Scene at the Tutorial Day

The Board

Co-ordinator
Dick Warburton   (02) 918 8132
Secretary
Terry Phillips   (02) 797 6313
Treasurer
Rolf Schreiber   (042) 84 2980
Directors
Robert Peverill   (02) 602 4168
Russell Welham   (043) 92 4000

Sub-committees

News Digest Editor
Geoff Trott   (042) 29 6629
BBS Sysop
Ross Mudie   (02) 456 2122
BBS telephone number   (02) 319 1009
Merchandising
Steven Carr   (02) 608 3564
Publications Library
Warren Welham   (043) 92 4000
Software library
Terry Phillips   (02) 797 6313
Technical co-ordinator
Lou Amadio   (042) 28 4906

Regional Group Contacts

Carlingford
Chris Buttner   (02) 871 7753
Central Coast
Russell Welham   (043) 92 4000
Coffs Harbour
Kevin Cox   (066) 53 2649
Glebe
Mike Slattery   (02) 692 0559
Illawarra
Geoff Trott   (042) 29 6629
Liverpool
Larry Saunders   (02) 644 7377
Northern Suburbs
Dennis Norman   (02) 452 3920
Sutherland
Peter Young   (02) 528 8775

Membership and Subscriptions

Annual Family Dues $25.00
Overseas Airmail Dues   AUS$50.00

TIsHUG Sydney Meeting

The next meeting will start at 2 pm on 5th of August at the Woodstock Community Centre, Church Street, Burwood.

Index

They're off

by Geoff Trott

The trials of owning a computer or two! Just when everthing should be running well, problems strike. I told you about the problems with the Editor's hard disk on loan from John Vandermey, when something became corrupted after the visit to Sydney for the full day workshop meeting. At about the same time my own system began to have problems with its RAMdisk, which had been performing well until then. The problems were that whole tracks on floppies were getting destroyed while moving files between RAMdisk and floppy. At the time I was trying to debug my c99 program to put the dictionaries back together (see article this TND) and was doing it on that system because the c99 compiler and the hard disk controller were also doing strange things to files. Having wiped out a few files on RAMdisk and also on floppies, I then had to reconstruct my working environment for c99. This was Funnelweb version 4.13 and c99 version 4.0. Unfortunately for me I had separated the include and library files from the compiler and so spent many frustrating evenings trying to get a program which used to work, working again with the wrong versions of the libraries for the compiler. I went over to Lou's place to make sure that it was not just my system and had the same problems there too. Just to make things interesting, there is an actual bug in the program which causes a spectacular crash if the data is just of the right length in one of the sectors when a change of letter is made. This would appear to be a bug in the c99 file routines as all my program does is to write out strings of various lengths. It appears as if the program almost fills up a sector with the last words starting with one letter and the final '@' followed by the next letter header, which are of one and two characters long respectively, cause something in the routine to change a pointer used by the screen as it fills with strange coloured patterns. Eventually I found a way to make it all run by adding a few more words to the dictionary.

Co-ordinator's Report

by Dick Warburton

Well, I have finally seen a Geneve, a real one, actually working. I was beginning to think that the Geneve was actually a myth, but my faith has been restored. What an anti-climax. First the monitor decided to play up, and then the Geneve became coy, and needed much TLC to get it up and running. It certainly runs fast, and I really like the idea of running Multiplan or TI-Writer in 80 columns at a reasonable speed. What a shame that it was not released earlier together with adequate software.

At the July meeting, in line with our policy to actively support software authors, we collected donations for the author of Funnelweb (Tony McGovern). $160 was raised from only 45 people present. I know that some of you are unable to get to meetings, and are very disappointed that you missed out on such an opportunity to contribute. Well, do not fret, because you can simply mail your donation to Terry Phillips who will pass it on. At the next meeting, the DM1000 program will be actively supported.

Whatever you do, do not miss the August meeting. It is our Buy Swap and Sell day. Get there early and grab a bargain. Help the club make a small amount on each sale. Do not forget that the club has Bankcard facilities available. I expect we will have a great day, and hope to increase the number of hardware selling days during the year.

At the July Directors' meeting, a number of decisions were made.

1. The printer ribbon re-inker is being purchased, and should be available for the July meeting.
2. A console tester was bought by the club for use by the console repair group.
3. The software competition is now under way. Projects can now be submitted at any time to the Directors. Details are available in the May issue of TND.
4. An approach will be made to other home computer user groups, exploring the possibility of a combined home computer show, hopefully in 1990.
5. TIsHUG will be formally listed in the white pages of the phone book as soon as possible.
6. Associate membership of TIsHUG, will, be permitted in a small number of cases, particularly where members are students and have no desire to receive TND.
7. A limited number of Editor Assembler manuals and modules is to be purchased from the States. At this stage the price is not clear, but we may be able to sell them for approximately 30 dollars each.

By the way, if you know anyone who has not renewed their membership this year, contact them personally, and simply remind them. If you know any prospective new members, bring them to the next monthly meeting. It will be a great day.

See you there. Dick Warburton.   o

Assembler Tutorial Weekend

by Ross Mudie

The Assembly Tutorial weekend held on 10th and 11th of June was an outstanding success. A total of 14 members, 4 of whom came from outside the metropolitan area, participated in the hands on beginner's assembly class. Each member brought their own computer and the computers were used to type in the assembly language source file, assemble the code and run the programs. The course included 17 hours of class time; typing in a simple TI BASIC program, typing in an assembly source file to emulate the BASIC program, explanations, assembly, debugging, modifications to provide extra features and of course more debugging at assembly time.

Whilst the class only really touched on a very small part of the overall assembler language, an understanding was gained of the differences between SOURCE, LIST and OBJECT files. Every member of the class went away at the end of the second day with sufficient confidence to "at least have a go" at typing in an assembly program from a book and to try to assemble it.

As teacher of the class I found that I was quite busy, especially sorting out people's problems at assembly time, but I enjoyed the challenge and the ultimate class achievement. A number of the class members expressed the desire for a follow on class which I propose will be held in the next long weekend on Saturday 30th September and Sunday 1st October 1989. (The Monday holiday gives the tutor the chance to recover). Numbers for the course will be limited to 15 maximum, with participants in the course just held being given first preference since it will continue where the previous course ended. (This is also in the school holidays).

Meals were arranged by various members of the course and this worked quite well. It is hoped that similar catering arrangements can be used in the future.

The concept of a weekend, "hands on" course was very successful. As tutor, I felt satisified with the task, where on the multi-subject tutorial days I have felt that the time was too short to provide a meaningful, especially hands on, presentation. The concept could of course be extended to other subjects such as learning TI BASIC, Extended BASIC, c99, Forth or a number of other languages on the computer. Members who would like to be involved in such weekend tutorials should write to the Directors of TIsHUG, PO Box 214, Redfern 2016 and advise of the type of course that you want to be involved in. If there are sufficient numbers of members for a subject then the chances of getting a weekend course together (probably not always a long weekend) are good.

A number of photographs were taken at the tutorial and these were shown around at the June meeting. These have been loaned to Rolf for possible use in the TND.

At the meeting on 1st June, two members asked when the club might arrange another beginner's assembly class. It is proposed to conduct this class on the October long weekend with Craig Sheehan teaching the group. This will be a hands on beginner's assembly class with each class member bring along their own expanded TI99/4A system. This new beginner's assembly group will be for a minimum of 6 members and maximum 15. If you are interested then book your place in the group now.

The classes will be held in separate rooms at the same venue which will allow pooling of catering arrangements. Anyone wishing to be included in either the continuation assembly class or the beginner's assembly class are asked to advise Ross Mudie as soon as possible to allow the necessary arrangements to be made.

To contact Ross Mudie use any of the following methods
1. TEXPAC BBS by mail to SYSOP.
2. Telephone at work (02)663 0141; leave a message on the answering machine if necessary.
3. Telephone at home (02)456 2122, during evenings 7pm to 9pm on week days. (Other times are OK on the weekends).

A weekend class is a very effective method of starting to program with assembly language. It does not have to always remain as a mystery!   o

For Sale

Secretary's Notebook

by Terry Phillips

I hope all who turned up at the July meeting had a good time. It was certainly a pleasant surprise to find a sunny Saturday for a change. Rolf tried his hardest with the Geneve demonstration, but unfortunately some problems prevented him from really putting it through its paces. I guess most of us would have second thoughts on going to the expense of importing a Geneve.

There are two new members to welcome this month, and they are:
Phil Hogan - Leura
Neil Williams - Farmborough Heights
Hope both of you enjoy your membership and that you can make it to some of the main meetings or the Regional Groups activities.

A further 6 renewals have been received, so with the new members included, our membership now stands at 205. Approximately 50 letters of reminder have gone to non renewing members, so with a bit of luck some of these may also renew. If you know someone who has not renewed, why not give them a gentle reminder? This past month has been rather quiet in regard to correspondence, so there is not a lot of news to report. This happens from time to time, particularly this time of year when the US Groups appear to go into a summer recess.

A former member, Jack Kenny (telephone (02)645 1278 has written advising of a list of hardware and software he has for sale. If interested in any of the following give Jack a call.
Console with 32K on board, Joysticks, Telephone Coupler (Modem), Extended BASIC, TI-Writer, 20 cassettes tapes of programs, 12 modules including Invoice Management, Personal Financial Aids, Programming Aids 1 and some programming books. Jack is after $400 for the lot but may be prepared to negotiate on individual items.

If anyone knows the current address of member Tony Imbruglia could they please get in contact with me as his TNDs are being returned marked "Left Address".

Next months meeting sounds like a good one with the Buy, Swap and Sell theme. This will be an excellent opportunity to maybe pick up a bargain or to offload those items you no longer need. Make sure you attend.

Please note I have not contributed a software article this month as no new disks have been received since my last contribution. Cheers for now and see you at the meeting. ○

Fairware Author

of the Month

by Geoff Trott

We, the users of the TI99/4A, rely on many people for our enjoyment of our computer, none more so than those who have written software which we use and rely on every time we use our computer. Some of this will be commercial software which we should have paid for and received value for our money in the form of a working program with good documentation, but the majority of software will be Fairware, which may not have cost anything and yet still provides a working program and good documentation. Software authors who produce good useful programs and release them for us all to enjoy under the fairware concept are the ones who are keeping us all going. If you look at the price of commercial programs for other computers which do the jobs that we are able to do with our fairware programs, you will find that $100 will not buy more than 1 program and you may well need $1000 to get a state of the art program. Fairware software costs the price of a disk initially but if we use the program the onus is on us to send a contribution to the authors to repay them for their

efforts and encourage them to continue development and perhaps write a new program as well. We can be sure that these authors are not relying on our contributions to live, as they do not ask enough and we do not send enough if anything at all.

TIsHUG directors have recognised that there is a problem here and that it is important that all users show their appreciation to these people who make our computing such a pleasure. One of the problems is that most software authors are overseas which means that there is the added complication and expense of currency conversion and postage to be overcome. Another problem may be not knowing the current address of the author. Our directors have decided to help with these problems by providing a clearing house for contributions to Fairware authors and to pay for the currency conversions and the postage. They also decided that TIsHUG will make a collection each month at the meetings for particular authors and their software. This means that the authors will receive a single substantial donation rather than many smaller ones. It also means that we can contribute as much as we think the software is worth to us or as much as we can afford at this time. So, while you are thinking about it, write down all the software that you use regularly and put a price next to each one to value its worth to you. Then put down next to that column what you have already sent to each author. Finally work out how you are going to contribute to those who are left.

TIsHUG now offers us alternatives to sending the money direct to the author. Of course sending the money direct to the author is the best way to get on an author's mailing list and to ask some pertinant questions about the software or about improvements which might be made in the next release. Another method is to contribute each month by mail or in person to the monthly fairware collection or to send in a contribution to be spread amongst several products and their authors. If you use this last method, be sure to send in a list of software and the amounts for each.

The Fairware software product for this month is DM1000, which comes from a user group, the Ottawa user group in Canada's national capital. Everyone with a disk system must have used this product at some time and I cannot remember my computer system without DM1000. Other similar programs have come along and in some respects replaced DM1000 but if you have ever used it then you should have made or now should make a contribution towards its development and possible evolution. Note that DM1000 is supplied with Funnelweb on the fairware principle. They are separate products and if you use both then you should contribute something for both products. Just get your contribution into TIsHUG as soon as you can and they will ensure that all of it reaches Ottawa. Do not forget the other authors of fairware software that you use and take up TIsHUG's offer of forwarding your contribution on to them. ○

SEND IN YOUR DONATIONS NOW!

Use a 150 ohm resistor shunt pack to terminate the data/control lines to your drive. Also, if selectable, set the Head Load option to be activated with the motor start up (HM). This will speed up disk I/O operations. Test the completed setup by formatting a disk and then writing and reading a one line BASIC program .

Finally, monitor the heat build up from the IC regulators and decide if you need to adjust the input voltages from the transformer, as described above, or whether additional heat sinking is required.

Next Month

In the August issue of the TND I will discuss hard disk power requirements. ○

Expanding XBs Powers
Writing Assembly Routines, part 3
by David Caron, USA

In my last article, I described four of Extended BASIC's assembly routines which are loaded into low memory with CALL INIT. (Actually these routines also exist in the MiniMemory and the Editor Assembler module but they are executed from different parts of CPU memory.) In this article I will present a more efficient way of printing letters to the screen from assembly using a loop.

Here it is:

```
       DEF  START      *This places the word "START" in
                        the DEF table along with its
                        start address.
VSBW   EQU  >2020       *The Extended BASIC assembly
                        environment has no reference
                        table, so the assembler
                        directive EQU (equate) must be
                        used to define the constants
                        VSBW, VMBW, VSBR and VMBR. Take
                        a look at pages 415 to 416 of
                        the Editor Assembler manual.
NEWREG BBS  32          *This is where our workspace will
                        be 32 bytes are reserved since
                        16 registers x 2 = 32.
STRING TEXT 'START'     *The TEXT directive places the
                        characters from the string in
                        the operand field directly in
                        memory in the form of their
                        ASCII values.
       EVEN             *Because START has an odd number
                        of bytes
START  LWPI NEWREG      *This instructs the computer to
                        use the memory locations
                        indicated by NEWREG as the
                        workspace registers.
       LI   R0,>0000    *Loads register zero with the 16
                        bit data 0
       LI   R2,STRING   *Loads R2 with the value of
                        STRING which is the first
                        address containing the series of
                        ASCII bytes making up the word
                        START.
LOOP   MOVB *R2+,R1     *The asterisk in this case does
                        not mean a comment is starting
                        but rather, now read carefully:
                        The byte at the address
                        indicated by R2 is copied into
                        R1, so the value of R1 becomes
                        83x256 (ASCII code of S, the
                        x256 means that 83 is the most
                        significant byte of R1, this is
                        how MOVB works). The + sign
                        after R2 means that the value in
                        R2 will be incremented by 1
                        after the MOVB operation has
                        been performed so the value in
                        R2 is now STRING+1. This
                        operation is what is known as
                        "Indirect Auto-Increment
                        Addressing". See page 58 of the
                        Editor Assembler manual.
       AI   R1,>6000    *adds hex 60 or decimal 96 to the
                        most significant byte of R1.
       BLWP @VSBW       *Sends the ASCII code out to the
                        screen.
       CI   R2,STRING+4 *Compares R2 with the address of
                        STRING+4 and sets the STATUS
                        bits to indicate less than,
                        equal to, and greater than
                        conditions
       JLE  LOOP        *If STATUS bits indicate a less
                        than (really a low) or equal
                        then go to LOOP.
       CLR  R0          *This code returns you back to
                        Extended BASIC
```

```
       MOVB R0,@>837C
       LWPI >83E0
       RT
```

And that is all there is to it! Each time the loop is repeated the next byte at the address in R2 is put in the most significant byte of R1, added by 96 for compatability with Extended BASIC and sent out to screen until R2 is > STRING+4, which means that we have sent all five characters of START to the screen and the loop exits.

Incidentally, you may have noticed that although we have declared VMBW VSBR and VMBR, we have not actually used them. Do not worry. We will be using them in the future and it is nice to already have them defined so that we can use them without looking up the calling address later on.

Why 96 must be added to all characters being sent to the screen

Take a look at page 403 in the Editor Assembler manual. Disregard everything except the Screen Image table and the Pattern Generator table. Notice that the Screen Image table starts at >0000, which is exactly what the above assembly routine takes for granted. Everything one sees on an Extended BASIC screen is in this table. The table contains 768 bytes as does the Extended BASIC screen (24x32 = 768). Each byte can be set up to represent any of the 256 characters which exist in the Pattern Generator Table.

The Pattern Generator Table contains 256 blocks of data where each block is one of the 256 characters and each block consists of 8 bytes which define the way the characters looks. (Think about CALL CHAR and how it works.) The total size of this table is therefore 8x256 = 2048.

Let us pretend for a minute that life is the way it appears on page 403. If, for example, the byte at VDP memory location 32 is set to 42, an asterisk would appear in the second row first column. Likewise setting all 768 bytes to 32 would be identical to a CALL CLEAR.

However, unlike the VDP RAM table on page 403 which is set up for Editor Assembler, the Pattern Generator table does not start at >0800 or >2048 but at >0000 or 0, the same address as the Screen Image table. Furthermore, the actual data for the Pattern Generator table has not been moved back 2048 bytes but 2048-800 = 7496. This means that the data for the Pattern Generator table starts at VDP address 768 and the Pattern table itself begins at 0. The programmers of Extended BASIC moved the Pattern table back so that they could use VDP memory starting at 2048 for Program space. The reason the actual data did not get moved back as far is due to the fact that the first 768 bytes would have been erased from the Screen Image Table. The net effect is that the Data for the Pattern Generator Table starts 768 bytes after the Table itself or 768/8 = 96 character blocks later.

This is essentially why 96 must be added to the character ASCII code, because the data for the character you want does not appear until 96 character blocks later.

If you still desire a more detailed explanation you will have to know all about VDP registers and how they relate to each other.

You are now probably wondering why Extended BASIC lets you access only 112 characters when there appear to be another 48 available (256-96-112=48). These 48 characters translate into 384 bytes of VDP memory (48x8) which is where sprite data is stored as well as Extended BASIC's variables. Therefore this memory cannot be used for character definitions.

In my next article I will introduce four more Extended BASIC Assembly utilities and explain how they can make the above assembly routine much more versatile.

Techo Time

with Lou Amadio

The July meeting at Woodstock was a short one for me as I had to leave early. The Two Way Expansion Interface described in the TND was displayed and at least two members are currently building one. The only part of this project which is difficult is obtain is the 60 way veroboard. I am currently trying to obtain some for the shop. Failing this, you could try to utilize narrower veroboard by overlapping and filling. Wire links passing through the boards and soldered on both sides will provide adequate strength.

The project for this month is a box and power supply for floppy drives. There was some interest in the disk drive boxes that were on display last month, but I may not have the time to produce any more. However, if there is sufficient interest, this is one project that I could use some help with. This would involve obtaining and marking up the aluminium sheet, as per the instructions below, ready for cutting and bending.

Console Repairs

We received a TI99/4A console from John Hagart of Gordanvale, North Queensland, for repair. From John's original letter, we thought the fault was associated with the VDP inductor, but it turned out to be a faulty 74LS138 chip. The console was dispatched back to John in good working order. John is having some difficulties with a light pen project described in the TND (June and October '86). Is there anyone able to help? If so, please write to

    J A Hagart
    8 Griffin St
    Gordanvale
    North Qld, 4865.

Still on the subject of repairs, Geoff managed to fix another 6 consoles which were left over from the tutorial day repair session.

The club now owns one of Geoff Trott's console testers. The unit is currently at Cyril Bohlsen's place. Cyril conducts a regular repair session where members are encouraged to get involved in the repair of their own console.

32K Memory Expansions

The club shop has just about run out of 6264 8Kx8 bit memory chips so we must soon look around for more. What is the demand for memory expansions? Please let me (Lou) know at one of the meetings.

Coming Soon

Next month I hope to be able to present an article on how to convert an IBM style keyboard for direct connection to the console. The information for the article came from Derek Wilkinson, of the Sutherland Regional Group, who developed the prototype.

I also plan to write an article on the power requirements of a hard disk system. I will be displaying a hard disk drive at the August meeting so that you will be able to see what goes on inside one of these very sophisticated pieces of computer hardware. o

Errata:
Direct I/O Interface
for PEB cards, by Lou Amadio

Please make the following corrections/additions to the Direct I/O Interface article which appeared in the July 1989 issue of the TIsHUG News Digest:

1) Cut tracks on both sides between the 60 way edge connectors on page 7, that is, we do not need pins 1 to 59 (odd numbers) of the bottom connector to go to pins 2 to 60 (even numbers) of the top connector. Do not cut these tracks if you decide to use the "Alternative Construction Method " described at the end of the article.

2) I have found that 6mm rubber feet beneath the bottom PEB card provides a better fit for the interface. You may have to modify the ones that you buy by grinding them to suit.

3) The spacing between the 44 way console connector and the back to back Veroboard may not be enough. Use 7mm rather than 5mm as indicated on page 7. This allows a bit more clearance for the 47 ohm resistors.

4) The 47 ohm resistors are mounted on the console side of the Veroboard directly behind the bottom 60 way connector. There is sufficient room either side of the 60 way connector to solder the common point for these resistors. If you are not using a speech synthesizer you can connect the common point to pin 1 of the 44 way connector. This means that the 7805 regulator and the two luF capacitors are not required.

5) Use direct wire links between the respective power rails (and similarly between the grounds) for the top and bottom 60 way connectors. This is important for the +9 volts unregulated rail as each card may draw up to 0.5 amperes and the Veroboard track may not be up to the task.

6) Transformer T1 is a Tandy 7015 not 7014 as indicated on the circuit diagram. You may also use Arlec transformers PT6672 and PT6978.

7) If you use the recommended spacing between the two 60 way connectors, do the final soldering between the connector pins and the Veroboard with the TI PEB cards plugged in. This will ensure the cards will fit as there is virtually no clearance.
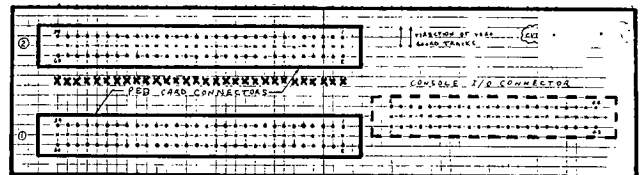
Power for Your Disk Drive
by Lou Amadio

Last month I described an inexpensive way to expand your console, using PEB cards, but without a PEB box. The most common peripheral device which is likely to be attached to an expanded system is a disk drive. Fortunately the cost of second hand floppy disk drives is now under the $100 mark and is therefore within the reach of most TI99/4A users. This month I will describe how to power and house these orphan drives.

Power Requirements

Floppy drives can be divided into two groups with regard to physical size and power requirements. Generally speaking, the full height drives (for example Shugart, MPI) require more power than the newer slim line drives which have a direct drive motor to spin the disk.

| Drive Type	12 Volt	5 Volt
Full Height	1.0 Amp	0.5 Amp
Half Height	0.5 Amp	0.5 Amp

The power supply described below is designed to power one full height drive or up to two low power half height drives. It is not designed to run a hard disk as this requires a much larger capacity on the 12 volt rail. Please note that if your drive uses a belt to spin the disk, then it is not likely to be a low power drive.

The supply electronics are mounted on a piece of Veroboard and housed either in a separate aluminium box or within the same case as the disk drive. I will be supplying a small number of preformed boxes to the shop to show those interested how simple they are to build.

The Box

As you can see from the diagrams, the design for a disk drive enclosure is very simple, involving only three right angle bends. It is the same for both a single full height or two half height drives. A single half height drive is housed in a smaller box and needs a separate enclosure for the power supply. If you buy a piece of scrap aluminium sheet (approximately 1 mm thick) and mark it out yourself, you should be able to get it cut and bent for a small fee at your local sheet metal works.

