the Keiraville shopping centre. Contact Bob Montgomery on (042) 28 6463 for more information.

LIVERPOOL Regional Group

Regular meeting date is the Friday following the TIsHUG Sydney meeting at 7.30pm. Contact Larry Saunders (02) 644 7377 (home) or (02) 759 8441 (work) for more information.

Last meeting was at Stan MacPuckle's house. It was a very small turn out. Stan had a problem with his RAMdisk and this was fixed at the meeting. There were demonstrations of some programs and we had a chat.

Meetings coming up.

July 8th 1988 at Larry Saunders home. 34 Colechin St, Yagoona West. (02)644 7377
August 12th 1988 ???
September 9th 1988 ???

NORTHERN SUBURBS Regional Group.

Hello again, this is Dick Warburton cordially inviting you to attend the next meeting of the Northern Beaches' Group. We have gradually improved our meetings,and have a small but regular group of 5 or 6 attending. Last meeting, we covered the effective use of printers with TI-Writer. We have organized our meetings so that we cover some aspect of FunnelWriter, look at problems in Extended BASIC programming, and cover a main topic as well.

Our meetings for the next two months are as follows:
Thursday July 28th at Dick Warburton's home, 7 Milga Road, Avalon. 918 8132. The main topic will be file handling.
Thursday August 25th at Dennis' home at 24 Woolrych Crescent, Davidson. A technical night: Clock cards; consoles; drives etc.

Naturally we try to fit in as much copying as possible. We also provide something to eat and drink. We welcome visitors. If you want any information please ring Dennis Norman on (02)452 3920, or Dick Warburton on (02)918 8132. See you soon.

SUTHERLAND Regional Group.

Any persons interested in joining the Sutherland Group on the third Friday of each month, are more than welcome. The format of the meetings are very informal as more often than not the conversation digresses onto matters purely social rather than related to computerisation. The supper is not bad either.

Meetings are held on the third Friday of each month. Group co-ordinator is Peter Young, (02) 528 8775. BBS Contact is Gary Wilson, user name VK2YGW on this BBS.

The May Regional meeting, saw a change of venue, with Derek Wilkinson of Gymea making available his home for the evening. One item of interest at the meeting was a new keyboard, with function keys etc., which has been adapted to the TI console. Along with the remote mounted widget board, Derek's computer is looking less like a 'Texas' each visit. Joe D'ambra has completed work on his clock project and two other members have purchased circuit boards and will follow his lead in the coming weeks. Many thanks to Gary Wilson who helped out new member, Kevin Taylor, by replacing a faulty chip on his 32K card, which was causing no end of trouble.

Future meetings will revert to the home of Peter Young at Jannali, on the third Friday of each month at 7.30pm, phone 528 8775.

TIsHUG in Sydney

Regular meetings are normally at 2pm on the first Saturday the month, except January and possibly other months with public holidays on that weekend, at the Woodstock Community Centre, Church Street, Burwood.

Meetings planned this year. July 2 - General interest. This will be a normal, 2pm start meeting, where if the GENEVE has not already been demonstrated at the June meeting, it should be available at this meeting. OK, so you came along to the June full day tutorial and went home with a wealth of knowledge, but now just what did the tutor mean when he said .....? This is what we call a follow up tutorial when you get the chance to ask, and hopefully receive an answer, to those nagging questions. Do not be shy. The quickest way to learn is to ask questions.

August 6 - Swap meeting and market day. Do you have some old modules, software or hardware you do not use anymore? Chances are your unwanted gear is wanted by someone else and this is your chance to sell it, trade it or give it away. This meeting will start at 2pm so bring along your unwanted items and perhaps go home with a few extra $$$ in your pocket. The group can accept no responsibility for goods on sale or to be swapped. All are sold/exchanged on an "as is" basis.

September 3 - Software demonstrations and purchase. Again this will be a normal 2pm start. At this meeting some of the latest software as advertised in MICROpendium will be demonstrated and hopefully we will have imported copies for sale direct to members at reasonable cost.

1(?)/10/88 9am, Full day Tutorial.

All TIsHUG Regional groups are invited to submit items for this column. Send details as mail to SYSOP on the BBS.

The information presented here is obtained from the BBS. Some Regional groups are not advising of their meetings, which makes the maintenance of this file on the BBS very difficult.

---

o