The larger box is designed to house both the disk drive(s) and the complete power supply. The drive itself supports both the base and the top cover using 4 screws for each piece as illustrated. Drill holes in the top and base to suit your drive(s) and also in the rear panel for the power cord and fuse holder. The holes for the Veroboard are best left until the board has been populated with the power supply components, particularly the IC power regulators and the chassis stand-offs. Four self adhesive rubber feet and a coat of paint will complete the job.
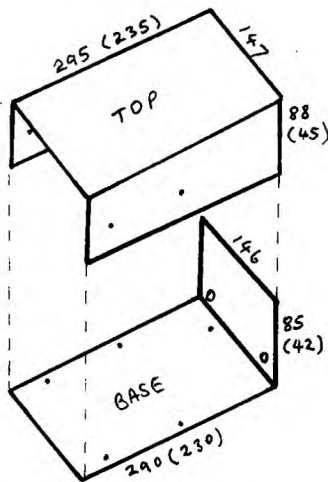

Measurements in brackets are for half height drives.

The Power Supply   see circuit diagram on page 16

The power supply uses a 15 volt, 2 Amp, multi-tap transformer, 4 diodes, 2 filter capacitors and two IC voltage regulators with associated bypass capacitors (4.7 uF, 25V). It is capable of supplying 1 Amp at both 5 and 12 volts simultaneously, which is enough for most situations. The 12 volt regulator is fed from a full wave bridge rectifier, while the 5 volt regulator is fed via a half wave rectifier and the centre tap of the transformer. The transformer specified (Arlec PT6978) has provision for both 15 volts and 12.6 volts with a centre tap for each. If, however, heat build up on the voltage regulators is a problem, you could try the 0-6.3-12.6 volt tappings in lieu of the 0-7.5-15 volts specified on the circuit diagram. If drive operation becomes unreliable, then you will have to revert to the higher voltages and use an additional heat sink for the regulators. This could simply be a "U" shaped channel made from scrap aluminium. Whatever method you choose will largely depend on the type of disk drive(s) that you are using.

All of the components (except for the power fuse) are mounted on a piece of Veroboard. The voltage regulators should be mounted first and located along one edge to support the board and to facilitate mounting to the aluminium chassis. Direct connection of the IC tabs to the case is possible as these are at earth potential. Use a little heat sink compound during final assembly to aid in heat dissipation

The edge of the Veroboard opposite the regulators will need 2 small chassis stand-offs to support the board. You could make your own, if necessary, by cutting the body of a plastic ball point to length. Use the diagram below as a guide.
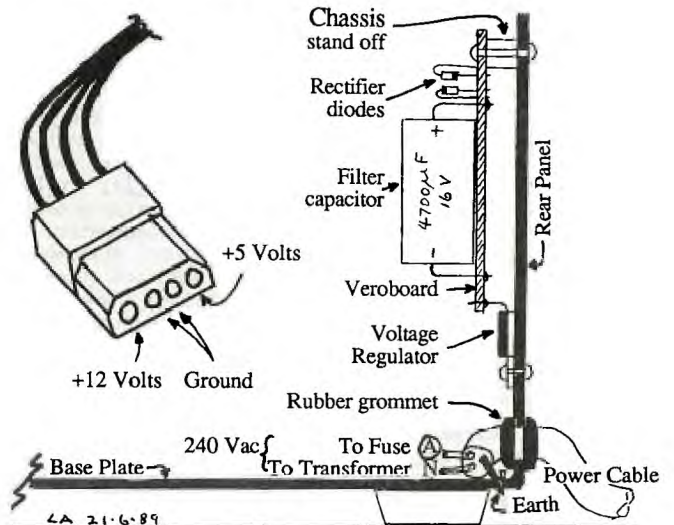
Details of plug and printed circut board for power supply

Mount the board on the rear aluminium panel as indicated. Locate the power transformer on the base plate and allow clearance for the disk drive, power cable and fuse. Use insulated multistrand copper wire rated for at least 3 Amps and keep point to point wiring short and neat.

The power cable passes through the rear panel via a rubber grommet and is clamped securely to the base plate. Solder the earth wire (green and yellow) to a metal lug and bolt to the metal chassis. The active 240 volt wire (brown) is connected via a 0.5 Amp fuse to the primary of the power transformer. The neutral wire (blue) is soldered to the other end of the primary winding. Insulate the 240 volt lugs on the transformer and the fuse holder with plastic adhesive tape. Note that there is no on-off switch as it is good practice to switch off at the wall outlet when you are not using your computer.

The output of the regulators is wired to a standard disk drive 4 pin female power plug (see diagram). Note the polarity of the power connections and use colour coded cable rated for at least 3 Amps. Use a separate ground return cable for each supply.

Double check all of your soldering and connections before applying power.

Testing

Before connecting the disk drive, test the output voltages of the regulators with a multimeter. If all is well, connect both power and data cables and test the system with your controller card. (You will need a length of 34-way ribbon cable and two 34-way solderless (IDC) card edge connectors for the control and data lines from the controller card to the drive.) If this is the only drive in your system, then set the jumper or switch to Drive 1.

The New TI99/4A

from **Asgard News**, typed by Larry Saunders

The TI-99/4A is ten years old this year, however, this is no excuse for having an "old" computer.

A ten year old person is merely a child. A ten year old dog is middle aged or even old. But a ten year old computer is ancient! The TI99/4A is ten years old this year. Ten years in the computer industry is the difference between vacuum tubes and transistors, between transistors and integrated circuits.

The TI99/4A computer system most of us know and love is actually more like 6 years old (circa 1983). It usually consists of a TI99/4A console (which supplanted the "4" in 1982), a Peripheral Expansion Box (from the beginning of 1983), a TI disk controller and SS/SD disk drive (also that time frame) a 32K card and an RS232 card. If you are perfectly happy with your computer, then perhaps you should skip this article for another. However, if occasionally you find yourself reading non-TI99/4A computer magazines longingly, ogling Macintoshes/STs/Amigas/PCs, and maybe even trying out a friend's, then perhaps you do want a new computer. If you have not purchased one of these other systems by now, then it is evident you still have not convinced yourself you "need" one. If that is the case, then perhaps all you really want is a "new computer": something different from the same old thing that will bring some of the fun back into computing again.

If all you have is the basic system outlined above, then you might want to first consider upgrading your disk system. A second drive, and more to the point, a new disk controller, can mean the difference between walking and running with the TI99/4A! A second drive will give you access to some programs that were difficult or impossible to use with a single drive; graphic packages, databases, word processors and more. A new disk controller (either the Myarc, AT or CorComp) will let you store more on a disk drive (twice as much in fact), and get it off the disk much faster (up to 4 to 5 times faster in some cases). This will make your old sluggish programs seems like they suddenly grew legs and learned how to run! The speed increase is particularly noticeable with disk intensive programs (of course), anything that reads and writes to the disk frequently. In fact, the speed improvement is so great you would think you have a new computer.

If you have a more expanded system with 360K drives, then maybe you just need a little something to bring some of the "zing" back into computing.

Do you look at an PC AT's keyboard enviously? If you do a lot of word processing or telecommunications, then the first thing you should consider is a Rave 99 keyboard. This little device, available with or without a keyboard (if you already have an IBM-style keyboard), will give your fingers new freedom. Not only is the keyboard more roomy, you can even define some keys to be certain oft-used commands or phrases. Available for $130 to $260, you will wonder how you ever managed without it.

Perhaps you feel working in 40 columns is for the birds. If that is the case, then your best bet is a Dijit or Mechatronics 80 column card. The Dijit cards plugs into your expansion box, while the Mechatronics device plugs into the side of the computer "boxcar" style. Both will give an excellent 80 column display, which is supported by a growing library of software, including an 80 column version of Funnelweb and the Telco terminal emulator. An 80 column expansion does not have to be an expensive proposition either. Both devices will function well with a cheap composite green or amber 80 column monitor, both of which provide clean, crisp 80 column displays of your online work or your latest novel or letter. The 80 column devices itself ranges in price from $146 to $300, which is much less than buying one of those dedicated word processors for $600 or more.

If you are tired of the memory shortage, and the keyboard and the 40 column display, then your best bet is definitely a Myarc Geneve 9640. Used as a TI99/4A you get an excellent keyboard with separate cursor keys, a built-in 192K RAMdisk (more on those later), an 80 column display with Telco, Funnelweb and the My-Word word processor that comes with it, and extra memory for those applications that can use it (My-Word gives you 56K of space for your documents). The Geneve will also give you a real speed boost; regular TI99/4A programs work up to 3 times faster. The Geneve is certainly cheaper than buying a Rave 99 keyboard, or for an 80 column display, and then another 192K RAMdisk. You are still ahead of the dedicated word processor, which are not fast or as capable.

In the software department, if you are into telecommunications, you will be interested in knowing that Telco, pretty much the best terminal emulator for the TI99/4A by most people's reckoning, really sings on the Geneve. Since most telecommunications service and bulletin board systems are oriented towards 80 columns, (not TIsHUG BBS), a Geneve or an 80 column upgrade for your TI99/4A is almost a necessity if you want to fully enjoy your on-line services. Additionally, Telco gives you many capabilities not found on $100 to $200 terminal emulators for IBM PC's, Macintoshes, et al, as well as by far some of the best documentation of any program for the TI99/4A, for a freeware price of only $20.

If you are into word processing on either a TI99/4A (with or without an 80 column display) or a Geneve, then you will probably want to get in line for PRESS (available soon for $59.95 US), which brings all the functionality of modern word processors to both machines, "what-you-see-is-what-you-get" writing, an unlimited document size, a 100,000 word spelling checker, and much more, also by the author of Telco. Press is a step above anything a dedicated word processor can provide, at 1/5 the price of a comparable PC or Macintosh program.

Purchasing an 80 column card and a Rave keyboard or a Geneve, or perhaps a new piece of software, can make word processing or telecommunications seem like a new experience. You will wonder how you ever did it before.

If you are tired of using Personal Record Keeper to keep your record collection straight, and occasionally eye purchasing a PC to better organize your life, you can save yourself a lot of money and aggravation with either a little equipment or software for your TI99/4A.

If you are serious about databases, you should have a disk system. The minimally configured disk system described in the beginning can take you a lot further than Personal Record Keeper, though. You might want to consider purchasing a modern database like TI-Base (which, at $25, makes $600 DBaseIV on a clone blush, and not just because of the price but because it can do almost as much). If learning a database language like TI-Base is not your bag, then perhaps you should consider one of the more "traditional" TI99/4A databases like PRBase (freeware) or First-Base ($59.95 US). First-Base is a quite fast, menu-driven database program with the ability to build sophisticated compound queries (like "list all the items in my inventory that cost more than $50 and more are in red", or something like that). The only disadvantage to First-Base seems to be the price, and some annoying bugs in printing out and sorting.

To really use TI-Base, First-Base or PRBase, you may want to move up to double sided, double density disks (360K). A new disk controller is a must, either the CorComp, Myarc or AT. If you have a large amount of data you wish to organize, you might want to buy more than just a regular DS/DD disk controller: the Myarc Hard and Floppy Disk Controller (or "HFDC"). The HFDC will let you put up to 720K on a single disk, controls up to 4 floppy drives of that capacity, and up to three 80Mb hard drives! At over $300 US for the card, and additional $500 or so to set up a 20Mb hard drive. it certainly is not a bargain, but considering you get the same capabilities only on a $1800 PC clone, it sounds a

lot cheaper.

The Myarc HFDC is a really nice piece of hardware that can enhance all of your programs, including telecommunications and word processing. With it you can have all of your programs, text files and data in one place at one time. You may literally never have to search for a disk again. Hard drives are also much faster than floppy drives; up to 10 times faster than with the flexible variety.

Spending $500 for a 20Mb hard drive on the TI99/4A may sound like a lot of money, but you have to spend usually that amount for one on a Macintosh, and up to $400 to $600 more for one of similar size on an Amiga or Atari ST. The HFDC will also let you have up to 240Mb of storage, while a PC, Amiga and ST can never come close (and 240Mb on a Macintosh would cost more than a Hyundai Excel). This is one instance where the "lowly" TI99/4A is more sophisticated and capable than its "more advanced" brethren.

A new database package, and if you can splurge, a hard drive (or at least higher capacity floppy drives), will make managing a database on your TI99/4A (or on a Geneve, which can do it 3 times faster), an entirely new change of pace.

If you are into spreadsheets, you will also benefit from using a Rave 99 keyboard with a separate numeric keypad that will make entering your figures a dream.

However, if you are not serious about spreadsheets there is only one real expansion avenue, purchase a Geneve and use the expanded version of Multiplan provided. The new version lets you have a much larger spreadsheet, in 80 columns, in a program that is much faster than on the TI99/4A. If spreadsheets are your life's work (if you are an accountant or something), then maybe it is worthwhile for you to buy a clone (Lotus 1-2-3 beats Multiplan hands down in many respects, and Microsoft Excel on the Macintosh beats everything else). However, if you do your taxes with a spreadsheet, manage a small business, or just putter around with the occasional table, then either a Rave keyboard or a Geneve will make the whole experience much more pleasurable than it was ever before.

If you are into publishing, either for a newsletter, for a school or organization, or even for a small business, then you have many possible avenues to improve the quality of your work. Publishing is usually differentiated from word processing in that the aim is usually to combine an illustration with text. If you are using TI-Artist and TI-Writer to do this, then you are to be commended for your tenacity. The alternatives to this method, fortunately, can be quite inexpensive.

If you use TI-Artist frequently you may want to purchase Font Writer II ($22.95), which to a limited degree will let you include pictures, fonts and borders in a TI-Writer document. If you can live with only one picture on a line and some patience while the program formats a document, then you can produce pretty nice looking documents. If you want a lot more precise control over your pictures, then you may want to look at Picasso (available in freeware and commercial versions), which will let you do half a page at a time, let you include a TI-Writer file, and paste TI-Artist pictures anywhere you want. The Picasso Utilities package ($9.95) will help a lot towards that end with utilities to use fancy fonts on your Picasso picture and print out the pictures perfectly for reproduction.

If you do not have a heavy investment in TI-Artist, you may want to look at The Printer's Apprentice, which is the only "true" desktop publishing programs for the TI99/4A. Like early desktop publishers on other computers, this is not "what-you-see-is-what-you-get", but more approximates a language. It is quite complex, not overly well documented, but can produce excellent output that would put a $200 program on a PC to shame. If you are willing to put in the time to learn it, at $50 or so with companion package (required for many things) you might save yourself thousands over going

with a Macintosh and a laser printer, and have document quality comparable to anything you can do on an average PC and dot matrix printer.

If you do not need pictures in your printed documents, you have many alternatives, including word processors and any of a number of freeware and public domain columnizing programs, or PRESS (when it is released), which allows you to mix columns of text to your satisfaction, as well as simple line graphics for tables and borders. (Press is a WYSIWYG word processor.)

The category of "desktop publishing" in the TI99/4A world sometimes includes specialized graphics programs. Some examples of these include Certificate 99 ($19.95), which allows you to make excellent signs and certificates, Calendar Maker 99 ($19.95), which will let you create complex picture calendars, a variety of banner programs, and any number of label programs which let you create labels with graphics and such. All of these programs are just as capable as anything on any other computer, usually cost only a tiny fraction of what they do elsewhere, and run on a basic TI99/4A system.

If you already have a graphics package (or several) you like to use, and just tired of waiting forever to finish up, or perhaps of flipping disks in and out of the disk drive, there are a number of ways to make your work go faster. One thing, of course, is higher capacity disks. Programs like Certificate 99, etc., are usually limited in graphics and fonts to those graphics you can cram onto one disk at a time. You can put 4 times as many pictures on a 360K disk as you can put on a 90K disk. Also, higher capacity disk controllers are usually 1 1/2 to 3 times faster than the TI Disk controller.

Another way to increase the speed is a hard-drive system as described above, which not only will let you use more pictures and fonts in many programs then you ever dreamed possible, but the programs will seem to just fly as the hard-disk reads and writes data 10 times faster than a floppy disk drive. Another way to get more "oomph" from your graphics programs is a Geneve, which will run them 3 times faster. A Geneve with 360K floppy drives can print a calendar from Calendar Maker 99, say, about 6 times faster than on a TI99/4A with 90K drives. Is $450 US too much to spend to make calendars faster? Well, the real question should be; do I want to spend $1000 to $1799 to buy a PC/Amiga/ST or $2000 Plus for a Macintosh to make the same calendars? If you answer to the first is "yes" then probably you are not interested in spending 2 to 3 times more still for another computer! If your interest is in games, if you have a basic disk system you really can play virtually every game for the TI99/4A. If you like arcade games and are tired of the TI99/4A's selection, then before you go out and buy a PC, ST or Amiga, I would go spend $200 on a Nintendo. The Nintendo's graphics are better than a PC's, and are on par with the best of the ST and Amiga. It is no wonder, the Nintendo uses the 9938 graphics chip found in the Geneve and the Dijit and Mechatronics 80 column cards.

If you like adventures, then you are more in luck. There is a large variety of adventure games for the TI99/4A ranging from text adventures like the Scott Adams and Infocom series to graphics adventures like Legends. If you like adventures and do not know what is available, Asgard Publishing has a 66 page fly sheet type book (called "The Adventure Reference Guide", I have a copy and if any one wants to have a look at it see me at the next Liverpool regional meeting), it lists over 200 adventure games for the TI99/4A (and over 100 available in the public domain). The neat thing about games is one good one (like Spad XIII) will really make you glad you have your TI99/4A and really glad you did not blow $1000 on a colour PC clone so you could play Pac-Man or some flight simulator that uses a thousand different key commands to fly the silly plane. The best thing about a good game, of course is that all you need is a basic TI99/4A system to enjoy almost all of them.

If you are love to program, either as a hobby or for school, then there is a whole plethora of software and hardware that will let you you write anything you want on your TI99/4A.

If you are into C at school, then c99 for the TI99/4A will generally interest you (even if it lacks *some full-C* capabilities like structures). The cost (it is freeware) is hard to beat also.

If you want to learn FORTRAN or Pascal, there are good to excellent versions of each available for the TI99/4A. FORTRAN 99 ($25 to 50) is an excellent FORTRAN IV-77 hybrid that is a pleasure to write in. Turbo Pasc'99 is a decent, if somewhat incomplete Pascal. There is also the P-Code system with UCSD Pascal, which is a much more complete derivation that lacks only speed and can be a real buy if you pick it up used.

There are also TI99/4A versions of many other popular and semi-popular languages, like Lisp (Artificial Intelligence), Pilot (education), Logo (ditto), Forth (for the HP calculator fiend) and a good selection of oddball languages. If you like good, old BASIC there are dozens of utilities to expand the use and function of Extended BASIC, including Super Extended BASIC and EZ-Keys Plus. There are also many programs to speed up the language (various assembly routines packages and such programs as Smash, Pre-Scan It! and Quick-run). Quick-run is a super fast starting program that will start even the slowest starting program you have in less than 5 seconds flat.

If you are into assembly you may want to look into the RAG MacroAssembler, or at least the Editor/Assembler module (if you are still using a MiniMemory). A program such as Explorer (a fancy debugger) will also really help out in this department. If writing a program in c99, FORTRAN 99, Assembly or Pascal there are utilities such as PrEditor (a programmer's editor), Batch-It (which lets you create batch files for compiling, assembling and linking code automatically), and many more that should be in your tool-box.

If your hardware is what has you down, a Rave 99 keyboard, or perhaps a 32K SuperSpace II from Databiotics ($60), will pick you up again. The SuperSpace includes a variety of programming tools, and is useful for assembly and compiled language programmers alike.

Do you use your computer for something else? Perhaps you use your computer for a little of everything mentioned above, but not enough of any one to justify buying a piece of hardware just to do that one thing faster. If this describes you, you are still in luck, though.

There are several pieces of hardware other than those mentioned (the Rave keyboard, higher capacity floppy disks a hard drive and the Geneve), which will make using your computer even easier than before.

One such item is a RAMdisk. A RAMdisk is a card full of memory chips that pretends it is a floppy disk drive The advantage to such a device is that it works about 10 times faster than a regular floppy disk. You can boot up your word processor in 2 seconds, or go from one part of a drawing program to another seemingly instantly. RAMdisks range in size from 90K to 1.5 Mbyte. Some are battery backed so they do not lose their contents when the computer is shut off, and some are not. Four or five manufacturers produce them, and they range in price from just over $100 US to over $800. Even a little 90K RAMdisk at the bottom of the range could "change your life", or at least the way you run every program you have.

Another such item is a GRAM device. A GRAM device (of which there are several types under various brand names), is a little device full of memory that pretends to be a cartridge (it also stops lockups that are common with Super Extended BASIC cartridge) The advantage to such things is that you put up to 4 to 5 cartridges in it at the same time in some of the larger ones. No longer will you have to hunt up your Extended BASIC cartridge again. It will always be right there on the main menu of your computer when you start-up. A really popular GRAM device is the P-Gram card, which has up to 72K of memory and plugs into your expansion box. Amaze your friends by running Extended BASIC or TI-Writer with no module in your cartridge port! It take only 40 seconds to vacuum the largest of cartridges to disk and the same to load into the P-Gram card. You can modify any cartridge (for example, Extended BASIC to Super Extended BASIC Editor Assembler, to Super Space, RS232 to PIO, etc.

Finally, if your problem is just plain speed, not the speed that the computer reads and writes data, or the speed in which you can select modules, or type, then you really only have 1 solution other than the Geneve (which at $450 US gives you a 3 times speed increase). The other solution is considerably less expensive ($100 or less), but entails invalidating your warranty, and finding someone technical to do it for you. It is called a "16 bit modification", and in technical terms means putting the 32K memory expansion inside your console on the 16 bit bus. Combine this with a faster timing crystal, and you can get a 25 to 35% speed improvement in your console. The only disadvantage is that some things do not work at a different speed (terminal emulators, for one thing). Which means you will have to disable the speed increase to run certain things. Another disadvantage is that unless you are a technical wizard who can juggle soldering irons, you will probably have to find someone who is. TIsHUG user group have some like that. If you really want this type of modification, be prepared to beg and/or name your firstborn after him/her. Some will do it for money, however (computer "habits" can be as bad as any drug addiction).

All in all, there is no reason you should be using a 6 year old computer. With any combination of the devices and software mentioned above you can make your TI99/4A compete with whatever any other computer can offer, often at a fraction of the price. You will not have the latest thing on the block, but there is quite a bit of comfort not being on the "bleeding edge", as it were. In fact, there is so much available for expanding your TI99/4A, it might as well be "The New TI99/4A". o

TIdbits, Mid South 99ers, Feb '89: Horizon RAMdisk now available with HM628128 static RAMs and will accommodate 1.5Mb without stacking chips. Also included is a switch to disable the Horizon RAMdisk in case of lockup. Files are preserved and the ROS may be reloaded. Review of the DKM BASIC Compiler, a list of over 90 orphaned computers, advice on cleaning keyboard key switches, useful list of TI99/4A product suppliers. March '89: TI-Base tips, features of the new 9640 FORTRAN, reviews of Calendar Maker 99 and Browse (text file utility) and the availability of MICROdex library files for TI-Base. April '89: Myarc's hard disk system reviewed, computer virus scam, miscellaneous club and program information. May '89: Release of V4.2 TI-Writer - a full featured word processor with many enhancements over V1.0, 9640 FORTRAN memory usage. June '89: Star NX1000 printers with ROM versions greater than 1.31 may not work with the TI99/4A, new programs include Page Pro 99, Page Pro Pics and Music Pro. Music Pro allows you to type notes on a staff and is described as a "word processor for music", a windowing program called cSHELL99 allows loading of programs (including c) and returns to the DOS shell, review of TI-sort, hard disk user tips, TI Forth for MDOS, limitations of the TEII module, overview of the Genial Traveler by Bill Gaskill.

LeHigh 99er, Winter '89: Review of Macflix and tutor, Tips From The Tigercub, a flat battery in your Minimem will actually stop it from working, slashed zero in TI-Writer .TL 48:48,8,47. Spring '89: review of XB:BUG which is an Extended BASIC debugging tool, TI-Base v2.0 help, attention grabber, Maze maker and Easy Grader programs in Extended BASIC, TI Artist for Beginners and TI-Writer "Control U" Commands. o

Newsletter Roundup

with Lou Amadio

Local Newsletters

TIUP TITBITS, May 89: Mention that Jim Peterson is attempting to catalogue the public domain software for the TI99/4A Community, flow diagram for the "Telephone Transfer of Disk" (Mass Transfer), an Extended BASIC Maze Maker program , a hardware project on how to convert the v2.2 consoles to V1.0 to enable use of non-TI modules and Extended BASIC programs to create "readable technical jargon", produce banners, and a label maker.

Melbourne Times, March '89: Add for the up and coming TI99/4A Fair: October 14th, 2pm at Deepdene Park, Whitehorse Rd, Deepdene, article by Graig Miller on the power of AND, how to transfer long BASIC tape programs to disk, review of TIBase V2.0, Peter Gleed continues his series of Multiplan Tutorials; Costing for a Profit

Hunter Valley 99ers, April '89: Instantaneous reloading of the ROS from module, problems with different combinations of GPL and MDOS for the Geneve, problems with European versions of TI-Writer, PEB card for prototyping, Joe Wright on MiniMemory programming, Forth SIG group activities. May '89: Calling for orders of QED Module boards (Upgradable to 64K) and the RD200 half Meg RAMdisk (Ph 049 486509), idea from Bob Carmony on using velcro strips to hold peripherals together, use of "ON ERROR" in Extended BASIC programs, report on the success of the local Microcomputer Exhibition which allowed the club to wave the flag for the TI99/4A, suggestion of a "Best of" newsletter from each group for the TI99/4A Fair in Melbourne this October, release of "Program Writer" fairware from Bob Carmony, update of geneology record keeping program from Joe Wright, Funnelweb tip: CTRL[;] will change lowercase to upper and vice versa for CTRL[.] and L before PIO will print out the line numbers with the text, article on Programs that Write Programs by Bob Carmony, Tony McGovern reports on bugs encountered with CorComp and AT disk controllers when using Funnelweb SD and DM1000, upgraded QD for the AVPC 80 column card, and concern at heat dissipation with Myarc PEB cards (an autotransformer would cure the problem) and lastly "Programming in Forth".

Bug Bytes from Brisbane, May '89: Notice of the release of c99 by Clint Pulley for the Geneve, a hard disk backup program in assembler to copy one half of the disk to the other half from Garry Christensen, Programming in sound in assembler, a good article (by Geoff Trott, ahem) on how to wire up the pio interface cable to overcome any incompatible problems with printers, reviews of a lotto program and of Disk Utilities.

Overseas Newsletters

Northern NJ 99er's, May '89: In defence of Extended BASIC by Art Byers, modifications to Horizon RAMdisks to provide RESET on power up and a disable switch to "hide" the Horizon RAMdisk from the rest of the system, TI-Writer tip: use RS followed by / / @/ for example to assist with bold facing sentences (answer Y to prompt). June '89: experiments with a text scanner (Omni-Reader) received the thumbs down, using BASIC with the CALLs in PRK and an article on "Disk Fix"ing.

TI-SIG (San Diego) April '89: "Death by Cleaning " where a monitor caught fire when cleaned with a liquid containing propyl alcohol, two Extended BASIC programs - one to display large numbers, the other on trick display with a CorComp disk controller and finally an article by Tony McGovern on "DSR-Links and the RS232".

Oakland Computer Club (founded 18 months ago) June '89: Reports on various club activities.

Rocky Mountain 99ers, May '89: Very useful article to assist in debugging console and module problems by John Guion and "Word Processing BASICs".

Sacramento 99er Users Group, Mar/Apr '89: FORTRAN V4.0 available for the 9640 offers many advantages over the TI99/4A version.

TI Focus May '89: An enhanced Extended BASIC (public domain) from Alexander Hulpke (Germany) with many new CALLs (needs 9938 VDP), review of Typewriter 99 by Tom Arnold (highly recommended), notes on Maze of Grog, review of Mechatronic 80 Column card, how to install an "enable" led on the Axiom printer interface, TI-Base tips - command file to delete all records.

PUG Peripheral, April '89: Bud Mills advises that Horizon RAMdisks will be available with 128Kx8 bit static RAMs, release of MX-DOS 3.0 for the TI99/4A which allows you to view or print text files, run or delete and use a graphical interface, installing true lower case in Forth, transferring PRBase files to TI-Base, RESET and DISABLE switches for the Horizon RAMdisk. May '89: "Hot Bug" fairware program from Charles Earle features a pop-up design complete with hexadecimal calculator, article on "High Res Graphics and the 99/4A", aligning disk drives, "Colister" program printer in Extended BASIC (10 lines) and two pages of "The Software Biggies" by Jack Sughrue.

Tacoma Informer, May '89: Myarc's hard disk system - setting up, formatting, accessing subdirectories and some history of things that might have happened to the TI99/4A.

TopIcs, LA 99ers, April '89: "The Power of AND " by Graig Miller, TI-Writer editor and formatter "Cue cards", cleaning printer heads, "Floating Point Arithmetic" in Forth, Extended BASIC word counter for DV80 files, number of useful tips for TI-Writer and an article on batch files. May '89: A new version of TI-Writer (V4.2) offers significant advantages, including speed, over the original TI version, Legends II almost ready for release, Pro Page 99 - lets you compose a full page of text and graphics, as well as column printing of text, if sprites are not used then CALL INIT::CALL LOAD(-31878,0) (?? original address not correct) will increase execution speed considerably, short program from Jim Peterson which clears and resets all, hints on sorting numerical strings with PRBase, how to print the "degree" sign with TI-Writer, "Faster Graphics" in Forth, and The Printers Apprentice tutorial by Ken Gilliland.

ROM Orange County Newsletter, April '89: Word lister program keeps track of words, converting printer codes to screen patterns, file handling and printing graph paper in BASIC.

Ottawa TI99/4A Users, May '89: Loading assembler programs from cassette, calling subroutines in assembler, review of EZ-Keys which gives a number of very useful function key utilities for Extended BASIC including disk catalogues, screen dumps and macros, Teaching PRBase step by step tutorial, an Extended BASIC shell sort routine and TI BASIC key codes.

Spirit of 99, Ohio 99ers, May '89: TI-Base Tutorial 4 by Martin Smoley - a must for database users, a comprehensive article on TI99/4A software, where to find it and how much it costs, more on cassette file management, configuring Funnelweb 4.1, text to speech in Extended BASIC and finally a tip for belt driven floppies - if the drive runs slow (regular formatting errors) try reversing the belt. June '89: Cassette files part 7, very useful TI99/4A Adventure Compendium, review of Dijit 80 column card, Jack Sughrue advises on "How to organise your life" (and have more time for your TI99/4A), optional 32K modification for the Horizon RAMdisk to replace the TI 32K card, TI-Base tutorial part 5

CIM 99, Montreal, May '89: Review of "The Mine" game. June '89: Review of Pinball.

TI-Base Tutorial

by Martin Smoley, NorthCoast 99'ers

Copyright 1988 by Martin A. Smoley

I am reserving the copyright on this material, but I will allow the copying of this material by anyone under the following conditions. (1) It must be copied in its entirety with no changes. (2) If it is retyped, credit must be given to myself and the NorthCoast 99ers, as above. (3) The last major condition is that there may not be any profit directly involved in the copying or transfer of this material. In other words, Clubs can use it in their newsletters and you can give a copy to your friend as long as it is free.

The last article I wrote on TI-Base was a review in the July/August newsletter. In that article I told of many problems I had with the PRINT command and other functions of TI-Base. I also said that I thought these problems would be corrected, and many improvements would be made. I would like to say that the second of those two statements is now the most important. I received (via Deanna Sheridan) a copy of TI-Base version 1.02 and a four page letter from Dennis D. Faherty, its author. In the letter he related to 10 previous errors that had been corrected (one of which was the PRINT error) and to a multitude of improvements and refinements he wanted to make on TI-Base. This information has made me ecstatically happy. I feel that TI-Base will become as popular as TI-Artist and at some point will be so popular that you will be able to get Command file routines from your club library just as you can now get Multiplan Screens or Extended BASIC programs. TI-Base is a great enhancement to the TI99/4A.

And now the Tutorial folks. First some house keeping. Marty's Theory: will signify the beginning of some text which should not be taken as fact, but as my interpretation of an item. <E> means press <ENTER>. (Further Explanation Later) means further explanation will follow and last for now is ">", the greater than sign. I will use ">" when program segments are displayed at the left of every line. The position immediately to the right of the ">" will be column one. Take the example >12345. You should think of the number 1 as column one. The > does not exist. It is for reference only, the same as when you type in an Extended BASIC program, at the head of each line you see > but it is not part of the program.

Let us get started. The first thing you do is make backups or copies of the original TI-Base disks and put the originals away in a safe place. If the originals arrived without the write protect slots on the disks being covered, do that first, then make your copies. The program will read and write to all of the disks used in the database process so you cannot write protect them. This means that you should not use original disks and you should make copies of everything at the end of every work session. Backing up does not matter a lot at this point, but if you lose a data base with three or four hundred names in it, and you do not have another copy, you are in for some agonizing re-appraisal.

Having stashed the originals put your copy of the TI-Base system disk in Drive 1 and a newly initialized SS/SD disk in Drive 2. Then select Extended BASIC and TI-Base will auto load. It takes a couple of minutes so be patient. After loading, TI-Base will ask for the date. This will be MM/DD/YY or Month, Day, Year. Enter the date, and use zeros, it is a good procedure. For example, July 9, 1988 would be 07/09/88. TI-Base will then save the date and DO the program called SETUP. For Your Information: in this system DO rep: the Extended BASIC RUN (more or less). When   ' is executed you will be left with a bunch of junk on the screen and a dot "." at the bottom left corner of the screen with the cursor flashing next to it (see SCREEN ONE). For Your Information: I will at least partially

explain any new item we encounter as they occur. I will also try to proceed "Top-Down" in programming and explanation.

```
>001  *       Welcome to TI-BASE
>002  *   QUIT will terminate TI_BASE
>003  *
>004  SET DATDISK=DSK2.
>005  DISPLAY STATUS
>---------------------------------------
>DATDISK = DSK2.      Database files on DSK2.
>PRGDISK = DSK1.
TI-Base
  System Disk = DSK1.
>PRINTER = PIO.       Printer port PIO/RS232 etc.
>LINE    = 080        Printer page width (Columns)
>PAGE    = 056        Printer page length (Lines)
>HEADING = ON         Print all headings
>TALK    = ON         Echo commands to the screen
>SPACE   = 01         Space between fields
>RECNUM  = ON         Show record numbers
>LSPACE  = 0256       Space available for LOCALS
>DATE    = 07/09/88   This is the Date you Entered
>---------------------------------------
>006  *      FUNCTION (7) for help.
>007  RETURN
>.                [ SCREEN ONE ]
```

All of the lines with line numbers (001-007) are part of the command file called SETUP. The lines without numbers are part of the STATUS display. Lines 1, 2, 3, and 6 are comment lines and are made comment lines by placing an asterisk "*" in column one of any line. Important: line 2 could be misleading. QUIT does not refer to FCTN[=] (Quit) in any form. You must never force the machine to quit or reset before you leave TI-Base by the proper procedure. Line 2 means type QUIT at the dot prompt and press <ENTER>. TI-Base will then take care of its house keeping (close all files, etc.) and exit to the TI99/4A system. Lines 4 and 5 are actual commands which can be included in a command file or typed in at the Dot Prompt. For example type the following exactly at the Dot Prompt. You will notice that the word CLEAR, cleared the screen and DISPLAY STATUS brought back the stuff between the dashed lines.

```
>SET DATDISK=DSK1. <E>
>CLEAR <E>
>DISPLAY STATUS <E>
```

You should also see that DATDISK now equals DSK1 (if all went well). If it did not work, type it in again and be careful of spaces etc. When you have made it that far type the following. This should reproduce the original SCREEN ONE.

```
>DO SETUP <E>
```

The RETURN in line 7 returns the system to the level prior to this program section. You typed DO SETUP from the Dot Prompt so when the RETURN is encountered we are returned to the Dot Prompt. If we executed SETUP from another command file, when we hit the RETURN the program would have gone back to the file that called it (Further Explanation Later). Let us do some house keeping. Type in the following.

```
>COPY DSK1.SETUP/C DSK2.SETUP/C  <E>
```

When you see the message "ready devices, press ENTER", just press <ENTER>. The command you have just entered will then go to drive one and run a subprogram of TI-Base to perform the COPY function. That subprogram will then COPY the command file named SETUP/C from drive 1 to drive 2. The first DSKx designates "FROM" and the second DSKx designates "TO" a drive number. The first name "SETUP/C" is the complete name of the setup command file and must be used in this instance. You recall that when a DO SETUP command is issued the /C is not included in the name (Further Explanation Later). The second name, or the name you are copying to, can be any name you wish (up to 10 letters) (Further Explanation Later). For Your Information: We have copied setup to drive 2 because if you type DO SETUP at any time TI-Base will look for it

there (try it and see). You should get a feel for what is on which disk as we go along. "OK, let us CREATE a database." Type in the command lines as you see them below. When you type CREATE TNAMES and press ENTER, you will immediately see [SCREEN TWO].

```
>CLEAR <E>
>CREATE TNAMES <E>
```

arrows to move, <ENTER> to advance

```
FIELD  DESCRIPTOR  TYPE  WIDTH  DEC

  1     _____   _    ___    _
```

[SCREEN TWO]

This is the screen in which you tell TI–Base the size and shape of the database you would like it to create for you. This is actually called the structure of the database, and that is why the command DISPLAY STRUCTURE will give you a screen like this one, but with all the pertinent information filled in. Note: a database must be in use at the time. The Descriptor is the name you will call a particular item, such as Last-Name, First-Name, Middle-Initial, etc. Marty's Theory: If you can keep these names short, like LN for Last-Name, or MI for Middle-Initial, later on when you are using those names to perform different tasks you will not have as much typing, and you will be able to get more on each line, plus (memory space is tight) (Further Explanation Later). The Type is a one character entry, either N, C, or D. N stands for Numerical, C is Character, and D means Date. Marty's Theory: Make all your fields C for Character unless you plan on performing a mathematical function on it. For example, the zipcode is all numbers but it should still be designated C for Character. The Date designation is used when you want the computer to enter a date for you, or when you are going to enter a date in the form MM/DD/YY. I do not want to go into this theory so early in the tutorial. Instead let us get going on TNAMES.

I have created a database call TNAMES using the information displayed in [SCREEN THREE]. Type in the data exactly as you see it so we can move along.

arrows to move, <ENTER> to advance

FIELD	DESCRIPTOR	TYPE	WIDTH	DEC
1	LN	C	15	
2	FN	C	15	
3	MI	C	2	
4	SA	C	25	
5	CT	C	20	
6	ST	C	2	
7	ZP	C	5	
8	PH	C	12	
9	XP	C	5	
10	GP	C	5	
11	ID	N	7	0

[SCREEN THREE]

When you are entering information these keys are active.

FCTN[1]= Del. Char.	Delete one character	
FCTN[2]= Ins. Char.	Insert one character	
FCTN[3]= Del. Line	Delete complete line	
FCTN[4]= Ins. Line	Insert a complete line	
FCTN[5] Not Used		
FCTN[6] Not Used		
FCTN[7]= AID	Brings up the help screens	
FCTN[8]= Save/End	Saves the STRUCTURE	
FCTN[9]= Escape	Discards the STRUCTURE	
ENTER = Next Col.	Moves to the next column	
Arrow Up Active	Moves to previous line	
Arrow Left Active	Moves <= one Char./Column	
Arrow Right Active	Moves => one character only	
Arrow Down Active	Moves down one line	

If you are apprehensive, type CREATE XP <E>. When the screen comes up type in all kinds of junk. Arrow up, down and backwards. When you see how it works press FCTN[9]. All your garbage will be thrown away and you can start in on TNAMES. While you are entering the information for TNAMES as in screen three the only place there may be a question might be in field 11. When you get to the TYPE column, enter N and press <ENTER>. At that point the cursor will jump to the WIDTH column and the DEC or DECIMAL column will be highlighted. This only happens when you designate N for numbers. You then type 7 in the width column and when you press <ENTER> the cursor will jump to the DEC column. You now enter the number of decimal places you desire. If you were planning on dollars and cents, you might use 2 as the number of places. We are using a whole number so enter a 0 for no decimal places. When you have entered field 11 press FCTN[8] and TI–Base will create TNAMES for you and ask if you would like to enter some data at this time. If you answer N for no, you will be kicked back to the Dot Prompt. If you have the stamina at this point, answer Y for yes and enter the data from my printout [SCREEN FOUR] at the top of page three of this tutorial. I have entered four fictitious names, and my own, in TNAMES. I will use this data in future tutorials.

([SCREEN FOUR] is not produced here as it requires special printing. If you follow the tutorial carefully, you should be able to construct it with your own printer. ED)

Having entered Y/es to enter data after the last screen, you should be in the APPEND mode, and you should see [SCREEN FIVE].

```
              APPEND

  LN    _____      000
  FN    _____
  MI    __
  SA    _____>
  CT    _____
  ST    __
  ZP    _____
  PH    _____
  XP    _____
  GP    _____
  ID    _____
```

[SCREEN FIVE]

While entering data the previously described key functions are in effect. When you finish typing in the Last-Name (LN) pressing <ENTER> will move you to the next field. You will notice that the numbers that run up at the far right of each line are actually keeping track of your character position. The ">" at the end of line SA is telling you that there are more spaces for characters past the highlighted area (in this case only one space). As you enter data and reach the end of the ID field, when you press <ENTER> a new blank screen will come up. At that point the cursor will once again be in the first position to start entering another last name. If you are on the last data to be entered and at the end of the last field, do not press <ENTER>. At that point you should press FCTN[8] to SAVE/QUIT. This does save, but it does not really quit, and you will have to press FCTN[9] to get back to the Dot Prompt. If you were worn out back when the question of entering data originally came up, you answered no and got out of the system. You can now get back in by typing the lines below.

```
>CLEAR <E>
>USE TNAMES <E>
>APPEND <E>
------------------------
>CLEAR <E>
>USE TNAMES <E>
>EDIT <E>
```

The CLEAR is not really necessary in this case but helps me see any new screen messages without the extra clutter. Note: the EDIT is only usable when you already

have data in the data base. I hope I have not been too confusing and you have been able to create the database and enter the data in screen four. If not, re-read this tutorial and consult your TI-Base manual. I would like you to have a small database and be able to do something with it by the end of this tutorial.

Something I have not covered adequately up to this point is the phrase CLOSE ALL, and what is happening at the bottom of your screen in the highlighted area. I previously stressed the point that you must type the word QUIT at the Dot Prompt in order to leave TI-Base. Doing so would cause TI-Base to look for and close any open databases before it quits to the TI99/4A system. When you are working with one database, and you would like to use another database you type CLOSE <E> at the Dot Prompt. If you are working with several databases and wish to do something else, you type CLOSE ALL <E>. The highlighted area at the bottom of the screen will give you information on files that are open. This is particularly helpful when your screen is blank and the cursor is sitting at the Dot Prompt. This information will consist of the name of a database which is currently open, and selected (Further Explanation Later), the record number which TI-Base is currently pointing at, and it will flash current system operations in the far right hand corner (Further Explanation Later). My point is that if you see a name and some record numbers at the bottom of the screen, you should type CLOSE ALL <E>, before starting any new major tasks. Assuming that you have managed to create the database named TNAMES and have typed in the information shown in screen four, I would like to run through a couple things that should be enlightening. Type in the items below as usual. The system will give you messages as the data is being sorted, etc. Read the messages and observe the printout.

```
>CLEAR <E>
>CLOSE ALL <E>
>USE TNAMES <E>
>SORT ON FN <E>
>PRINT ALL FN,MI,LN
```

```
>SORT ON LN <E>
>PRINT ALL LN,FN,MI
```

```
>SORT ON ZP <E>
>PRINT ALL FN,MI,LN,ZP
```

```
>SORT OFF <E>
>PRINT ALL FN,MI,LN,ZP
```

```
>SORT ON XP <E>
>PRINT ALL FN,MI,LN,XP
>CLOSE ALL <E>
```

I am attempting to show the unbelievable flexibility of this program. Merely by typing in a few lines of text at the Dot Prompt you can sort the data on a different field, and print out only the fields you want, in the order you want. At this point you probably get confused by the different nature of this programming language. When you have used it for a while you will think it is the greatest record keeping system to come out for the TI99/4A, bar none. With the use of the APPEND mode you can add as many new records as you wish, and with the EDIT mode you can correct or change any information in the database. For Your Information: before moving on I want to fill you in on screen four. In order to get that printout, I previously set my printer to condensed print. I then entered SET LINE=134 at the Dot Prompt: 134 was the only length that worked properly (I tried several). Then I typed USE TNAMES <E> and PRINT ALL <E>. I do not know where the end characters in each line came from.

Now it gets interesting. We are going to create a small program, or create a COMMAND FILE. However, create is not the right terminology. The phrase is MODIFY COMMAND (filename) <E>. Filename is any name you would like to call the command file. It should be eight characters or less in length, and do not add any of the identifiers you may have picked up along the way (/C). Just type everything following exactly as you see it.

Take your time typing and allow time for the computer to do its job each time you press <ENTER>.

```
>CLEAR <E>
>CLOSE ALL <E>
>MODIFY COMMAND LBLS1 <E>
```

```
>* Command file LBLS1    "LABEL Prog."
>*
>SET TALK OFF
>SET RECNUM OFF
>SET HEADING OFF
>SET LINE=80
>CLEAR
>LOCAL TEMP C 40
>LOCAL BLNK C 1
>USE TNAMES
>SORT ON ZP
>TOP
> WHILE .NOT. (EOF)
>    REPLACE TEMP WITH "            ";
>      | " Exp. Date " | XP
>      PRINT TEMP
>      PRINT BLNK
>      REPLACE TEMP WITH TRIM(FN) | " ";
>            | MI | " " | LN
>      PRINT TEMP
>      PRINT SA
>      REPLACE TEMP WITH TRIM(CT) | ", ";
>            | ST | " " | ZP
>      PRINT TEMP
>      PRINT BLNK
>      MOVE
> ENDWHILE
>CLOSE ALL
>SET TALK ON
>SET RECNUM ON
>SET HEADING ON
>RETURN
```

```
>FCTN (8)        This will save the command file.
>DO LBLS1  <E>         This will run the file.
```

The information starting with CLEAR and ending with DO LBLS1 is everything you must type in to create and run a small program that will produce mailing labels from the database named TNAMES. It is that easy, and yet it is quite complicated. I will take the last half page of this article to give you some idea what is going on. The rest must wait until the next tutorial. I hope that what you have done so far has run successfully and your mind has not turned to mush.

The line MODIFY COMMAND LBLS1 <E> is the line that invokes TI-Base's Editor. This establishes that a command file is being created and will (if successful) be saved to the DATDISK under the name LBLS1. At the time the file is saved the identifier /C will be attached to the name LBLS1 to produce LBLS1/C. This is why you cannot use 10 characters in the file name. Once you are in the editor the previously described keys are active (FCTN[1], FCTN[2], FCTN[3], Arrows, etc.). Lines that start with an asterisk "*" are comment lines. For Your Information: do not use more than a couple of comments, they eat up memory (Further Explanation Later). All of the lines that SET something OFF are house keeping. LOCAL TEMP C 40 initializes the variable named TEMP. TEMP will hold up to 40 characters (C). The variable BLNK can hold 1 character (C). At this point both variables are initialized blank or empty. We will refill and/or use them later. In the next three lines we are telling TI-Base to USE TNAMES and SORT that database ON the zipcode field (ZP). When it is done we want it to go to the TOP, or beginning of the database.

The next part of the program is a chunk. The chunk I refer to is everything from WHILE to ENDWHILE inclusive. This is the part of our program that does most of the work. When our program executes the word WHILE it does the whole line. This actually says to TI-Base , WHILE you do .NOT. hit the (EOF), or End Of

Programming c99

part 1
by Craig Sheehan

Many members have expressed the sentiments that BASIC is too slow, and assembly language is too time consuming both to learn and write. An easier way to obtain the speed of assembly and the ease of a language like BASIC is to use a compiled language. With a compiled language, like 'c99', you write your program using an editor, like Funlwriter, and then pass the text through a program called a compiler, that produces a machine language output.

This series of articles will concentrate on one such compiled language, 'c99', and will assume no prior knowledge. 'C' is a universal language used widely on all computers from home machines to large mainframes. Being so widely used, it offers the advantage that provided you have a 'C' compiler for a new computer, you can transport all your programs to this new computer.

Before you start using 'C', you must have a compiler. In the case of the TI99/4A, a compiler that supports a subset of the language has been written by Clint Pulley, and is available as a fairware program from the club shop as a flippy for the usual media fee.

We shall now launch into our first program, which like all first programs, simply prints a message on the screen. Whilst it does nothing useful, it serves as good exercise on using the compiler and showing the basic syntax of a 'C' program. The program is shown in Figure 1.

```
---------------------------------------------
/* First 'c' program */

extern printf();

main()
 { printf("Your first 'C' program\n");
   exit(0);
 }

         Figure 1 - Your first 'C' program
---------------------------------------------
```

The program may seem simple, which it is, but you may think at first that actually getting into a runable program is a little complicated. However, after doing it a few times, it will become second nature.

You will need two disks: one with the 'C' compiler (available from the club shop), and the other a blank initialised disk. From side A of the 'C' disk, copy the following files onto the blank disk: CFIO, CSUP, PRINTF and STDIO. Whilst we will not use all these files in this part, it will be handy, and save a lot of disk swapping, in later parts.

Now type in the program. This done using the editor of Funlwriter's Editor Assembler mode (option 1 on the Assembler menu rather than the Formatter menu; space toggles between them). You should use this editor as any carriage return (cr) characters in the program will ensure that will not work.

Do not be too concerned with the number of spaces between commands since 'C' is a free format language. This means that so long as the program has at least one space where I have put at least one, it does not matter how the program is set out. In fact, you could condense it into a single, unreadable paragraph, and it will still compile successfully. Once entered, save it to the blank disk. I will use the filename "DSK1.PRINT;C". The last two characters of this filename are used to make it easy to find 'C' programs on disk catalogs.

Having typed in the program and saving it, we now compile it using the following procedure:

(1) Load the 'C' compiler from the 'C' disk using a program loader such as Funlwriter (option 3 - Loaders followed by option 3 - Program). The filename to enter is "DSK1.C99C" assuming the 'C' disk (side A) is in drive one.

(2) Answer "N" to both of the two prompts "Include c-text" and "Inline push code".

(3) Place the disk with the 'C' program into drive one. For the input filename, type in the filename of the 'C' program that you wish to compile. In this case it is "DSK1.PRINT;C"

(4) For the output filename, enter the name you wish to call the assembly language output. For this example, I will use "DSK1.PRINT;S". The suffix ";S" signifies that this file is an assembly language file.

(5) Once the compiler is finished, quit the program and select the assembler from Funlwriter (option 2 - Assembler).

(6) Type in the name of file output by the 'C' compiler for the "Source file name". In this case it is "DSK1.PRINT;S". Then enter a filename for the "Object file name". I will use "DSK1.PRINT;O".

(7) Leave the print device name blank and delete the "R" from the options, so that only the "C" remains. Press FCTN[6] and your 'C' program will be assembled. The resulting file "PRINT;O" is the machine language file that can be run.

Having finally compiled this short program, it can now be loaded. From Funlwriter, select loaders (option 3) followed by load/run (option 4). Place your previously blank disk in drive one and enter the the following filenames in sequence: "DSK1.CSUP", "DSK1.PRINTF" and "DSK1.PRINT;O". Then press <ENTER> without a filename. Finally, enter "START", press <ENTER>, followed by FCTN[6], and if all goes well, your efforts will result in the message "Your first 'C' program" on the screen.

Having seen the result of the program in Figure 1, we will examine its operation. The first line is simply a comment. Any text that appears between the "/*" and "*/" characters is ignored by the compiler. The second line tells the compiler that the PRINTF function is required, and that it will be loaded separately at a later time. The rest of the program contains its body. "main()" defines a subprogram called main, whose statements are enclosed by curly brackets ({ ... }). Each statement is ended with a semicolon (";"). In this case there are two statements: the first is the printf command, that prints the message on the screen, whilst the second gives you the option of either returning the title screen or rerunning the program.

I can not stress enough that if you want to learn 'C', or any other skill for that matter, you must practice what is shown here and then experiment. For example, you may be wondering what the two characters "\n" do, since that are not printed on the screen. Place these two characters in the middle of a message and see what happens (do not forget to recompile it). You can also find out what error messages the compiler gives: leave out a semicolon, or a bracket and see what happens. Only by experimenting (it cannot damage the computer) can you learn and understand.

```
-------------------------------------------
              Next Month
We will examine PRINTF more closely and
   look at some basic loops in 'C'.
-------------------------------------------
```

For Sale

Clock Programs

for Hard Disk

by Ben Takach

A number of cards designed for the TI99/4A feature battery backed real time clocks. Many TI99/4A users have purchased or assembled one of these cards from kit sets, yet one finds hardly any programs, which have utilised the date and time features of the inbuilt real time clock. I have to confess that many of my own programs have ignored the clock ticking away in my CorComp Triple Tech card for over half a decade. The arrival of the Myarc hard disk card changed it all. I soon got sick of setting the clock every time the system was booted up. A number of time related programs were incorporated in various programs which required time or date information. My programs naturally have been written for the Triple Tech and Myarc combination. The programs will have to be edited accordingly to suit John Paine's or Peter Schubert's cards.

Here is the basic, stand alone program, which was used in part in other programs to obtain the time or the date or both.

```
100 !SAVE WDS1.UTI1.TIME
101 OPEN #1:"CLOCK" :: INPUT #1:A$,B$,C$ :: CLOSE #1
102 SEC$=SEG$(C$,7,2):: MIN$=SEG$(C$,4,2)::
    HR$=SEG$(C$,1,2):: DAY$=SEG$(B$,4,2) ::
    MON$=SEG$(B$,1,2):: YR$=SEG$(B$,7,2)
103 OPEN #1:"TIME",INTERNAL,FIXED :: PRINT
    #1:SEC$,MIN$,HR$,DAY$,MON$,YR$ :: CLOSE #1
110 CALL CLEAR
120 DISPLAY AT(10,1):"SYSTEM CLOCK SET/DISP.  PROG."
130 PF=0 :: IF P$="" THEN P$="PIO"
140 DISPLAY AT(12,1):"Set/Display/Print clock or  End
    the program? (s/d/p/e)"
150 CALL KEY(3,R,S):: IF S<>1 THEN 150
160 IF R=83 THEN 170 ELSE IF R=80 THEN 340 ELSE IF R=68
    THEN 350 ELSE IF R=69 THEN 640 ELSE 150
170 CALL CLEAR :: IF SEC$="" THEN SEC$="55"
180 DISPLAY AT(4,5):"YEAR ? ";YR$
190 DISPLAY AT(5,5):"MONTH? ";MON$
200 DISPLAY AT(6,5):"DATE ? ";DAY$
210 DISPLAY AT(7,5):"HOUR ? ";HR$
220 DISPLAY AT(8,5):"MIN.  ? ";MIN$
230 ACCEPT AT(4,12)SIZE(-2):YR$
240 ACCEPT AT(5,12)SIZE(-2):MON$
250 ACCEPT AT(6,12)SIZE(-2):DAY$
260 ACCEPT AT(7,12)SIZE(-2):HR$
270 ACCEPT AT(8,12)SIZE(-2):MIN$
280 DISPLAY AT(10,1):"Any correction ? (y/n)"
290 CALL KEY(3,R,S):: IF S<>1 THEN 290
300 IF R=89 THEN DISPLAY AT(10,1):RPT$(CHR$(32),28)::
    GOTO 230 ELSE IF R<>78 THEN 290
310 OPEN #1:"TIME",INTERNAL,FIXED
320 PRINT #1:SEC$,MIN$,HR$,DAY$,MON$,YR$
330 CLOSE #1 :: GOTO 110
340 DISPLAY AT(18,1):"PRINT DEVICE ? ";P$ :: ACCEPT
    AT(18,16)SIZE(-3):P$ :: PF=1
350 CALL CLEAR
360 OPEN #1:"TIME",INTERNAL,FIXED
370 INPUT #1:SEC$,MIN$,HR$,DAY$,MON$,YR$
380 CLOSE #1
390 IF MON$="01" THEN MO$="JAN"
400 IF MON$="02" THEN MO$="FEB"
410 IF MON$="03" THEN MO$="MAR"
420 IF MON$="04" THEN MO$="APR"
430 IF MON$="05" THEN MO$="MAY"
440 IF MON$="06" THEN MO$="JUN"
450 IF MON$="07" THEN MO$="JUL"
460 IF MON$="08" THEN MO$="AUG"
470 IF MON$="09" THEN MO$="SEP"
480 IF MON$="10" THEN MO$="OCT"
490 IF MON$="11" THEN MO$="NOV"
500 IF MON$="12" THEN MO$="DEC"
510 IF PF=1 THEN 580
520 DISPLAY AT(10,9):DAY$;".";MO$;".19"&YR$
530 DISPLAY AT(11,9):HR$;" H.";" ";MIN$;" MIN."
```

```
540 FOR PAUSE=1 TO 4000 :: NEXT PAUSE ! ** you may
    shorten the loop delay at will**
550 DISPLAY AT(20,1):"Hold down key B to exit
    fromdisplay loop until minutes  are incremented."
560 CALL KEY(3,R,S):: IF R=66 THEN 110 !** or END at
    your option **
570 GOTO 360
580 DISPLAY AT(12,1):"Print 1 Date only, 2 Time only, 3
    Date & Time." :: OPEN #1:P$,OUTPUT !*insert spaces
    between options 1,2 and 3 according to desired
    screen display image **
590 CALL KEY(3,R,S):: IF S<>1 THEN 590 600 IF R=51 THEN
    610 ELSE IF R=50 THEN 620 ELSE IF R=49 THEN DF=1 ::
    GOTO 610 ELSE 590
610 PRINT #1:DAY$;".";MO$;".19"&YR$ :: IF DF=1 THEN 630
620 PRINT #1:HR$;" H.";" ";MIN$;" MIN."
630 CLOSE #1 :: DF=0 :: GOTO 110
640 END
```

How does it work?

Line 101 inputs the time and date string from the Triple Tech card.

Line 102 converts the string to a form, which the Myarc card can understand.

Line 103 will set the clock of the Myarc Hard Disk Card.

Lines 120 to 160 is the menu segment displayed on the screen. The SET option will enable the user to change the setting of the Myarc clock, if for any reason a different date or time setting is to be used.

Lines 170 to 300 deal with the entry data of the time and date. The entry routine uses an endless loop to correct any of the entered data. I use this routine in many of my programs where a number of questions have to be answered. The default data is already displayed, one may accept it or change it at will.

Lines 310 to 330 will change the setting of the Myarc clock.

Lines 360 to 380 will input the time and date information from the Myarc clock.

Lines 390 to 500 are used to change the numerals 01 to 12 into the name of the months. This is the best way to avoid the frustrating confusion of date display conventions. Most of the clock cards of USA origin use the mm.dd.yy convention (including Myarc).

Lines 520 to 570 are used to display the date and time at the centre of the screen. It is a continuous display, which is incremented every minute. Seconds are not displayed. Displaying the seconds is an overkill, the second count can hardly be read, thus confusing, besides the display of the seconds is almost always inaccurate. The program moves on when the letter "B" is pressed at the appropriate time. Impatient users, who get frustrated by the long delay may shorten the pause in line 540. Key unit 3 (as we all know), returns the upper case ASCII code regardless of the keyboard status set by the ALPHA LOCK key.

Finally lines 580 to 630 execute the print-out options.

A segment from the above program was incorporated in the LOAD file of the hard disk manager program, and also saved as an independent program by the title TIMESET. Timeset is incorporated in most of the LOAD programs I am using.

```
100 ! SAVE WDS1.UTI1.TIMESET
110 OPEN #1:"CLOCK" :: INPUT #1:A$,B$,C$ :: CLOSE #1
120 SEC$=SEG$(C$,7,2):: MIN$=SEG$(C$,4,2)::
    HR$=SEG$(C$,1,2):: DAY$=SEG$(B$,4,2) ::
    MON$=SEG$(B$,1,2):: YR$=SEG$(B$,7,2)
130 OPEN #1:"TIME",INTERNAL,FIXED :: PRINT
    #1:SEC$,MIN$,HR$,DAY$,MON$,YR$ :: CLOSE #1
```

Other programs, which require the input of date, such as an invoice writing program or the Keating's folly company car log book program may use another segment of the time program.

```
330 OPEN #1:"CLOCK" :: INPUT #1:A$,B$,C$ :: CLOSE #1
340 SEC$=SEG$(C$,7,2):: MIN$=SEG$(C$,4,2)::
    HR$=SEG$(C$,1,2):: DAY$=SEG$(B$,4,2) ::
    MON$=SEG$(B$,1,2):: YR$=SEG$(B$,7,2)
```

350 OPEN #1:"TIME",INTERNAL,FIXED :: PRINT
    #1:SEC$,MIN$,HR$,DAY$,MON$,YR$ :: CLOSE #1
360 GOSUB 1690

460 DISPLAY AT(7,1):"DATE ? ";DA$

490 ACCEPT AT(7,8)SIZE(-10):DA$

1690 REM DATE INPUT ROUTINE
1700 OPEN #1:"TIME",INTERNAL,FIXED
1710 INPUT #1:SEC$,MIN$,HR$,DAY$,MON$,YR$
1720 CLOSE #1
1730 IF MON$="01" THEN MON$="JAN"
1740 IF MON$="02" THEN MON$="FEB"
1750 IF MON$="03" THEN MON$="MAR"
1760 IF MON$="04" THEN MON$="APR"
1770 IF MON$="05" THEN MON$="MAY"
1780 IF MON$="06" THEN MON$="JUN"
1785 IF MON$="07" THEN MON$="JUL"
1790 IF MON$="08" THEN MON$="AUG"
1800 IF MON$="09" THEN MON$="SEP"
1810 IF MON$="10" THEN MON$="OCT"
1820 IF MON$="11" THEN MON$="NOV"
1830 IF MON$="12" THEN MON$="DEC"
1840 DA$=DAY$&"."&MON$&".19"&YR$
1850 RETURN

The above program lines were extracted from my invoice writing program. Note line number 490. This line gives a choice to accept or change the date, if for any reason the invoice date is not the actual date it was prepared.

Lines 1730 to 1830 may be considered by some observers to be superfluous embellishments. I included these, because of the number of conventions used worldwide to write the date. This way the day and the month is clearly defined.

Compare the lines 390 to 520 of the time program and lines 1730 to 1840 of the above program segment. If lines 390 to 500 are omitted from the former, then MO$ in line 520 have to be changed to MON$. Lines 1730 to 1830 may be omitted without any further change to express the months in arabic numerals. Some European countries use roman numbers to write the months. If this convention is adopted then MON$ will be made equal to "I" to "XII".

In conclusion, I use another finger and time saving routine to ensure that my Myarc RAMdisk is ready and waiting for me at DSK2 every time the system is turned on. This program unfortunately will not run from Extended BASIC, thus it has to be loaded and run from BASIC in two moves.

100 CALL PART(32,400,80)
110 CALL EMDK(2)
120 CALL CLEAR
130 PRINT "PUSH ALPHA LOCK DOWN, THEN TYPE: BYE & PRESS
    RETURN": : : : : : : : : :
140 END
150 REM PROGRAM NAME: START

The purpose of line 130 may need some explanation. This program is very useful if the computer is used by non computer minded office staff. My system is also used to send and receive telex messages. The outgoing telexes and the incoming messages are stored on the RAMdisk to save time. This program prepares the RAMdisk and ensures the use of the mandatory use of capital letters. The next program is started by the command: RUN "DSK1.TERMINAL".

100 !SAVE DSK1.TERMINAL
110 OPEN #1:"CLOCK" :: INPUT #1:A$,B$,C$ :: CLOSE #1
120 SEC$=SEG$(C$,7,2):: MIN$=SEG$(C$,4,2)::
    HR$=SEG$(C$,1,2):: DAY$=SEG$(B$,4,2) ::
    MON$=SEG$(B$,1,2):: YR$=SEG$(B$,7,2)
130 OPEN #1:"TIME",INTERNAL,FIXED :: PRINT
    #1:SEC$,MIN$,HR$,DAY$,MON$,YR$ :: CLOSE #1
140 CALL CLEAR :: DISPLAY AT(1,1):"CHECK SYSTEM
    CONFIGURATION: MODEM SET TO: 300 orig
                    switch NOT ON BELL (up)"

150 DISPLAY AT(5,1):"auto/manual switch off
                            PRINTERSELECT:
                    switched to MP"
160 DISPLAY AT(10,1):"PRINTER: .   paper
    roll fed in,   platen press lever engaged,
    power & on line lights on"
170 DISPLAY AT(14,1):"pitch set to 12 char/inch   (dip
    sw. 1 off, 2 on)"
180 DISPLAY AT(18,1):"PRESS ANY KEY WHEN READY TO
    PROCEED" :: CALL KEY(0,R,S):: IF S<>1 THEN 180
190 CALL CLEAR :: DISPLAY AT(12,1):"the name of the
    parameter   file is: LOGON
200 FOR PAUSE=1 TO 500 :: NEXT PAUSE
210 RUN "DSK1.FASTTRM1XB"

The time/date routine is again incorporated in this program besides its other functions. It will remind the operator to prepare the printer, then it will run the terminal program.

The examples show the numerous applications of the real time clock in a TI99/4A environment. Not to speak of the enjoyment of the astonished looks and the inevitable question from the onlooking fellow 99'ers "how did it do it?"   o

Disk Drive Power Supply

Tigercub Public Domain Catalog #2

by Jim Peterson, Tigercub Software, USA

This file contains details of disks 634 to 890.

634. STRANGE MUSIC (337)
Axel F (Bob Gagle); The Battle of New Orleans (E. McFall); Wilbur the Musical Worm (L. Newcombe); Beethoven Symphony #5 in C Minor (W. Brewster); Shaded Velvet (Jeff Gatlin); Demonstration and Demonstration #2 (Hank Mishkoff); Assorted Music Demonstration (R. Kazmer); Molly Darling, Solo Mit Oompah and Song of Samarkand (J. Peterson); Down in the Valley, Strange Music and Strange Music #2.

635. CHUCK BERRY TUNES by Ken Gilliland (200)
School Days, Reelin' and Rockin', Brown Eyed Handsome Man, Thirty Days, Too Much Monkey Business, Mabellene, and Madhouse; now all prescanned.

636. CHRISTMAS SONGS with graphics by Karl Romstedt.
Jingle Bells, Santa Claus Is Coming To Town, We Wish You A Merry Christmas, Hallelujah Chorus, Angels We Have Heard On High, It Came Upon The Midnight Clear, Silent Night, Medley (Knecht); The First Noel (Christensen); Twelve Days of Christmas, Deck the Halls (Romstedt).

637. ASSORTED MUSIC #9 (337)
Hill St. Blues Theme (G. Mras); Larry's Fiddle Tunes (L. Schott); War Eagle (Mark Jones); Here Comes The Sun (D. Duplissey); Hey Diddle Diddle, Looneytune, Oh Susanna, The Entertainer, Let It Be, Old McDonald Had a Farm, Rock Around the Clock, 59th St. Bridge Song, Close Encounters of the 3rd Kind.

638. CLASSICAL MUSIC #2 (338)
Jesu Joy of Man's Desire (J. Clulow); Goldberg Variation #1 (B. Pomicter); Moonlight Sonata (K. Noesner); Prelude in C Major (Bud Wright); Music by Merlin; Concerto #3 (Corey); 18th Century Drawing Room, Bach Mini-Concert, Tales from Vienna Woods, Invention #13.

639. ASSORTED MUSIC #10 (341)
Fantasy (S. Dicks); Seasons in the Sun (E. Bickerman); Oxygen Part IV (Christensen); Arirang (J. Peterson); Easy Winners, Pachbel's Canon in D, Pink Panther Theme, Calliope, Gymnopedies, and untitled music #1, #2, #3.

640. MARCHES and COLLEGE SONGS (336)
Military Music collection by Landrum and Knecht; also Anchors Aweigh, College Fight Song, Patriotic Medley, The Eyes of Texas, Go Hawks, King William's March; and July 4th (Terrence Murphy)

641. SING-ALONG MUSIC #2 (345)
Don't Fall in Love With a Dreamer (Knecht); It's a Small world (D. White); The Jordache Look (Ted Peterson); A Capital Ship, Old Dan Tucker (R.D. Fetters); A Hard Day's Night, Please Help Me I'm Falling, Shine On Harvest Moon, Ob-La-Di, Hello Goodbye, Mame, Twinkle Twinkle Little Star, Five Foot Two.

642. SING-ALONG MUSIC #3 (236)
Fame, What I Did For Love, Puff the Magic Dragon, Hey Jude, You've Got To Hide Your Love Away, America the Beautiful, Only Love Can Break Your Heart, I Have Decided to Follow Jesus, Peace in the Valley, Hallelujah Thine the Glory, What a Friend We Have in Jesus.

643. CHRISTMAS MUSIC (351)
Coventry Carol, Joy to the World, Jingle Bells, Adeste Fideles, Merry Christmas, Frosty the Snowman, Deck the Halls, Merry Christmas #2, all anonymous; Season's Greetings (Brian Sutton)

644. CHRISTMAS SING-ALONG (340)
Hark the Herald Angels Sing, Adeste Fideles, Holly Jolly Christmas, Rudolph the Red-nosed Reindeer, The Little Drummer Boy, Rockin' Round the Christmas Tree, The Christmas Song, Let it Snow, Sleigh Ride, The Twelve Days of Christmas, Twelve Days #2, Silent Night, all anonymous; Christmas Sing-Along (C. Schwerin)

645. CLASSICAL MUSIC #3 (352)
Polonaise in G Minor, Prelude in G Minor, Syrinx, Waltz by F. Carulli, Der Hase in der Landschaft, Haydn's Sonata #5, Rustic Dance, The Masterpiece, Choral by Schumann, Country Dance by Beethoven, Constante, Two Note Melody by Schumann, Rondo a la Turca, Hornpipe by Purcell, Adelita, Lagrina.

646. ASSORTED MUSIC #11 (350)
Fiddler on the Roof (B. Jackson); Himmel und Erde (M. Tsukroff); Land of Moran (F. Krauter); Feels So Good, Hey Paula, Seahorse, Pluto, Greensleeves, Row Row Your Boat, Michelle Yesterday, America, Ave Maria.

647. MUSIC DOODLERS and TINYTUNES (248)
Autochorder, Bell Music, Minor Chorder, Brown Music #2, #3, #4, #5, Musical Bargraph, Mockingbird, Musik, Tunes #1, #2, Frankie and Johnnie and #2, Wabash Cannonball Wearing of the Green, Wildwood Flower, Buffalo Gals, all by Jim Peterson; Musical Joysticks (B. Nelson); Pocket Canon (S. Holl); Steinway 99/4A (M. Christianson); Pop! Goes the Weasel (J. Clulow); Solo (T. Niemietz) and Solo mit Oompah; Rustic Dance (John Charlton); and Echo, Boogie Bass, Echo Sound Effects, Joystick Music Player, MICROpendium Organ (with added bass), Music, Musique, Careless Love, Keyboard Music Player, Colour Organ Mission Impossible; Baa Baa Black Sheep (J. Taylor); TI-99/4A Player Organ (Sam Moore Jr.)

648. RHAPSODY IN BLUE, GEORGE GERSHWIN'S CONCERTO IN FOUR PARTS, by Don Maguire, 287 sectors.

649. ASSORTED MUSIC #8 (354)
Blue Danube Waltz (S. Williams); Dueling Banjos (G. Christianson); The Entertainer (Ed Butcher andc); Gavoti Hoedown (J.S. Foster andc); Incredible Hulk Theme (J. Cummings); In My Life (D. Duplissey); Blue Boogie (K. Noesner); Frere Jacques; Ghost busters ; The Godfather Theme.

650. CHRISTMAS MUSIC WITH GRAPHICS (357)
Christmas Card, Jolly Old St. Nicholas (Regena); Frosty The Snowman, Santa Claus (K. Wakefield); Merry Christmas Medley (Houston User Group); Christmas Card, Merry Christmas, Ziggie's Christmas, Christmas Card, all anonymous; Christmas Medley (Bob Lanouette)

651. SORGAN II by Jerome Trinkle (145) A fantastic keyboard music player.

652. CHRISTMAS MUSIC WITH GRAPHICS (352)
Seasons Greetings (H. Stevenson); Deck the Halls, Do You Hear What I Hear, Gesu Bambino, O Holy Night, We Wish You A Merry Christmas, Lullay, It Came Upon A Midnight Clear, Lo How A Rose E're Blooming, The Christmas Song, all anonymous.

653. POP DEMO V1.1 (225)
Fantastic music in assembly, by Roman Majer (copyright abandoned). In The Mood, Amorada, Waltz of the Fleas, Charleston. NOTE: Must be run from the Editor/Assembler module.

654. CHRISTMAS MUSIC WITH GRAPHICS (255)
Away In A Manger, O Holy Night, Joy To The World, Silver Bells, Jingle Bell Rock, Blue Christmas, I Heard The Bells, Here Comes Santa Claus, Frosty The Snowman, Rudolph The Red-nosed Reindeer, all by Gerry Myers with graphics by J. Crosson, G. Higgs, Larry Williams, Dick and Kay Webber.

655. ASSORTED MUSIC #12 (349)
Le Secret (S. Johnson); Over the Rainbow (S. Davis); The Boxer (T. and B. Berg); Chopsticks (P. Doyle); I'd Like to Teach the World to Sing (R. Weitkowski); Star Trek Theme (B. Fishman); Star Wars Theme (M. Benson); Release Me (C. Crowder); Snoopy, Music, Never on Sunday, Song of Joy, European Folksongs, Selections, Gadego Demonstration, Be Still My Soul

701. MUSICAL EDUCATION #3 (350)
     Ear Training for Music (Rob Williams); Guitar Calculator (B. Pomicter); Major Key Signatures (Regena); Chords (J. Clulow); Music Magic (C. Burris); Music Maker (C. de Marti); Music Maker (Coppens/Swinnen); Canon of Purcell (C. Arribet); Music Maker (Tsoi/Leung); Intervals: Lesson 1, Rhythm, Transposition.

702. MUSICAL EDUCATION #2 (318)
     Mystery Words, Intervals, and Rhythm (J. Clulow); Name the Note (Regena); Notes Rests and Beats (D. Pribble); Programmable Metronome (D. Thrasher); Rhythms; Keyboard Computer (B. Pomicter); Drum Machine; Bach Invention in F (J. Clulow); Computer Music (S. Anderson)

703. MUSICAL EDUCATION #3 (188)
     Let's Learn Notes; Music Terms Quiz; Guitar Tuning Program; Instrument Tuner; Violin Tuning; Music Player; A Study in Musical Scales; Minuet from Linz Symphony

710. AMERICAN FLAGS (360)
     Dixie, Confederate Flag (Landrum and Murphy); Georgia State Flag (M. Brown); Indiana State Flag (A. George); Iowa State Flag and Song; Maryland State Flag and Song (R. Ambrose); Ohio State Flag and Song (H. Jareszewski); Oregon Flag (B. Schappert); Tennessee State Flag (C. Ringold); Texas Flag (C. Ehninger); Wyoming State Flag and Song (C. Whitelaw); American Flag (Ed Stamm)

711. FLAGS OF THE WORLD (345)
     Australiana (G. Jones); Australian Flag (J. Volk); Canadian Flag, Anthem (L. Sablauskas); O' Canada (B. Knecht); Flag-o-Rama; Flags and Nations Volume 1 and Volume 2 (K. Williams): Flags of Nations (L. Brown); Know Your World Flags (C. Christianson)

712. U.S. STATES (341)
     State Capital Quiz (T. Siseman); States and Capitals; U.S. States (J. Pulliam); Chloropleth Map (Dent/Whitelaw); Eastern U.S. Capitals (D. Smith); Identify States, of New England, of South, and of West, all by Regena; Let's Build America (C. Flotta); Sea of States (J. Phillips)

713. U.S. STATES #2 (200)
     States and Capitals; Ohio Regions (K. Kuehnle); States; States and Capitals (R. Schenk); States of the USA (J. Peterson)

714. WORLD GEOGRAPHY (179)
     Nations of America and Asia; Nations of Europe and Africa; Australia (P. Pruszinski); Geographical Distance (D. Thorpe); Geography Quiz (Richards/Cardo); Nations of the World (Jim Peterson); World Map.

730. AMERICAN HISTORY (48)
     American Presidents; Presidents of the U.S. (J. Peterson)

750. ELEMENTARY LETTERS and NUMBERS, WITH SPEECH (343)
     Talking Back, Speakmath, Read*Listen*Answer, Alphabet Song, Reading Comprehension, Pre-Speller, Alphasong, See-Spell-Speak, all by J. Peterson; Speller (R. Binkowski); Alphabet Recognition (L. Tutchings); Alphanum Delight (J. Taylor); Alphaspeak (K. Romstedt); See-N-Say (Lyons/Peterson); Abraham Lincoln (R. Learned); Abbeybet Plus (E. Neu)

751. CHILDREN'S PROGRAMS W/SPEECH (357)
     Mother Goose, Sick Robot Jokes, Rocky Robot Sings Old McDonald (S. Moore Jr); Speech Fun (Valentine/Foster); Out House (P. Anderson); Pano the Train (J.S Foster); Alpha Sing (J. Peterson); Reef Knot, Bowline (K. Denham); Kroakers (O'Berg/Hitch); Ernie and Bert.

752. ALPHABET FOR PRESCHOOL (329)
     Apple Cruncher (J. Smieja), Under the Sea ABC, Talking Alphabet (D. Veith), Colour Bars, Capitalize (G. Heine) and Big Letters (P. Yorke), all with speech; Alphabetize; Large Lower Case Letters (D. Carrera); Lower Case Convert (B. Falkin); Word Shooter (D. Steffen); Word World (A. Falco)

753. CHILDREN'S PROGRAMS W/SPEECH (335)
     Word Twins (S. Moore Jr); Madlib with speech (P. Yorke); Magic Machine (D. Harris); Mind Reader (J. Peterson); Pig Latin (P. Yorke); Piggy Bank (J. Heine); Secret Numbers (A. Graham); Sing-Along (V. Wicks); Singing Happy Birthday (McCluskey/Knecht); Abraham Lincoln (R. Learned); Visit from St. Nicholas, Speech Demonstration, Shuttle Blasting Off, Simon I.

755. COLOURS, SHAPES, DIRECTIONS (173)
     Spatial Relations (K. Martin) with speech; Building Blocks (T. Castle); Building with Bricks (TI-Soft); Colour Game (K. Kuehnle); Draw; Shape Art(Ed Neu); Spatial Relations (K. Martin, w/speech)

760. SPELLING PROGRAMS (324)
     Speak Spell Flash in 5 parts, with speech, by Sam Moore Jr.; Spelling Tutor Age 9-10 (Towers/Gill), with speech; Spelldown (Dheins); Spelling Quiz (Orwig/Hodges); Spelling Skill (O. Hebert)

770. VOCABULARY and READING (293)
     Adjective to Adverb, Noun to Adjective, Learning to 'ing' It, Plural Endings, by Jim Peterson; Animal Multitudes (Jack Sughrue); Doctor Who (V. Maker); Vocabulary (D. Hillebrand); Vocabulary Quiz (Rugg/Feldman); Syllables (Orwig/Hodges); Reading Practice (with Terminal Emulator II speech); Speed Reading; Tense Time; Synonyms and Antonyms (R. Brown); Read-Fast.

780. PRESCHOOL MATH (341)
     I Went to Visit a Farm (S. Nichelson); House Numbers (K. Romstedt); Preschool (J. Schwaller) with speech; Preschool Block Letters (H. Drake); Speakmath (Peterson) with speech; Tell the Time (Rozler/Horwitz); Monkey Business (T. Castle); Be a Clown (P. Rauktis); Climbing Math (J. McDonald); Counting Coins (K. Monson); Counting Fun (J. Brantley); Eric Adds Up (P. Leathley); First Math (C. Christensen)

790. ELEMENTARY ADDITION and SUBTRACTION (282)
     Add-Ed (G. Mineo) with speech; Basic Addition and Subtraction (M. Dewese) with speech; Colour Math (M. Griggs); Counting Lesson (Jenkins/Kersling) with speech); Elementary Arithmetic; Laserithmetic (C. Christensen); Snertle.

791. ADDITION and SUBTRACTION (337)
     Math Competency Buying, Math Competency Earning (Regena); Math Drill with speech (L. Hughes); Math Quiz (R. Shirk); Math Teacher (Orwig/Hodges); Math Tutor (Fuller Ent.); Math Practice (B. Rutherford); Number Speaker, Changemaker, Making Change, Add and Carry, Take Away (Peterson); Numtalk (A. Persson); Superfly, Math Baseball.

796. MULTIPLICATION and DIVISION (348)
     Basic Number Facts, Hare and Tortoise, Long Division (C. Christensen); Be Fruitful and Multiply (R. Schenk); Division Tutor (S. Lisonbee); Homework Helper Division, Homework Helper Fractions (Regena); Greatest Common Denominator, How Many?, Math Quiz, Magic Nines (J. Peterson); Math Practice (Fetters); Missile Math (J. Adelmann); Russian Multiplication (B. Traver); Fractions.

797. MULTIPLICATION andC (224)
     Multiplication (J. Craig); Multiplication S. Maschue); Multiplication Madness (Juckett/Yorke); Multiplication Practice (R. Ambrose); Multiplication Skill (O. Hebert); Decimal to Fraction; Roman Number Conversion (A. Rogers); Multiplication Invaders (D. Perkovic); Number Speaker (Peterson) with speech; Train (W. Koetke); Decimal to Fraction; Speed Math; Train (W. Koetke); Basic Number Facts (C. Christianson); Times Bomb (HV 99ers)

800. HIGHER MATH (355)
     Algebra (D. Decker); Catapult Capers (M. Wade); Linear Regression (Poole/Borchers); Binary Numbers (G. Schechter); Board of Galton (S.D. Netherlands); Coordinate Geometry; Curve Area Analysis (W. Zipf); Incremental Algorithm (A. Dray); Leibniz Series

Tigercub Software Catalog #3

by Jim Peterson, Tigercub Software, USA

TC-120. HAUNTED GRAVEYARD. You must cross the graveyard on a dark stormy night. Lightning flashes will light your way, but you may be struck by the lightning. If you fall into an open grave you will never be seen again, and if you bump into a gravestone you will be chased by a ghost. 5 screens of increasing difficulty. Challenging, and fun! TCX-1120 in Extended BASIC.

TC-121. SPALLING TEECHER. Accepts spelling list by input, offers options of displaying words by flashing for a preselected interval or scrolling in multiple prints or filling the screen with letters in sequence; checks and corrects each letter in answer by a flashing checkmark; repeats misspelled words, also repeats them at end of lesson. TCX-1121 in Extended BASIC.

TC-122. TIGERCUB GRAPHICS DESIGNER. This program makes it easy and fast to design full screen block graphic pictures and designs in up to 15 colors then prints them out as strings which can be copied and incorporated into programs. TCX-1122 in Extended BASIC (disk only) saves the design to disk in MERGE program format, ready to run as a program or to merge into a program.

TC-123. HOMONYMY. Explains and quizzes homonyms, adds one more note to a tune for each correct answer. Loaded with 518 words, just about every true homonym. 3 skill levels. TCX-1123 in Extended BASIC.

TC-124. ANTONYMY. Similar to the Synonymy program but teaches antonyms and uses words of elementary school level. Prints a word and a list of 6 words from which an antonym is to be chosen to add a note to a melody. TCX-1124 in Extended BASIC.

TC-125. HIEROGLYPHY. A challenging cryptographic puzzle. Decipher one of a dozen messages from "ancient Egypt" or have a friend type in a message for you to solve. Also includes KRYPTOGRAMMER, a more conventional cryptogram puzzle maker.

TC-126. 4x4 BIT DOODLER. Use the joystick to doodle in medium resolution graphics, even write your name; or put a grid on the screen and design detailed drawings to put in your programs.

TCX-1126 in Extended BASIC (disk only) saves the design to disk as a MERGE format program which you can load and run or merge into another program.

TC-127. OLD TIMER PUZZLE. The grand-daddy of the coin switching puzzles, and the best of them, now in a computerized version in which you progress through 8 levels of difficulty or let the computer show you how. TCX-1127 in Extended BASIC.

TC-128. TEN THOUSAND SIGHTS. Just another kaleidoscope program but this one offers you 8 absolutely unique patterns, all with an infinite number of changes, including the Christmas Ornament which you design yourself, Checkerscope, Dancing Lights, Random Kaleidoscope, the Tigercub's Kaleidoscope, Kaleidoscreen, the almost right Krazy Kaleidoscope, and the Rorschach Kaleidoscope, the computer inkblot test! TCX-1128 in Extended BASIC.

TC-129. MECHANICAL APTITUDE TEST. A very challenging and entertaining computerized version of the broken block puzzles used in IQ tests, ACT tests, etc., with endless variations. One of my best! TCX-1129 in Extended BASIC.

TC-130. MATH HOMEWORK HELPER. For addition, subtraction or multiplication of numbers of up to 9 digits, addition of up to 9 numbers, generates random problems or accepts input of student's homework. It will not solve the problem for the student but it will not let him make a mistake; accepts input of answer from right to left, refuses erroneous digits with a deep groan and a disappointed look!

TC-131. JUNIOR SPEEDER READER. A version of #TC-74 Speeder Reader with words of elementary school level. Teaches the kids speed reading, reading comprehension, memory retention and spelling, while they have fun with the ridiculous sentences. TCX-1131 in Extended BASIC.

TC-133. WHITE KNIGHT. You are the White Knight, and your Quest is to gather the Golden Treasures while avoiding the Scarlet Dragons. A fun game of strategy for the kids. TCX-1133 in Extended BASIC runs a little faster.

TC-134. MUSIC PROGRAMMING TUTOR. Everything about programming music in BASIC that I could get into one program. In BASIC on cassette only. TCX-1134 in Extended BASIC with much more, on disk only, requires Memory Expansion.

TC-135. BARS AND BALLS. A solitaire game of shifting bars in order to let balls drop into a bucket. Addictive. TCX-1135 in Extended BASIC.

TC-136. THREE COIN WEIGHING PUZZLES. These classic puzzles are ideal for computerization, and here are three of the best.

TCX-1137. SOUNDMAKER. A very versatile utility program to develop sound effects, then save them in the form of actual program lines. Requires Extended BASIC; disk only.

TCD-501. GENERAL PURPOSE MULTIPLE CHOICE TEST PROGRAM with 4 pages of printed instructions which make it easy for anyone with no programming experience to construct any number of multiple choice tests on any subject, with many options and variations. Because of the extra printed documentation, this program is $4.00.

****

Some folks have been asking if my monthly newsletter to the user groups was available as back issues or by subscription. The answer is sorry, no but the complete contents are available, all keyed in and ready to run, as five full disk collections, now at greatly reduced prices

TIPS FROM THE TIGERCUB (#1) $10 postpaid

A full disk containing the complete contents of the Tips from the Tigercub newsletters #1 to #14, and more besides 50 original programs, routines and files all for just $10 postpaid. Contents include programs to make your own WORDSEARCH PUZZLES and CRYPTOCODE puzzles, to design your kaleidoscopic CHRISTMAS ORNAMENT, to "turn the TV upside down", and to print, spell and speak any number up to 999 vigintillion. Also included is the computer composed music of the SONG OF SAMARKAND, the secret of the two octaves of bass notes, the routine to play tremolo music, the BELL MUSIC program, and three different random music composers. Others are the ANAGRAMMER to unscramble word puzzles, the MAGIC SQUARE MAKER, the ADDER-UPPER, HEARING TEST, the TWO WAY PRINT routine, DOLLARS AND CENTS printer, the extremely useful 28 COLUMN FORMATTER, and much, much more.

TIPS FROM THE TIGERCUB #2 $10 postpaid

A full disk containing the entire contents of the Tips from the Tigercub newsletters #15 to #24, containing over 60 programs, routines and files including utility programs such as DOWNCHAR to design special printer characters, CURSOR CHANGER to design your own custom cursor, EXTRACTOR to save any block of lines from a program, COMPARE to prepare a mergeable file of differing lines in two programs, LINEWRITER to easily format screen text, NAMELOADER to catalog and run programs from their full titles, instructions and programs for a GROCERY SHOPPING LIST, and for a method of printing documents in 2 or 4 columns in one pass, and a novel idea for a membership list. There are games and

entertainment programs such as the AUTOMATIC MOUSE MAZE, QUICK AND DIRTY DOODLER and KEYZAP, music programs such as ALABAMA 4th OF JULY, MICRO-ORGAN and DANCING STICKMAN; unusual displays such as the hypnotic ETERNITY, also PATCHES, WILL O' WISP, MOSQUITO and KALEIDOSQUARES; educational programs such as PLURALS, TOUCH TYPING and ALPHASONG, as well as ALPHASING which actually sings the Alphabet Song. And there are files and routines to show you how to write programs that write programs, to select random numbers without repeating, to shuffle numbers quickly without repeating, and to reveal many other little known tricks of programming the TI99/4A home computer.

TIPS FROM THE TIGERCUB #3 $10 postpaid

Another disk full of 62 programs, routines, tips and tricks from newsletters #25 through #32. Contents include extremely useful utilities such as CATWRITER and QUICKLOADER for a fast loader by complete program name rather than filename; DOUBLECAT to check the contents of a backup against a master; CHECKER to compare two programs; PROGLISTER to list a program to the printer in exact 28 column format; GO-SEARCH to find and list GOTO and GOSUB lines before modifying a program; GRAFWRITER to save a graphics screen; music and graphics programs GREENSLEEVES, FRANKIE JOHNNIE; challenging computer puzzles; remarkable 3 dimensional sprite displays 3-D SPRITES and SPRITE SHUFFLE; printer routines for useful charts and STREAMER to print vertical banners in any size; a program MIND READER that actually reads your mind three times; tips on using TI-Writer; a program to code and encode messages which is very difficult to break; little known secrets of the TI99/4A's CALL SOUND; valuable programming tips and tricks 2 dimensional sort, ACCEPT long lines, etc.; the tiny but challenging COVER UP game; and much, much more.

TIPS FROM THE TIGERCUB #4 $10 postpaid

A full disk of 48 original programs, routines and files from Tips news letters #33 through #41. Contents include BXB, a 4 sector routine to permit any BASIC program to run in Extended BASIC or any Extended BASIC program to use 17 character sets; DECOMPACTOR to break an Extended BASIC program into single statement lines before modifying it; KEYSEARCH to search a file for one to several key words simultaneously and in combination with other keywords; and PRINTSPEAKER to convert all text and input prompts to TEII speech. The MULTIMEMORY CALCULATOR features 5 memories and 5 windows, plus multiple inputs! There are innovative musical programs including MOLLY DARLING with changeable volume, tempo, key and scale; WILDWOOD FLOWER in the style of a hammered dulcimer; SOLO MIT OOMPAH to play along to a tuba accompaniment; and SHENANDOAH with beautiful graphics. KEYTALK and SEE-N-SAY are preschool education programs using speech; ADD & CARRY and TAKE AWAY are excellent elementary education; MATH QUIZ shows how to work the problem if a wrong answer is given. ESCHER ART and BASKET WEAVING are fascinating graphics programs of endless variety, GORDIAN KNOT is a 3 dimensional maze doodler, BLOB is a spooky one for Hallowe'en, and there are still more. The SORT WATCHER allows you to watch any sort in progress, MAGIC NINES is a mathematical curiosity which predicts your answer, and SPEED TYPING TEST is a very unusual typing program. And there are puzzles, including a one liner 34 lines long!

TIPS FROM THE TIGERCUB #5, $10 postpaid

Another full disk of 49 programs and routines from Tips newsletters #42 to #50. Includes the educational programs ADJECTIVE TO ADVERB, NOUN TO ADJECTIVE, PLURAL ENDINGS, HOW MANY FROGS, and PRESPELLER; the extremely versatile PRINTALL to accept all standard printer codes and print multiple column multiple files; the unique SUPERTRACE to trace a program statement by statement to screen or printer; an improved CATWRITER to create your own QUICKLOADER menu loaders, and with utility files to modify it; several useful subprograms with demonstrations routines; various routines demonstrating peculiarities ties of the TI99/4A; several one screen "tinygram" programs; and much more.

****

NUTS BOLTS DISK (No. 1) $15.00 postpaid

100 Extended BASIC utility programs in MERGE format, ready to be MERGEd into your own programs. Line numbered consecutively in high numbers so that they will not interfere with your program or with each other. Contents include:

13 type fonts including giant, stylized, slanted, enlarged, upside down and inverse characters, compressed numbers, Russian, slashed zero, etc.

13 text display routines including scrolling, fade in, instant screens, columnizing, etc.

10 screen wipes including border and wipe, curtains, fade out, etc.

8 pauses including key holds and stop and go, music while you wait, music while you read, etc.

3 programming aids including a screen grid, check routine, and the kill quit routine.

9 data saving and reading routines including some little known memory saving methods.

12 sorts and scrambles for both numeric and string data, inserting data, shuffling, etc.

3 time and date, perpetual calendar, etc.

And protection routines, printer aids, joystick and cursor controls, math, music routines, etc.

Plus a tutorial program on using subprograms, and the Tigercub Menu Loader, and 5 pages of documentation with an example of the use of each subprogram. o

For Sale

Myarc Floppy Disk Controller card. Upgrades your TI99/4A system to double sided/double density. Main advantage is 100% speed increase over TI controller card in disk copying, formatting etc. Comes complete with manual and the BEST disk manager, bar none! Optional with 80 track EPROM ($10 extra), this upgrade allows up to 720K of storage (2880 sectors on disk). All this for only $150. Phone (042)84 2980 up to 10:30pm

Full height DSDD 80 track disk drive, gives 720K (2880 sectors on disk) of storage. Standard fitting in PEB. Suitable for use with Myarc FDC (fitted with 80 track EPROM) or Myarc HFDC. Uses standard 5.25 inch floppy disks and also reads/writes 40 track formatted disks. If you already own a Myarc FDC, I will throw in the 80 track EPROM upgrade for $10 if you buy the drive! Asking $100. Phone (042)84 2980 up to 10:30pm

TI RS232 PEB card for use with any printer (either RS232 or Centronics, Dot Matrix or Daisywheel), or any modem. Complete with full documentation, only $145. Printer cable (works with any printer) $20 extra. Phone (042)84 2980 up to 10:30 pm o

There are many other applications within programs to which strings can be put. If you want to learn how to use strings fully, think up some tasks for which strings can used and then write a program to carry out the task. Also explore the use of the other string functions that were not discussed here.

Refere _._.:
Users ˙ ˙ ¸nce Guide. Pages II-10, II-15 and II-99 to II-103.
Extended BASIC Manual. Pages 50, 60, 110, 145, 160, 166, 179 and 188. o

Legends part 3

typed by Larry Saunders

Potions (I,D):

There are six different potions available in Legends. Actually, there are three types, Version 1.1 and two types in early versions and they come in three different strengths. Potions are entirely beneficial. Some will heal wounds and others will restore magic points. The following is a list of their effects:

HEALING1 restores 1 to 9 hit points,
HEALING2 restores 1 to 20 hit points,
HEALING3 restores 1 to 40 hit points,
MAGIC1 restores 6 to 18 magic points,
MAGIC2 restores 7 to 25 magic points.

Version 1.1 has a new potion, STEALTH: This replaces the potion "MAGIC #3" described below. If a STEALTH potion is consumed by any party member, an aura of invisibility will envelop the entire party. This allows the party to sneak by monsters undetected. Note: this potion wears out in time.

MAGIC3 restores 8 to 30 magic points.

Potions can be purchased from either the alchemist of Wizard's Rock or found in the Dungeons.

Terms and abbreviations

AC,ARM... Armour class, the higher the better armour you have.
ATTK... Attack skill.
CAST... Spell Casting skills.
CDN... Condition, or number of hit points remaining before death.
DEFY... The Resist Magic Spells ability.
EXP... Number of Experience points.
HIT... Number of Hit points the player can have.
PRO... Protection bonus.
PROTECT LEVEL... The AC or armour class of an item.
STA... Character status -this area will turn blue if a character is dead or otherwise will display the characters class.

Suggestions for Playing

(1) Find a way to enter the Wizard's Rock Temple. It will help you throughout your quest.
(2) Find the correct numbers to activate the Teleporters. A fast escape from some areas can be an excellent decision.
(3) If you find information of value, write it down.In some areas you will need very specific information.
(4) Passive magic spells can sometimes be more effective than active magic spells depending on the monster you are facing.
(5) Always rest at the Legends Inn after training a character so he or she is up to his or her maximum hit and magic points.
(6) Remember that all dungeons have multiple levels (as many as four). Sometimes the doors are hard to find, but they are there.

Combat Reference Sheet

1) FIGHT 2)GREET 3)RUN 4)THREATEN 5)SURRENDER 6)ATT FORM

FIGHT: Switches Legends into individual mode.

GREET: This is essentially saying "hello" to the monsters. The monsters may want to avoid a confrontation and greet you and leave, or attack you despite your friendly overture.

RUN: Version 1.1 Can select this if you want to attempt to flee from combat. Early versions must Combat at least once.

THREATEN: Sometimes, if you intimidate the monsters enough, they will give you gold to leave them alone.

SURRENDER: If you have enough gold, you can bribe the monsters into letting you pass.

ATT FORM: If you want to change the order in which your characters fight, use this option, ATTack FORM to do it.

Choosing FIGHT brings up the following menu:

1)HIT 2)LUNGE 3)PARRY 4)CAST

HIT: Take a normal swing at your opponent.

LUNGE: A less controlled, wilder, but potentially more devastating swing.

PARRY: Used for placing a character in a defensive stance during the monsters turn to attack.

CAST: Used to cast magic spells or evaluate monsters.

Legends Magic Spells Reference Sheet

Magic Used in Legends

Magic spells play a very important part in Legends. Using spells properly and with good judgement can determine the success or failure of even the strongest party. Magic spells fall into two basic categories:

Active and Passive spells.

Passive magic spells always work. These include healing spells and other spells to protect the members of the party in combat. Active magic spells are more difficult to cast successfully. These spells take into consideration the cast spell skill of the caster versus the resist magic ability of the adversary the spell is being cast upon. This type of spell is offensive in nature and is intended to cause damage to an opponent.

Each spell requires a certain amount of energy to cast. These are called Magic. A character can restore depleted Magic points by drinking Magic potions or by staying at the local inns to rest.

Types of Spells

FIRESTORM(1-3): This fires a blast of pure energy at the opponent the spell caster is facing in combat. Depending on the spell level and skill of the caster the spell will do the following damage:
FIRESTORM1 - 1 to 10 damage
FIRESTORM2 - 1 to 40 damage
FIRESTORM3 - 1 to 99 damage

DISPEL MAGIC: Neutralizes any monsters spells currently in effect.

LIFESTEAL: Does no immediate damage but robs all monsters you are fighting of some of their hit points.

ROT ARMOUR: Destroys all or part of a monsters armour, making them more vulnerable to attack.

RESIST MAGIC: Renders the party less vulnerable to spells cast by attacking monsters.

WEAKNESS: Lessens the amount of damage a monster can do if he strikes a character.

TURN UNDEAD: Will immediately destroy any one undead creature. Works only on undead creatures.

HEALING(1-2): Version 1.1, (1-3): early versions. Allows the spell caster to heal some or all of the injuries of any character, including the caster. Depending on the level of the spell and the skill of the caster it will restore hit points as noted:
HEALING1 - 1 to 10 hit points

Jenny's Younger Set

I am very pleased to announce that I have received two contributions this month, one from our almost regular Vincent Maker and the other from our resident adventure guru, Crocodile Jones, helping another member stuck in the middle of an adventure. Hope you all enjoy and are stimulated to try out your hand on TI-Writer or BASIC and send in a contribution. How about "What I did on my school holidays", computing wise of course!

Dear Jenny,
    I have received a letter from David Meldrum.

Dear Crocodile Jones,
    I am having trobles with adventure #12 (Golden Voyage). How do you tie the rope onto something to get back up the stair case on the rocky strand? Thank you
                    yours sincerely,
                    David Meldrum

Dear David,
    The commands are,"TIE ROPE" then you will be asked where. Type in,"TO STAL(IGMITE)" then type, "IN PIT".
                    Thanks for the question,
                    Crocodile Jones

Dear Jenny,
    Here is a program that will give a tune of "Annie's Song". Hope you like it,
                    Vincent Maker.

```
100 REM ANNIE'S SONG
110 REM AUTHOR UNKNOWN
120 REM ESPECIALLY FOR
130 REM LEANNE VOGELS.
140 REM PROGRAM BY
150 REM VINCENT MAKER
160 CALL CLEAR
170 PRINT """ANNIE'S SONG."""
180 FOR T=0 TO 7
190 PRINT
200 NEXT T
205 FOR VERSE=0 TO 2
210 CALL SOUND(250,784,0)
220 CALL SOUND(250,784,0)
230 CALL SOUND(250,698,0)
240 CALL SOUND(250,659,0)
250 CALL SOUND(497,784,0,262,3)
260 CALL SOUND(497,698,2,294,5)
270 CALL SOUND(497,330,0)
280 CALL SOUND(125,659,0)
290 CALL SOUND(125,659,0)
300 CALL SOUND(250,659,0,262,3)
310 CALL SOUND(250,698,0)
320 CALL SOUND(250,784,0)
330 CALL SOUND(250,587,0,392,3)
340 CALL SOUND(497,494,0)
350 CALL SOUND(497,494,0,497,3)
360 CALL SOUND(497,330,0)
370 CALL SOUND(125,587,0)
380 CALL SOUND(125,587,0)
390 CALL SOUND(250,587,0,294,3)
400 CALL SOUND(250,659,0,294,3)
410 CALL SOUND(250,698,0)
420 CALL SOUND(497,784,0,262,3)
430 CALL SOUND(497,698,0,247,3)
440 CALL SOUND(497,220,0)
450 CALL SOUND(125,659,0)
460 CALL SOUND(125,659,0)
470 CALL SOUND(250,659,0,262,3)
480 CALL SOUND(250,698,0,262,3)
490 CALL SOUND(250,784,0)
500 CALL SOUND(497,880,2,220,5)
510 CALL SOUND(497,880,2,698,5)
520 CALL SOUND(497,880,2,220,5)
530 CALL SOUND(125,784,0)
540 CALL SOUND(125,784,0)
550 CALL SOUND(250,784,0,294,3)
560 CALL SOUND(250,698,0,294,3)
570 CALL SOUND(250,659,0)
580 CALL SOUND(497,784,0,262,3)
590 CALL SOUND(497,880,0,294,3)
600 CALL SOUND(497,330,0)
610 CALL SOUND(125,784,0)
620 CALL SOUND(125,784,0)
630 CALL SOUND(250,784,0,262,3)
640 CALL SOUND(250,698,0,262,3)
650 CALL SOUND(250,659,0)
660 CALL SOUND(250,587,0,392,3)
670 CALL SOUND(250,494,0,392,3)
680 CALL SOUND(250,494,0,349,5)
690 CALL SOUND(250,494,0,330,3)
700 CALL SOUND(250,587,0)
710 CALL SOUND(250,587,0,294,3)
720 CALL SOUND(250,659,0,294,3)
730 CALL SOUND(250,698,0)
740 CALL SOUND(497,784,0,262,3)
750 CALL SOUND(497,698,0,247,3)
760 CALL SOUND(745.5,220,0)
770 CALL SOUND(250,659,0)
780 CALL SOUND(250,698,0,294,3)
790 CALL SOUND(375,784,0,294,5)
800 CALL SOUND(250,880,0)
810 CALL SOUND(497,831,0,196,5)
820 CALL SOUND(250,494,0,196,3)
830 CALL SOUND(125,523,0)
840 CALL SOUND(250,494,0)
850 CALL SOUND(497,784,0,196,3)
855 NEXT VERSE
860 CALL CLEAR
870 PRINT "IF YOU WANT TO HEAR THE SONGAGAIN PRESS 'A'
    IF NOT       PRESS ANY OTHER KEY."
880 CALL KEY(0,H,J)
890 IF J=0 THEN 880
900 IF H=65 THEN 100
910 END
```

File do everything between WHILE and ENDWHILE. If you do encounter the (EOF), or in this case the end of the database, then go to the next line after the ENDWHILE. The next line inside the loop will REPLACE the empty space in the variable TEMP with a bunch of blank spaces, the phrase " Exp. Date " and the club members Expiration Date (XP). The vertical lines "|" mean concatenate or stick together, the same as "&&" in Extended BASIC. So all three of those items are put into TEMP. Those items are then printed with the line PRINT TEMP. PRINT BLNK is the equivalent of "print a blank line". The next REPLACE takes FN (First Name), TRIMs off all the trailing blank spaces, sticks one space back (" "), attaches MI and another space (" "), puts LN (Last Name) on the end of that and sticks the whole mess into our variable TEMP. Now you see why TEMP had to hold up to 40 characters. The semicolon ";" at the end of these long lines is telling TI-Base that I could not get it all on 1 line and it should look for more on the next line down. TEMP is then printed as before. SA or Street Address is printed directly with no fancy stuff and the process is repeated for CT, ST and ZP. The blanks are thrown in for proper spacing to the next label. MOVE, moves the database to the next record and ENDWHILE sends you back to the WHILE statement to start over with the next name and address. The rest of the program is rather boring. When you finally run out of records the program jumps past all this to the CLOSE ALL. TNAMES is closed, everything you turned OFF is turned ON again, and the program is over.

    Important, next month I will work with larger programs, using the Funnelweb Programmer's Editor. The program on this page (LBLS1) is about the best you can write using the Modify Command Editor. I will also get into the use of printer control codes. Control codes can be imbedded in the program with the Funnelweb Editor, but not with the TI-Base Editor. I will cover some of the (Further Explanation Later)s and I will probably go over everything many times. In TI-Base there are several ways you can write a program to accomplish the same task. When I encounter that situation I will compare the previous program. This should give you more contact with TI-Base logical procedures.

Making your own Dictionaries

by Geoff Trott

I have been using the Dragonslayer Auto Spell-Check program for some time to provide a first check for typographical and spelling errors in articles. It works quite well from RAMdisk and from the Hard Disk but has always had some deficiencies in the dictionaries for my taste. What I am interested in finding are abbreviations (it's, let's and so on) as well as incorrect spelling as I do not believe that these help a reader's understanding and ease of reading. They certainly annoy me as I am reading. Unfortunately the dictionaries supplied with this good program contain a lot of these abbreviations and also do not contain some simple words like costs, bank and many plurals and tenses of verbs. This means that the list of words to be checked at the end of a run is longer than it needs to be, slowing down the process. Perhaps I should first explain how to use the program and how it works.

The program loads in two parts. The first part allows the colour of the screen to be set before loading the second part, which is larger and presumably contains the actual program. This is probably loaded into low memory expansion (>2000 to >3FFF) as the file whose spelling is to be checked is loaded next and can be as large as any file which can be loaded into TI-Writer and so probably fills high memory expansion (>A000 to >FFFF). Two things to note here. Funnelweb may be able to hold larger files than the Spell-Check can manage and any odd non-printing characters in the file can cause the program to crash. This last condition has ocurred to me with files from the BBS, which have an odd character inserted in them during transmission and also with a file produced with MyWord on a Geneve where a blank line produces a NULL character in the file whereas TI-Writer has a space character. Funnelweb happily read in the MyWord file and saved out a space character for the blank line. This is a good reason to use transliterates instead of special characters for printer control codes.

Once the file to be checked is loaded, the program searches the file for all the words which start with 'a' or 'A'. These, together with their position in the file, are stored in VDP RAM on the 'stack'. The time taken to do this depends on the size of the file and the number of words starting with the particular letter it finds. Having stacked all these words, the program starts reading the first dictionary file and checking the words in the stack against those words in the dictionary starting with 'A'. It removes all words from the stack which it finds in the dictionary and then goes on to repeat the procedure for words starting with the next letter of the alphabet. After running through all the letters of the alphabet in the first dictionary, the program starts reading the second dictionary. All the letters of the alphabet are in this dictionary too, but now the program does not have to read the file in the editor buffer as all the words to be checked are in the stack. The run through the second dictonary is much quicker as a consequence.

After using both dictionaries, the program asks if there is a user dictionary you wish to use for checking. I have never used this option but I assume that it runs much as the second main dictionary does. Once all dictionaries have been used you are required to look at each word still in the stack and decide what you want to do with it. This ranges from Viewing, Discarding, Changing or Adding the word to a user dictionary. It is also possible to go back to a previous word if you wish to change your mind. All in all a useful and easy to use program with very few problems aside from the dictionaries.

In order to change the dictionaries it is necessary to find out how they are constructed, what words are currently in the dictionaries and how to build some new dictionaries. The dictionary files are Display Variable 80 files but are full of non printing characters so they are not readable using TI-Writer. Looking at the files with Disk Utilities showed the following structure. The main dictionary files start with an asterisk '*'. Then each letter of the alphabet starts with the letter itself followed by an asterisk and then all the words starting with that letter in a compressed form, finishing with the '@' symbol. If there are no words to be included starting with this character then a NULL character is included between the <letter>* and the '@' symbol.

The words are compressed by using a byte to say how many letters of the current word are used in the next word followed by the letters of the rest of the next word. This means that the plural of a word would take two extra bytes, a byte count followed by 'S', no matter how long the word is. The byte count is identified by having 128 added to the count which sets its most significant bit and makes it larger than any ASCII code. For example, consider the first few words starting with C. By the way, all words in the dictionaries are upper case but case does not matter when checking a word. If the word is not in the dictionary, then each version of the word with different mixtures of case are retained on the stack. The first 3 words are CAB, CABAL, CABALISTIC. The file would contain:
@
C*
<129>AB<131>AL<133>ISTIC<...
where the numbers in <> represent the decimal values of the bytes and the '@' is the end of the 'B' words.

Here is the listing of the Extended BASIC program which reads a dictionary file and outputs the words starting with a particular letter to a file whose name is that letter.

```
100 INPUT "DICTIONARY FILE ?":A$
110 OPEN #1:A$,DISPLAY ,VARIABLE 80,INPUT :: S$="@"
120 INPUT "FIRST LETTER ?":L$ :: OPEN
    #2:"DSK1."&L$,OUTPUT,DISPLAY ,VARIABLE 80 :: IF
    SEG$(S$,1,1)>L$ THEN CLOSE #1 :: GOTO 110
130 IF EOF(1)THEN 100
140 LINPUT #1:A$ :: LA=LEN(A$)
150 IF LA=0 THEN 130
170 IF SEG$(A$,2,1)="*" THEN S$=SEG$(A$,1,1):: GOTO 130
180 IF SEG$(S$,1,1)<L$ THEN 130
185 IF A$="@" THEN PRINT #2:S$ :: CLOSE #2 :: GOTO 120
190 FOR I=1 TO LA :: B=ASC(SEG$(A$,I,1)):: IF B>127 THEN
    B=B-128 :: PRINT #2:S$ :: S$=SEG$(S$,1,B)ELSE
    S$=S$&CHR$(B)
200 NEXT I
210 GOTO 130
220 STOP
```

The line doing all the work is line 190, where the line read in is examined one character at a time and when the character code is greater than 127 this indicates the start of a new word and the end of the current word. Until these characters are found, characters are appended to the current word. When a character with a code greater than 127 is found, the current word is output and then the next word is started by using the first B characters of the current word.

Once the dictionaries have been unloaded into separate files for each letter, they then can be edited to add, change and remove words. The files contain words starting with the same letter, one word to a line. In order to add the plurals of nouns and tenses of verbs, I decided to use a similar approach to the dictionaries in order to save space. Because it is being done in an editor (programmer's editor), I use a space to delimit the end of a word and the first word on the line is the common part of the word. That means that the simple plural only requires a space followed by an S, like:
CATALOG S
but other ones will require two entries because of the change in the word.
CITIES
CITY
All the words must be in alphabetical order to obtain the best packing and for the dictionaries to work correctly. Tenses of verbs require the endings ED, ING

and S · with sometimes ER. This can lead to similar problems as the plurals of nouns. However, considerable savings in file size and typing can be obtained using this approach.

Having edited the files (which is the long job and is not yet complete) it is now necessary to put the dictionaries back together again. To this end I have written a c99 program which does the following jobs:
  read the 26 word files one at a time;
  extract each word from the format mentioned in the previous paragraph;
  compress the word using the format required by the dictionaries;
  append the data to the current record, keeping track of the number of characters up to the 80 allowed in display variable 80 files; and
  output the records to the file in such a way as to fill up each sector as much as possible.

This last task finishes the current record at a length which will fit in the available space on the current sector. This causes the resulting dictionary file to be currently 10% smaller than the original ones with many more words in the dictionaries (mainly plurals and tenses of verbs for the letters A, B and C). The original dictionaries did not have words for all the letters of the alphabet in both dictionary files. They were split evenly between the two files so that both files were about the same size. Words starting with C, P and S were in both files. One consequence of this is that all single letters are effectively in the dictionary except for C, P and S. I have arranged my new dictionary files to have in one file, words of all the letters (344 sectors long) while the second file only has the extra words starting with C, P and S (72 sectors long).

Here is the listing of the c99 program:

```
#include "DSK7.STDIO"
#include "DSK7.STRINGI"
int unit1,unit2,len;
char c,buff[81],at[2]="@",ffcr[3]="\014\015",
    al[27]="ABCDEFGHIJKLMNOPQRSTUVWXYZ",nc[4]="\0";
char fname[21]="DSK2.\0",sl[81],*p3;
char word[81],wore[81];
int sect,line;
main()
{ char *p,*pb,*pw,*ps;
  int ls,ln,lb,n;
  unit1=fopen("DSK1.DICTX","w");
  fwrite("* ",1,unit1);
  sect=253;line=80;
  puts("processing files\n");
  for (len=0;len<26;++len) {
    strcpy(sl,fname);
    stncat(sl,&al[len],1);
    puts(sl); putchar(10);
    unit2=fopen(sl,"r");
    wore[0]=al[len];
    wore[1]=NULL;
    p3=sl;
    fgets(buff,80,unit2);
    lb=strlen(buff);
    sl[0]=buff[0];
    sl[1]='*';
    fwrite(sl,2,unit1);
    sect=sect-3;
    if (sect<0) sect=254;
    if (lb == 1) {
      fputs(nc,unit1);
      sect=sect-1;
      if (sect<0) sect=254;
    }
    while (lb>1) {
      pb=buff;
      pw=word;
      ps=0;
      for (ls=0;ls <= lb;ls++) {
        if (*pb == ' ') {
          *pw = NULL;
          putwd();
          if (ps == 0) ps = pw;
          pw=ps;
```

```
          pb++;
          if (*pb == ' ') {
            pb++;
            ls++;
          }
        }
        else *pw++ = *pb++;
      }
      putwd();
      if (fgets(buff,80,unit2))
        lb=strlen(buff);
      else { lb=0;
        fwrite(sl,80-line,unit1);
        sect=sect-81+line;
        line=80;
        p3=sl;
/*      putchar(48+sect/10);putchar(48+sect%10); */
      }
    }
    fclose(unit2);
    fwrite(at,1,unit1);sect=sect-2;
    if (sect<0) sect=254;
  }
  fwrite(ffcr,2,unit1);
  fwrite(ffcr,2,unit1);
  fclose(unit1);
}
putwd()
{
  char *p1,*p2;
  int n,ls;
  for (p1=wore,p2=word,n=0;(*p1==*p2) ;p1++,p2++,n++) ;
  if (*p1>*p2) {
    puts(wore);
    puts(" and ");
    puts(word);
    puts(" are out of order\n");
  }
  *p3=0;ls=strlen(p2)+1;
  if ((line<ls) | (sect<81-line+ls)) {
    fwrite(sl,80-line,unit1);
    sect=sect-81+line;
    line=80;
    p3=sl;
/*  putchar(48+sect/10);putchar(48+sect%10); */
    if (sect<ls) sect=254;
  }
  *p3++=128+n;strcpy(p3,p2);line=line-ls;
  p3=p3+ls-1;
  strcpy(wore,word);
}
```

Next month I will talk about this program and the procedures used in the program for manipulating strings. These were part of the original c99 release disk, but needed some debugging. In the process I will explain some of the C language syntax.   o

Multi-purpose Problem Solving Procedures

This flowchart (modified for mature audiences) details the latest high-tech methodology in handling the ups and downs of multi-purpose problem solving.
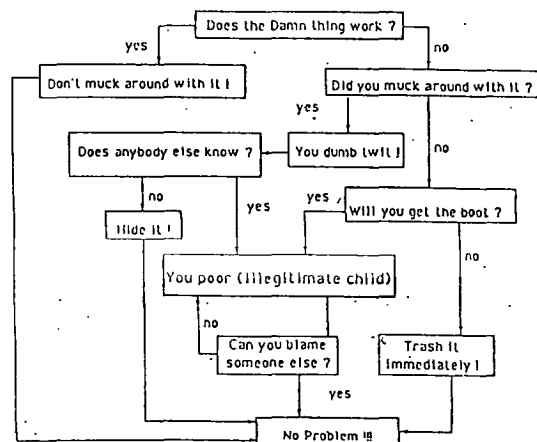
Tips from the Tigercub #33

by Jim Peterson, Tigercub Software, USA

Tigercub Software
156 Collingwood Ave
Columbus, OH 43213

Distributed by Tigercub Software to TI99/4A Users Groups for promotional purposes and in exchange for their newsletters. May be reprinted by non-profit users groups, with credit to Tigercub Software.

Over 130 original programs in BASIC and Extended BASIC, available on cassette or disk, only $3 each plus $1.50 per order for postage and packing. Entertainment, education, programmer's utilities.

Descriptive catalog $1, deductable from your first order.

Tips from The Tigercub, a full disk containing the complete contents of this newsletter numbers 1 through 14, 50 original programs and files, just $15 postpaid. Tips from the Tigercub volume 2, another disk full, complete contents of numbers 15 through 24, over 60 files and programs, also just $15 postpaid. Or, both for $27 postpaid.

Nuts & Bolts (No. 1), a full disk of 100 Extended BASIC utility subprograms in merge format, ready to merge into your own programs. Plus the Tigercub Menuloader, a tutorial on using subprograms, and 5 pages of documentation with an example of the use of each subprogram. All for just $19.95 postpaid.

Nuts & Bolts No. 2, another full disk of 108 utility subprograms in merge format, all new and fully compatible with the last, and with 10 pages of documentation and examples. Also $19.95 postpaid, or both Nuts & Bolts disks for $37 postpaid.

Tigercub Full Disk Collections, just $12 postpaid! Each of these contains either 5 or 6 of my regular $3 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; my own programs on these disks are greatly discounted from their usual price, and the public domain is a free bonus!

```
TIGERCUB'S BEST              PROGRAMMING TUTOR
PROGRAMMER'S UTILITIES  BRAIN GAMES   BRAIN TEASERS
BRAIN BUSTERS!     MANEUVERING GAMES     ACTION GAMES
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
```

For descriptions of these send a dollar for my catalog!

I found a bug in Nuts & Bolts #2 which prevents using HIGHCHAR after HEAVYCHAR. To fix it, remove the write protect tab and:

```
MERGE DSK1.HEAVYCHAR
RES 21008,1
SAVE DSK1.HEAVYCHAR,MERGE
```

Replace write protect tab.

While they last, and the supply is limited, I will sell a single Texas Instruments cassette interface cable for $2 with any order for cassette software.

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

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

```
100 CALL CLEAR :: CALL SCREEN(16):: FOR SET=2 TO 9 ::
    CALL COLOR(SET,5,16):: NEXT SET :: ON WARNING NEXT
    :: RANDOMIZE
110 DISPLAY AT(21,1)ERASE ALL:">>>TIGERCUB SORT
    WATCHER<<<": :"Wait, please - generating":"random
    array...." :: DIM A$(101),B$(101),ST(25,2)
```

```
120 FOR J=1 TO 100 :: FOR L=1 TO 3 ::
    B$(J)=B$(J)&CHR$(INT(26*RND+65)):: NEXT L :: X=J ::
    A$(X)=B$(X):: GOSUB 1020 :: NEXT J
130 DISPLAY AT(3,1)ERASE ALL:"(1) BUBBLE SORT": :"(2)
    SHAKER SORT": :"(3) SWAP SORT": :"(4) SHUTTLE SORT":
    :"(5) EASY SORT"
140 DISPLAY AT(13,1):"(6) QUICK SORT": :"(7) RESORT
    SORT": :"(8) SHELL SORT": :"(9) RESERVED": :"Type
    number of choice"
150 ACCEPT AT(21,23)VALIDATE(DIGIT)SIZE(2)BEEP:K :: IF
    K<1 OR K>10 THEN 150
160 DISPLAY AT(24,1):"Size of array? (10-100)" ::
    ACCEPT AT(24,25)VALIDATE(DIGIT)SIZE(3):G :: IF G<1
    OR G>100 THEN 160
170 ON K GOSUB 230,300,430,500,550,650,850,910,25000 ::
    DISPLAY AT(22,1):W;"SWAPS":C;"COMPARISONS" :: C,W=0
180 DISPLAY AT(24,1):"Choose (1)Menu or (2)Resort" ::
    ACCEPT AT(24,7)VALIDATE("12")SIZE(1):Q :: IF Q=1
    THEN 130
190 DISPLAY AT(24,1):"Change which position? 0" ::
    ACCEPT AT(24,24)VALIDATE(DIGIT)SIZE(-3):P :: IF P=0
    THEN 210 ELSE IF P<1 OR P>G THEN 190
200 DISPLAY AT(24,1):"Change to?" ::
    ACCEPT AT(24,12)SIZE(3):A$(P):: X=P :: GOSUB 1020 ::
    GOTO 190
210 DISPLAY AT(22,1):" ":" " :: GOSUB 1010 :: N=G :: ON
    K GOSUB 240,310,440,510,560,660,860,930,25010 ::
    DISPLAY AT(22,1):W;"SWAPS":C;"COMPARISONS" :: C,W=0
    :: GOTO 180
220 REM *BUBBLESORT*
230 CALL CLEAR :: GOSUB 980
240 FOR J=2 TO N :: C=C+1 :: IF A$(J)>=A$(J-1)THEN 260
250 T$=A$(J):: GOSUB 1050 :: A$(J)=A$(J-1):: X=J ::
    GOSUB 1020 :: A$(J-1)=T$ :: X=J-1 :: GOSUB 1020 ::
    W=W+1 :: F=1
260 NEXT J :: C=C+1 :: IF F=0 THEN 280
270 W=W+1 :: F=0 :: W=W+1 :: N=N-1 :: GOTO 240
280 RETURN
290 REM *SHAKERSORT*
300 CALL CLEAR :: GOSUB 980
310 W=W+1 :: L=1 :: W=W+1 :: R=N
320 W=W+1 :: F=0 :: FOR J=L TO R-1 :: C=C+1 :: IF
    A$(J)<=A$(J+1)THEN 340
330 T$=A$(J):: GOSUB 1050 :: A$(J)=A$(J+1):: X=J ::
    GOSUB 1020 :: A$(J+1)=T$ :: X=J+1 :: GOSUB 1020 ::
    W=W+1 :: F=1
340 NEXT J :: C=C+1 :: IF F=0 THEN 410
350 W=W+1 :: R=R-1 :: C=C+1 :: IF R=L THEN 410
360 W=W+1 :: F=0 :: FOR J=R TO L+1 STEP -1 :: C=C+1 ::
    IF A$(J)>=A$(J-1)THEN 380
370 T$=A$(J):: GOSUB 1050 :: A$(J)=A$(J-1):: X=J ::
    GOSUB 1020 :: A$(J-1)=T$ :: X=J-1 :: GOSUB 1020 ::
    W=W+1 :: F=1
380 NEXT J :: C=C+1 :: IF F=0 THEN 410
390 W=W+1 :: L=L+1 :: C=C+1 :: IF L=R THEN 410
400 GOTO 320
410 RETURN
420 REM *SWAPSORT*
430 CALL CLEAR :: GOSUB 980
440 FOR J=1 TO N-1 :: W=W+1 :: R=J :: FOR JJ=J+1 TO N ::
    C=C+1 :: IF A$(R)<=A$(JJ)THEN 460
450 W=W+1 :: R=JJ
460 NEXT JJ :: C=C+1 :: IF R=J THEN 480
470 T$=A$(J):: GOSUB 1050 :: A$(J)=A$(R):: X=J :: GOSUB
    1020 :: A$(R)=T$ :: X=R :: GOSUB 1020
480 NEXT J :: RETURN
490 REM ***SHUTTLE SORT*****
500 CALL CLEAR :: GOSUB 980
510 FOR J=1 TO N-1 :: FOR JJ=J TO 1 STEP -1 :: C=C+1 ::
    IF A$(JJ)<=A$(JJ+1)THEN 530 :: T$=A$(JJ):: GOSUB
    1050 :: A$(JJ)=A$(JJ+1):: X=JJ :: GOSUB 1020
520 A$(JJ+1)=T$ :: X=JJ+1 :: GOSUB 1020 :: NEXT JJ
530 NEXT J :: RETURN
540 REM ****EASY SORT******
550 CALL CLEAR :: GOSUB 980
560 W=W+1 :: D=1
570 W=W+1 :: D=2*D :: C=C+1 :: IF D<=N THEN 570
580 W=W+1 :: D=INT(D/2):: C=C+1 :: IF D=0 THEN 630
590 FOR J=1 TO N-D :: W=W+1 :: Y=J
600 W=W+1 :: Z=Y+D :: C=C+1 :: IF A$(Y)<=A$(Z)THEN 620
    :: T$=A$(Y):: GOSUB 1050 :: A$(Y)=A$(Z):: X=Y ::
    GOSUB 1020 :: A$(Z)=T$ :: X=Z :: GOSUB 1020
610 W=W+1 :: Y=Y-D :: C=C+1 :: IF Y>0 THEN 600
620 NEXT J :: GOTO 580
630 RETURN
```

```
640 REM *QUICKSORT*
650 CALL CLEAR :: GOSUB 980
660 W=W+1 :: L=1 :: W=W+1 :: R=N :: W=W+1 :: T=0
670 T$=A$(INT((L+R)/2)):: GOSUB 1050 :: W=W+1 :: J=L ::
    W=W+1 :: JJ=R
680 C=C+1 :: IF A$(J)>=T$ THEN 710
690 W=W+1 :: J=J+1
700 GOTO 680
710 C=C+1 :: IF A$(JJ)<=T$ THEN 730
720 W=W+1 :: JJ=JJ-1 :: GOTO 710
730 C=C+1 :: IF A$(J)<>A$(JJ) THEN 760
740 C=C+1 :: IF J>=JJ THEN 760
750 W=W+1 :: J=J+1 :: GOTO 730
760 C=C+1 :: IF J>=JJ THEN 780
770 W=W+1 :: H$=A$(J):: A$(J)=A$(JJ):: X=J :: GOSUB 1020
    :: A$(JJ)=H$ :: X=JJ :: GOSUB 1020 :: GOTO 680
780 W=W+1 :: J=J+1 :: W=W+1 :: JJ=JJ-1 :: C=C+1 :: IF
    J>=R THEN 800
790 W=W+1 :: T=T+1 :: W=W+1 :: ST(T,0)=J :: W=W+1 ::
    ST(T,1)=R
800 W=W+1 :: R=JJ :: C=C+1 :: IF L<R THEN 670
810 C=C+1 :: IF T=0 THEN 830
820 W=W+1 :: L=ST(T,0):: W=W+1 :: R=ST(T,1):: W=W+1 ::
    T=T-1 :: GOTO 670
830 RETURN
840 REM ***RESORT SORT******
850 CALL CLEAR :: GOSUB 980
860 FOR J=2 TO N :: C=C+1 :: IF A$(J)>=A$(J-1)THEN 900
870 T$=A$(J):: GOSUB 1050 :: FOR L=J-1 TO 1 STEP -1 ::
    A$(L+1)=A$(L):: X=L+1 :: GOSUB 1020
880 C=C+1 :: IF A$(L-1)>=T$ THEN 890 :: A$(L)=T$ :: X=L
    :: GOSUB 1020 :: GOTO 900
890 NEXT L
900 NEXT J :: RETURN
910 REM *SHELLSORT*
920 CALL CLEAR :: GOSUB 980
930 W=W+1 :: M=N
940 W=W+1 :: M=INT(M/3)+1
950 FOR J=1 TO N-M :: FOR JJ=J TO 1 STEP -M :: C=C+1 ::
    IF A$(JJ)<=A$(JJ+M)THEN 970 :: T$=A$(JJ):: GOSUB
    1050
960 A$(JJ)=A$(JJ+M):: X=JJ :: GOSUB 1020 :: A$(JJ+M)=T$
    :: X=JJ+M :: GOSUB 1020 :: NEXT JJ
970 NEXT J :: C=C+1 :: IF M>1 THEN 940 :: RETURN
980 REM *RENEW ARRAY*
990 FOR J=1 TO G :: A$(J)=B$(J):: X=J :: M$=A$(J)::
    GOSUB 1020
1000 NEXT J :: N=G
1010 DISPLAY AT(24,1):"A to abort    P to pause" ::
     RETURN
1020 RR=X
1030 IF RR>20 THEN RR=RR-20 :: GOTO 1030
1040 CC=1-5*(X>20)-5*(X>40)-5*(X>60)-5*(X>80) :: DISPLAY
     AT(RR,CC):A$(X);:: W=W+1 :: GOSUB 1060 :: RETURN
1050 DISPLAY AT(22,14):"T$=";T$ :: W=W+1 :: GOSUB 1060
     :: RETURN
1060 CALL KEY(3,K1,SS):: IF SS=0 THEN 1090
1070 IF K1=65 THEN 130
1080 CALL KEY(3,K2,SS):: IF SS<1 THEN 1080
1090 RETURN
```

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

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

```
100 !BASKET WEAVING by Jim Peterson
110 CALL CLEAR :: W=11 :: T=2 ::
    CH$="A5A5A5A5A5A5A5A5FF00FF0000FF00FF" ::
    CALL CHAR(142,CH$):: CALL COLOR(14,2,W,13,2,W)::
    CALL SCREEN(W)
120 CALL HCHAR(1,1,143,768):: CALL CHAR(134,CH$)::
    CH=142
130 FOR C=1 TO 31 STEP T :: FOR R=1 TO 23 STEP T ::
    CALL HCHAR(R,C,CH):: NEXT R :: FOR R=24 TO 2 STEP -T
    :: CALL HCHAR(R,C+1,CH):: NEXT R :: NEXT C
140 CH=ABS(135*(CH=142)+143*(CH=134)):: RANDOMIZE ::
    T=INT(3*RND+2)
150 FOR R=1 TO 23 STEP T :: FOR C=2 TO 32 STEP T ::
    CALL HCHAR(R,C,CH):: NEXT C
```

```
160 FOR C=31 TO 1 STEP -T :: CALL HCHAR(R+1,C,CH):: NEXT
    C :: NEXT R :: CH=CH-1 :: W=INT(14*RND+3)::
    T=INT(3*RND+2)
170 IF CH=134 THEN CALL COLOR(13,2,W):: GOTO 130 ELSE
    CALL COLOR(14,2,W):: GOTO 130
```

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

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

```
1 !TO PRINT A HANDY REFERENCE CHART OF ASCII TO HEX CODE
  - MODIFIED FROM READING-BERKS AUG 85
90 OPEN #1:"PIO" :: PRINT #1:CHR$(27);CHR$(77);CHR$(5)
100 FOR X=32 TO 63 :: FOR Y=X TO X+64 STEP 32 ::
    CALL CHARPAT(Y,Y$):: PRINT #1:Y;" ";CHR$(Y);"
    ";Y$;:: NEXT Y :: PRINT #1:"" :: NEXT X
```

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

The User Group newsletters are full of good editorials, reminding people that they had better pay for their freeware or there will not be anymore. I totally agree with that, but I cannot help thinking that if there had been as much emphasis on paying for commercial software instead of pirating it, there would still be a lot more good programmers supporting the TI99/4A!

Memory full! Jim Peterson   o

6.9 Character definition.
    Characters may be redefined similar to the BASIC CALL CHAR sequence. TEXT80 uses the control sequence CHR$(27);CHR$(38);CHR$(0);CHR$(n);"DEFSTR$" for character definition. DEFSTR$ is as described by the BASIC manual under the CALL CHAR subprogram. The value of n is the ASCII code of the redefined character. Use the control sequence CHR$(27);CHR$(38);CHR$(1) to reset all redefined characters to their original ASCII definitions. This control code sequence is equivalent to the BASIC CALL CHARSET subprogram. At this time, dependent on the dip switch setting of bank 2 switch 3, either the USA ASCII character set or the German Character set is reloaded. The code in each case contains 255 characters. These are between ASCII 32 and 127, the same as those used by TI-Writer and the rest up to ASCII 255 as a PC.

6.10 Accepting an existing screen content with INPUT.
    At OPENing of TEXT80 an input prompt will be cleared from the screen. If text already displayed on the screen wished to be passed on to INPUT then use the control sequence CHR$(27);CHR$(105). The sequence CHR$(27);CHR$(106) will restore the starting status.

7. Redefining the screen colours.

    The TI99/4A displays 16 colours on the screen. These may be separately defined by the 80 column card, i.e. we can redefine colour 1, which is normally black to be white. The redefined colours remain until the computer is turned off or reset by hardware. One should use this option with due care; one may quickly finish up with an invisible screen (black on black etc.)!

String Manipulation

by Craig Sheehan

In BASIC or Extended BASIC, there are two different *types of* variables: numeric and strings. Whilst many programmers make extensive use of numeric variables, strings are largely ignored, used for only storing complete words or sentences. But the string operators supplied allow much more varied and powerful use than this.

For this tutorial, we will only concern ourselves with four of eight (nine in Extended BASIC) string operators. The first is SEG$. SEG$ is used for quite literally chopping up strings. It has three arguments: the first is the string to be chopped, the second the first character required and the third contains the number of characters required. A few examples should illustrate this more clearly.

```
(1) PRINT SEG$("TIsHUG",3,2)    prints  sH
(2) PRINT SEG$("A word",3,4)    prints  word
(3) PRINT SEG$("Two words",1,3) prints  Two
```

The second function is POS, which finds occurrences of a string within another string, returning the character at which the two strings were matched. There are three arguments: the first is the string to be searched, the second the string you wish to find in the first, and the third holds the character number that you wish to commence the search from. Zero is returned if the string can not be matched after the starting character.
For example:

```
(1) PRINT POS("A TITLE","TITLE",1) prints  3
(2) PRINT POS("A:BC:DEF:",":",3)   prints  5
(3) PRINT POS("NOT HERE","N",3)    prints  0
```

LEN returns the total number of characters in a string. Its single argument is the string that you wish to know the length of. Finally, the last operator is concatenation. Placing an ampersand (&) between two strings has the effect of joining the two strings together to form a single string.
For example:

```
(1) PRINT LEN("HOW MANY CHARS") prints  14
(2) PRINT "TI-99"&"/4A"         prints  TI-99/4A
(3) PRINT SEG$("HOHUM",3,3)&SEG$("HOHUM",1,2)
                                prints  HUMHO
```

So far, I have only described what can be found in the manuals (see references), but using combinations of these four operators, it is possible to do some complex tasks. Consider the task of locating spaces within a string. One way of tackling this to examine each character in a string using SEG$ and comparing it to the space character. The program below prints the character position of each space in the string.

```
100 A$="Where are the spaces?"
110 FOR CH=1 TO LEN(A$)
120 IF SEG$(A$,CH,1)=" " THEN PRINT CH
130 NEXT CH
```

The output to this program is 6, 10 and 14. If you cannot see how the program works, stop for a moment and trace the operation of the program by hand. Another equally valid way of achieving the same result is to use the POS function to find each space. When we find a space, we start our next search from the following character. When a value of zero is returned, then there are no more spaces in the string.

```
100 A$="Where are the spaces?"
110 P=0
120 P=POS(A$," ",P+1)
130 IF P<>0 THEN PRINT P :: GOTO 120
```

Again the output will be 6, 10 and 14. Once we can locate spaces within a string, we can find entire words.

If we define a word to be any set of characters separated by spaces, then if we know where one space is, and where the following space is, then SEG$ can be used to chop that word out of the string. However, using this method may cause the first and last words of a string to missed, since they may not have a space in front , or after it respectively. This can be remedied by simply adding a space to the beginning and end of the string. All of these ideas are combined in program below. It would be advisable to run through this program by hand to get a feel for what is happening.

```
100 A$="Print each word on a separate line"
110 B$=" "&A$&" "
120 LASTSPACE=1
130 NEXTSPACE=POS(B$," ",LASTSPACE+1)
140 IF NEXTSPACE<>0 THEN PRINT SEG$(B$,LASTSPACE+1,
    NEXTSPACE-LASTSPACE-1):: LASTSPACE=NEXTSPACE ::
    GOTO 130
```

The output from this program will be the words Print, each, word, on, a, separate, line; printed on subsequent lines.

Isolating separate words is only one use of strings. Another use is in altering text. For example, say you have a file containing words that have american "IZE" instead of "ISE". Although using TI-Writer's replace string would be faster, a BASIC program could be written that achieves the same result. Basically, we must find each occurrence of the string we want to replace, OLD$, and then replace it with NEW$ by chopping up the original string into two halves, excluding OLD$, and inserting NEW$ in the middle. This is done using SEG$ and the concatenator.

```
100 A$="RANDOMIZE SUMMARIZE"
110 OLD$="IZE"
120 NEW$="ISE"
130 CH=1
140 P=POS(A$,OLD$,CH)
150 IF P<>0 THEN A$=SEG$(A$,1,P-1)&NEW$&SEG$(A$,
    P+LEN(OLD$),LEN(A$)-P-LEN(OLD$)+1)::
    CH=P+LEN(NEW$):: GOTO 140
160 PRINT A$
```

This program's output will be RANDOMISE SUMMARISE. If you do run through this program by hand, you will notice that the next occurrence of OLD$ is searched for from the character after the substitution . The reason for this is if NEW$ contains OLD$. For example, if OLD$="THE" and NEW$="THEIR", you do not want the "THE" from NEW$ to replaced again as this would result in a program that would eventually crash when the string had grown to its limit of 255 characters.

For final example demonstrating the versatility of strings, consider the task creating a draw for a knock out tournament, consisting of sixteen competitors, randomly. To do this, I will place the characters A to P in a string, each character corresponding to a player. Select a character from the string at random, add that character to a "draw" string, remove the character from the original string and then repeat the above until no characters are left in the original string. Using this method ensures that there will be no duplication of players. Finally, we print out the draw using the "draw" string. The program that carries out the above algorithm is shown below.

```
100 A$="ABCDEFGHIJKLMNOP"
110 B$=""
120 RANDOMIZE
130 FOR DRAW=16 TO 1 STEP -1
140 P=INT(RND*DRAW)+1
150 B$=B$&SEG$(A$,P,1)
160 A$=SEG$(A$,1,P-1)&SEG$(A$,P+1,DRAW-P)
170 NEXT DRAW
180 FOR T=1 TO 15 STEP 2
190 PRINT SEG$(B$,T,1);" vs ";SEG$(B$,T+1,1)
200 NEXT T
```

Again, it is advisable to run through this program by hand in order to understand it fully.

Special Function Key Emulation

by Lou Amadio

Encouraged by Derek Wilkinson's effort at interfacing an IBM keyboard to the TI99/4A and the special dual contact keys that he had to modify, I experimented with a way of providing CTRL[U] and FCTN[R] using just one keystroke for each. As you may recall, these functions are used in TI-Writer for the following.

CTRL[U] changes the cursor to Special Character Mode. This mode allows the generation of ASCII characters below 32 - that is control characters.

FCTN[R] generates the Escape character (>1b in Special Character Mode). Escape, in conjunction with other keyboard characters, is used to invoke the special features of your printer such as double strike or emphasized print, etc.

I initially tried to emulate a dual key action by wiring the normally open contacts of a double pole relay across the appropriate key switches on the back of the keyboard. A small momentary action push button switch, in series with a resistor, was used to power the relay from the 22 volt console supply.

Operation was, however, unreliable, as I found that the computer would sometimes generate the wrong character or just intermittently lock up.

In order to introduce a time delay into the second "keystroke", I decided to use the second set of contacts on the relay to power another relay. The normally open contacts of this second relay were then wired across the second key switch. This process proved to be very reliable, provided that a diode and capacitor were soldered across the relay coils to prevent the back emf from getting into the computer's power supply.

I mounted the 4 miniature relays required for the two functions on a small piece of Veroboard. The board was located above the console power supply and powered via a suitable resistor. The relays that I used were rated at 10 mA at 12 volts. Thus a 470 ohm resistor was used between the 22 volt supply and the push button switch.

The miniature push-button switches were mounted on the upper console shell. The CTRL[U] switch was located to the left of the "A" key and the FCTN[R] switch was located to the right of the "=" key. I used a red push button for CTRL[U] and a black push button for FCTN[R] as these closely resemble the coloured spots on these keys.
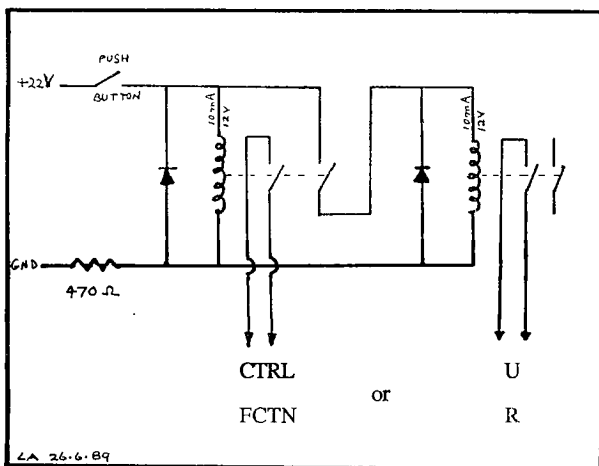
Notes:

1) The FCTN[R] switch will generate a "[" character when the keyboard is in normal mode.

2) Using the above technique, it should be possible to emulate the cursor control keys. The switches could be mounted on the module port bay next to the <ENTER> key. However, in this instance, the number of relays required is too cumbersome, so a more elegant (electronic) solution is required.   o

HEALING2 - 1 to 20 hit points
HEALING3 - 1 to 32 hit points (not available in Version 1.1)

STEALTH: Version 1.1 only. When this potion is consumed by any party member, an aura of invisibility will envelop the entire party. This allows the party to sneak by monsters undetected. Ideal for Dark swamps. NOTE: this potion wears out with time.

SLOW: Version 1.1 only. This is Ranger spell #2 and Wizard's spell #3. Casting this spell reduces all monster attacks per turn to 1. This remains in effect until a monster cast a DISPEL MAGIC spell. Some monsters can also cast the SLOW spell. The party has to cast a DISPEL MAGIC spell to negate its possibly disastrous consequences.

Casting Spells

Magic spells can only be used in combat. To cast a spell, press "4" on the individual combat spell option list. Legends will prompt "CAST WHICH SPELL". At this point press the number of the spell you wish to cast. You will not be able to cast the spell if you have not learned it yet (that is you are not at a high enough level) or if you do not have enough magic points. If either is the case, press the number "exit cast phase" or choose another spell.

Spell Magic points.

Class	No.	Sp.	Level	Type	Req'd
Wizard	1	F	1	A	2
	2	DISPEL MAGIC	2	A	3
	3	STRENGTH	3	P	3
Version 1.1	3	SLOW	3	P	3
	4	FIRESTORM2	4	A	4
	5	ROT ARMOUR	5	A	5
	6	LIFESTEAL	6	A	6
	7	FIRESTORM3	7	A	6
	8	RESIST MAGIC	8	P	7
	9	exit cast phase			
Cleric	1	HEALING1	1	P	2
	2	PROTECTION2	2	P	3
	3	HEALING2	3	P	4
	4	DISPEL MAGIC	4	A	4
	5	TURN UNDEAD	5	A	5
	6	WEAKNESS	6	A	6
	7	HEALING3	7	P	.
Version 1.1	7	STEALTH	7	P	6
	8	FIRESTORM2	8	A	5
	9	exit cast phase			
Ranger	1	HEALING1	1	A	2
	2	PROTECTION1	2	P	2
Version 1.1	2	SLOW	2	P	2
	3	FIRESTORM1	3	A	3
	4	HEALING2	4	P	4
	5	FIRESTORM2	5	A	5
	6	PROTECTION2	6	P	5
	7	exit cast phase			
Fighter		Press CAST will "evaluate" the monsters which displays information on monster armour, class, strength and damage capacity.			

Advanced Diagnostics Review

Author unknown

A product of Miller's Graphics
1475 W. Cypress Ave.
San Dimas, CA 91773 — Price: $19.95

Advanced Diagnostics is a well done program that is typical of the products that we have come to expect from Miller's Graphics. The program, written by Steve Mildon, executes flawlessly, provides a few important hardware diagnostics, several diskette diagnostics, and contains some nice diskette utilities.

Various parts of the program are executed by issuing commands, and therein lies a powerful feature of the program. The program can execute a single command or multiple commands that are entered into a two line portion of the display screen, or it can execute a large sequence of commands that have been placed in a command file. The commands, which are two letter mnemonics or two complete American words, can be placed in command files by any DIS/VAR 80 text editor, such as TI-Writer. Several good examples of command file usage are included with the program. One example command file prompts for ten diskettes to be inserted into a diskette drive (one at a time), names each diskette, and proceeds to format each.

Well, so much for preliminaries. As you can probably tell, I like the product and I feel that I have received my money's worth. But it is probably impossible for a person who has purchased a product with their own money to give an unbiased review, so I shall try to fight this bias as I proceed.

I will begin with hardware considerations. The program requires a diskette drive, memory expansion, and software (firmware?) that can load and run assembly programs. Both TI and CorComp disk drive controllers are supported. Miller's Graphics states that the program requires either the Extended BASIC cartridge, the Editor Assembler cartridge, or the Mini Memory cartridge. But my copy of the program runs fine from the file utilities menu of the Disk Manager program that is part of the CorComp disk drive controller software.

Two hardware diagnostics commands that I found useful are Check Memory and Motor Speed. The Check Memory command performs a thorough memory test on CPU Scratch pad RAM, VDP RAM, Memory expansion RAM, and Mini Memory RAM (if present). Program progress is monitored on the video display as the test proceeds, so one is not left to wonder about "just what in the world is going on" at this point. The current area of memory being tested (VDP RAM, Memory Expansion RAM, etc.), the test being performed (memory refresh or bit shift), and the address range of the memory being tested is reported. If an error is found, the memory location of the error is reported, helping one to isolate a bad memory chip. But more data is needed to turn this into information that can be acted upon. Suppose that Advanced Diagnostics finds a memory location that tests bad, and reports "Bad Value at >xxxx". How does one locate the chip that is bad? A good question. One that I have no answer for. It seems to me that it would have been useful, and relatively easy, for the program to report "Check U108 on the System Board", or something to that effect. Supporting information such as component location diagrams might also be included in the user manual. Alas, at the present time this information is simply not there, and I feel that this is a weak point of the product, one which should be easily fixed.

The Motor Speed command will check the RPM of the selected diskette drive and report it on a nice colourful bar graph. The display has a moving pointer that is updated according to a user selectable sample rate parameter. If a drive motor tests good, this pointer will vary around 300 RPM in a green coloured area of the graph. If it tests bad, the pointer will jump into a red coloured area. As with the Check Memory command, more supporting information is needed if the user intends to adjust the motor speed on his drives. But, given the fact that TI99/4A owners use many different manufacturer's diskette drives, I do not think that it is reasonable to expect Miller's Graphics to provide instructions for performing this adjustment.

Other hardware related commands include Seek Track (to test the stepping function of diskette drives), and Use DSR; a command that acts as a toggle switch to either use the Device Service Routine ROM on the disk controller card or to use Advanced Diagnostics to set the time that it takes for a disk drive's stepper motor to step from track to track. CorComp owners will find Head Step to be a useful command in that they can use the Advanced Diagnostics program to determine the best head step time for each of their diskette drives. They can then use this information to set hardware DIP switches on their controller card for each diskette drive according to the instructions provided with that product. The usefulness of these two commands (Use DSR and Head Step) for the TI controller user is not as evident. The commands can be used to determine the best head step times but how would one use this information to set up a hardware configuration is not clear to me. Again, a little more documentation on the use of the information that can be gleaned from operating the program would be useful.

In the area of diskette diagnostics and utilities, Advanced Diagnostics really shines. The appendices in the users manual contain the best information that I have seen on the TI99/4A disk format with only one important omission that I could detect. The contents of Sector 0, Byte 16 appears to have been inadvertently omitted from the diskette map. Therefore, if you have the product, you should fill in this information on page 29 of the users manual. Sector 0, Byte 16: >50 diskette is backup protected; >20 diskette is not backup protected.

Diskette related commands that are very useful include Check Disk, Copy Read, Copy Write, Edit Sector, Write Sector, Format Disk, and Find File. The only additional command that I wish were included would be a Find String command. But let us not be picky.

Check Disk reports on diskette usage and reports any bad sectors that are found. More importantly to me, it lists "fractured files" on good diskettes. Fractured files generally arise because files are deleted, updated or otherwise modified from day to day under normal usage. And they tend to become scattered all over the diskette surface. This causes the stepper motor to search all over the place for portions of files to read, or for "holes" upon which to record parts of new files. Thus, the stepper motor has to search a bit to find all of the file that it is trying to access. This causes undue wear on the stepper motor (as well as the nerves, since one has to listen to the stepper motor chatter away as it goes about its business of earning a living). To clean up a diskette that is badly fractured, one copies the diskette, file by file, to a new diskette. The files on the new diskette then reside in contiguously numbered sectors and can be copied back to the original diskette using a sector by sector copy program (for speed).

A sector by sector copy program? No problem. If you do not own one, you can use the Copy Read and Copy Write commands to create one. Copy Read reads in up to 36 contiguous sectors to a copy buffer, whereas Copy Write writes them to the diskette.

Now this is nice. But do you know what would be really nice? It would be really nice if a file by file copy utility were in Advanced Diagnostics. As it is, you must change to the Disk Manager program (TI), or the Disk Manager program (CorComp) to do a file by file transfer. The Edit Sector command allows one to read in a sector from the diskette, display it in hexadecimal or ASCII (DISPLAY), and edit the sector

using a built in full screen editor. The edited sector can then be written to diskette. Of what use is this feature? A great big whole bunch of use. I had an occasion to use this feature as I was writing this review. It seems that my TI-Writer Multiplan working diskette catalog could not be accessed (probably because I had screwed up in trying out a new communications program). Anyway, I had TI-Writer, Multiplan, a letter to my Mom, and a previous version of this review on the diskette, and I simply did not want to lose them (one gets sloppy as they become 'experienced'). The Check Disk command indicated that sector 18 of my diskette was mapped as being "bad". Being a doubting Thomas, I tried to access sector 18 and found out that I could not. Next, I looked at sector 1, the file descriptor index record, and I noticed that I had a pointer to sector 18; therefore sector 18 should have contained a file descriptor record. I also noticed that the file descriptor record that should have been residing in sector 18 was, alphabetically, the last file on the diskette. At this point, I took a chance and replaced '18' in sector 1 with '00' (using the Edit Sector and Write Sector commands). I then tried to access sector 18 again. It worked! I could access sector 18. Sector 18 contained all standard formatting information, so the thought struck me that, perhaps, this $49 wonder did not recognize that anything had ever been recorded in sector 18. At this point, I looked at the allocation bit map of sector 0, and noticed that sector 18 was not marked as having been used. (Note that if I had known what I was doing, I would have checked this out before modifying sector 18!). Well, It looked like everything was in order so I tried out the Disk Directory command and I saw a nice disk catalog! Right before my eyes! TI-Writer worked, Multiplan worked, and my review was intact! Folks, I felt good. Advanced Diagnostics paid for itself right there.

If you think that the exercise mentioned above took years of accumulated experience, etc., forget it! It did take a little bit of mental exercise, but all of the requisite information for solving my problem was contained in a five page section of the Advanced Diagnostics user's manual. This manual is great, and it is only 34 pages long. It is not intimidating at all.

Closing out this segment of the review, the Format Disk command can be used to format diskettes in all diskette formats from single sided, single density; to double sided, double density. The ability to use all of these features depends on your disk controller and your diskette drives, of course.

The Find File command is useful for locating files on diskettes. This command indicates the starting sector and ending sector of the file if the file is located in contiguous sectors, or various starting and ending sectors of fractured files.

The remaining commands fall into the class of "those things that make the program a little more useful and life a little nicer" category. These commands include Beep, Change Colours, Select Drive, Output Device, Output Width, Pause, and Time Delay.

Well, the review would not be complete without a few more words on command files. Command files are executed by the command 'Command File DSKx.xxxx', where DSKx.xxxx is a DIS/VAR 80 file that contains any of the commands that I have discussed (and all commands that I have missed, inadvertently). The best way to gain experience in the use of command files is to run a few of the examples that are distributed with the product. Then view the example command file on your video display (TI-Writer, remember?), or obtain a printout of the command file and study it a bit. If you examine the Command File 'DSK1.DEMO1', you will discover that command files can run other command files if the next command file is the last command in the command file that calls it (whew!).

I expect that most purchasers of Advanced

Diagnostics will have to change one command file immediately upon receipt of the product. The command file "DSK1.DIAGCONFIG" executes immediately upon entering the program. This file, as distributed, contains the command OD "RS232.BA=4800. etc...". So, if you have a parallel printer, you will probably want to change this command to OD "PIO".

In summary, I enjoy using this product and it is worth $19.95 to me. But I cannot call the review complete without mentioning a pet peeve. This product is copy protected. I presume that the copy protection is of a particularly nasty form. Since we novices are told in all of the TI99/4A manuals to make real sure that we have a backup copy of all important programs and data, and to never use the original, I think that it is less than forthright of Mr. Miller to sell this program without noting the fact that he intends to force you to disobey the laws of common sense.

So, if you have $19.95 that you can afford to lose, and you do not intend to become dependent on this product, I recommend this program highly. On the other hand, if your pocketbook can not withstand the (potential) loss, or if you have a tendency toward flying into a rage over the suffrage of moral indignity, I would advise you to keep your money in your pocket. Many of the functions that this program provides are in other programs that are in the public domain, and I am quite sure that all of them will be, eventually.   o

The colour definition is likewise executed through a file, called DEFCOL. DEFCOL is also a mandatory DISPLAY VARIABLE 80 type file. The simple OPEN format (using the default values) is: OPEN #1:"DEFCOL". After that the various colours may be defined by PRINT statements. The PRINT command in Extended BASIC for white text on black background is:
    PRINT #1:"1777,7000"

We have redefined colour 1 to white and colour 7 to black. We use a 4 digit number for each colour. The first digit is the colour number (0 to 9, A to F: note that one has to subtract 1 from the numbers we normally use in BASIC). The other 3 digits define the intensity of the three colour components. It starts with the red component, then blue and green last. The acceptable value is between 0 and 7. 0 is minimum intensity and 7 represents maximum intensity. This option permits a large variation of individually determined colour mix. The file should be closed in the usual manner after the colour definition is completed.

APPENDIX

Mechatronic RGB Monitor Cable Connection.

9 pin D male pl.	Signal	Type
1	Gnd.	
2	Gnd.	
3	R sig.	Analog level
4	G sig.	Analog level
5	B sig.	Analog level
6	N.C.	Not used
7	+ 5v	max.load 50mA
8	Sync	TTL composite
9	N.C.	Not used

o

For Sale

A TI99/4A Beige console, Peripheral Expansion Box, two full height DSDD disk drives, Disk controller card, RS232 card and 32K Expansion Memory card. $600 for the lot. Phone Ron Holten (044)43 0539.   o

Mechatronic
80 Column Card
User's Manual
translated by Ben Takach

1. General description.

The 80 column card will produce 80 columns in 26 rows under program control. It also offers an enhanced graphic display. One can by suitable programming produce 256x212 dots in 256 colours, or 516x212 dots in 16 colours. In addition it is compatible with most of the known programs and hardware which means these will work as before on your TI99/4A. However a few enhancements are also available with these. The 16 standard colours of the TI99/4A may be chosen from 512 colour tones, as well as the position of the display on the VDU may be shifted within certain limits.

The card plugs into the expansion socket on the right hand side of the console. One has to start the installation by opening the console, in order to gain access to the video processor chip. This will be removed and replaced by a 40 way ribbon cable, which will also be plugged into the 80 column card. Step by step installation instructions will follow. Power for the card is taken from a separate power supply, a 6v dc plugpack (9v would be better ED), which plugs into a concentric socket on the rear of the card. This must be switched on before the console is powered up.

The monitor connector is on the back of the card. It is a 9 pin sub D type socket (identical to the joystick port or the tape recorder interface). The monitor output is RGB analogue signal with TTL synchronisation. Thus one needs an RGB monitor. Mechatronic offer a modulator as an optional extra, which has a composite video output and also a HF signal for TV use. The use of a TV set is not recommended. The 80 column display on the TV set is not very satisfactory. The audio signal is taken, as before, from the appropriate pin of the Din socket from the rear of the console.

The operation of the computer for all existing programs remains the same as before. The monitor attached to the 80 column card is used for all display purposes. The usual title screen appears after power up, and you may proceed as usual. Some consoles and a few modules are exceptions. These will be detailed in the following paragraphs.

The card is fitted with 3 sets of DIP switches. Switch bank 1 consists of 8 switches, these are used to centre the display on the VDU. Switch bank 2 is used to select the PAL or NTSC operation mode, as well as selection of US or German character sets. A switch is also provided to enable or disable the push buttons (more of these in the next paragraph). Switch bank 3 sets the DSR address. The switches must be set for CRU base address >1000 to guarantee trouble free operation. Other addresses may not be selected.

There are 2 push buttons on the PC board. These can be accessed through the steel cover plate by a ball point pen or a pencil. Mechatronic was forced to incorporate these control units as TI has used in some of the consoles and modules "false" VDU control codes. In practice, if the display is blank or the screen is filled with colourful garbage then the left hand side button is pressed to correct the situation. In such case one also has to push the QUIT key when the console is to be reset. The push button enable or disable dip switch is turned off with standard consoles to avoid malfunction and possible lockup due to inadvertent operation of the push button. The second push button so far has no function. (Perhaps it could be wired for interrupt routine?)

A resident EPROM contains the operating system and software. Its basic operating mode is the same as that of the console based video processor. The only difference is that the available VDP register space is 2 bytes shorter. It may result in the inability to load special copy protected programs. However these are very rare since the introduction of the various third party Disk Controllers, as these are also not compatible.

How Does It Work?

Text Mode, General, OPEN.

One invokes the 80 column text mode from console BASIC or Extended BASIC exactly as if it would be a printer. The device name is "TEXT80". Only DISPLAY 80 mode is allowed. The open statement also contains an up to 8 bytes long (16 Hex digits) VDP Register definer. This may be omitted in Extended BASIC mode (Extended BASIC mode definer is the default). Optionally, the cursor starting address may also be included in the definer. For example, "TEXT80.0000E00020000000" is the definer for Extended BASIC with cursor start address at the top left hand corner of the screen.

The 80 column mode is established by an OPEN statement, and a CLOSE command or statement will restore the standard TI99/4A format.

Control Character Codes.

Special screen functions are implemented by the use of control characters, just the same as it is done with printers. These may be embedded in the text. For example CHR$(17) will kill the display. Others are designated for the following screen functions:
 -cursor positioned at the upper left-hand corner,
 -highlighting display segments by blinking,
 -to stop all blinking (works on the entire screen)
 -to position the cursor anywhere on the screen,
 -define the size of an input value or string, (just like ACCEPT AT SIZE statement in Extended BASIC)
 -colour definition selection,
 -define the blink frequency
 -CHAR definition,
 -Input prompt default accepted as input.

The resident software also permits the redefinition of colours (e.g. inverse display mode), by the "DEFCOL" OPEN mode. A file may be opened in DISPLAY VARIABLE 80 format by the statement OPEN #1:"DEFCOL", to print selected data in the redefined colour.

A supplementary disk is supplied with the card. This software enables the use of the TI-Writer module in the 80 column format.

The card comes in a solid steel enclosure, it is fitted with 20 ICs, a 21 MHz crystal, several transistors in addition to the dip switches and push buttons as well as the usual passive components. The heart of the unit is the Yamaha V9938 Video Processor Chip. The components are mounted on a high quality double sided glass fibre PC board.

2. The installation of the 80 column card.

The 80 column card plugs into the I/O port of the console situated on its right hand side. However some internal modification has to be completed beforehand. The original video processor chip has to be replaced by a 40 core ribbon cable. You have to open the console. One will need a suitable phillips driver and a standard screwdriver.

Ensure that you are not wearing any material likely to cause static charges.

The 80 column card is fitted in a sturdy steel protective enclosure. The top cover is held in place by 4 small screws. These have to be removed and the 40 way ribbon cable has to be unplugged from the IDC header plug. Now we can start on the console.

First we have to open the console. Place the

console upside down on a table, and remove the 7 screws from its bottom cover. If you have a black and silver unit, then the on/off switch has to be pulled carefully forward. This is only held in place by clamping in a recess. Now the bottom half of the case may be lifted off.

Next remove the 5 screws securing the mother board and the console power supply, and disconnect the power supply plug from the main board. The AC input socket may be pulled out of its recess in the case. Take out the power supply board and put it aside. Now you can lift the mother board a little to gain access to the keyboard plug connection. Carefully pull the plug off the pins. Do not use any force, the wires are stiff but very thin. These break easily.

Now the mother board may be removed from the case. The metal shields are held together by 3 screws and two spring clips. Mark the position of the clips before you remove them for the subsequent assembly! Now remove the clips and the 3 screws and the 2 fixing screws which hold the metal cover to the heatsink on the VDP processor chip. Lift off the shield, which covers the component side of the board. Carefully take out the video processor chip and its heat sink. This is marked TMS9929A or TMS9918, however due to the liberal amount of thermal conducting paste used and the presence of the heat sink, it is generally not readable. Now you can thread the ribbon cable fitted with the 40 pin header plug through the I/O port's earth shield. This has to be unscrewed from the lower metal shield and removed. Ensure that the black marker of the ribbon cable points towards the centre of the board. Check the position of the locating notch on the IDC line socket, this must face up. Flatten the ribbon as much as possible on the motherboard.

Reassemble the console in reverse order. The video processor and its heat sink will not be needed anymore. Ensure that everything fits together without any strain, the components should fit snugly on their designated locating spigotts. The frequent lockup due to dirty module contacts is well known. Use this opportunity and clean them thoroughly. Position the console after assembly at its usual location, plug in the 80 column card and the 40 way ribbon socket. The steel cover of the card is then replaced. The two outer screws only will be replaced. Thus the installation is completed.

3. Setting up the 80 column card.

As already mentioned, the computer may be used as before using the monitor attached to the 80 column card. After switch on, the usual TI99/4A title screen will greet you, and you may proceed as usual. You will need an analog RGB monitor with TTL composite synchronisation.

3.1 Dip switch settings.
Dip switch bank 1 enables the shifting of the display on the screen within some limits. All of the switches in the off position means (from the video processor out) the display is in the centre of the screen. If the display is not in the centre then you may correct it as follows:

S1 on: up, S1 off: down
S2 to S4 degree of up or down displacement
S5 on: left, S5 off: right
S6 to S8 degree of left or right displacement

Dip switch bank 2 is used for a number of functions. In detail:

S1 off: The video processor is in PAL (50 Hz) operation mode
S1 on : The video processor is in NTSC mode
S2 on : Interrupt push button is enabled
S2 off: Interrupt push button disabled (definition of the push button is given below)
S3 on : USA ASCII character set is loaded
S3 off: German character set is loaded
S4 to S6 are not used

Dip switch bank 3 is used to select the CRU base address of the 80 column card. Caution! These must be set to >1000. Other addresses are not allowed. Thus all switches of bank 3 must be set to on. If you have another peripheral operating on >1000 then these must be reset to another CRU address.

3.2 The function of the 2 push buttons.
There are two push buttons on the card. These may be operated using a pointed object (pencil or ball point pen) through the holes punched in the card cover. These cumbersome units had to be fitted to the card as TI has unfortunately used "false" data to set the original video controllers of some of the consoles and also in some of the TI99/4A modules. Thus if your console or some of the modules used gives a blank screen or colourful garbage then push the left hand side push button briefly. The video processor of the 80 column card will then switch over and everything will function all right. However you have to remember to push the QUIT key twice in such case when you leave the particular module in order to see the correct title screen and to get it to function normally. If you have no such module or console, then it is advisable to switch S2 of dip switch bank 2 off to avoid any malfunction through inadvertent operation of the push button. The computer is likely to lock up if the button is pressed when it is not necessary. You will lose any program or data in the buffer as well as data on any diskette in the drive may be corrupted or lost.

The second push button is for future expansion.

4. The built in software.

The 80 column card is fitted with an EPROM, which contains the resident software. In the first instance this will take care of the initialisation of the card after switch on. It could take up to 1 second after initial switch on. In this base mode the video processor emulates the original TI99/4A processor, with one exception: the VDP RAM space is about 2 bytes smaller. It can happen that some early release specially copy protected disk programs will not load any more. However such protection practices are no longer used since the appearance of third party disk controller cards will not work with them either.

5. Text mode.

Now one can invoke an 80 column text display mode from console BASIC or Extended BASIC. The 80 column mode is treated like a file or a printer. The valid file name is "TEXT80". DISPLAY, VARIABLE 80 is the only valid file type. "TEXT80" may be optionally augmented with some extensions, these are added following a . (dot) delimiter. Since this text file may be OPENed out of any main program, the opened file has to be properly CLOSEd again in order to restore the video processor's register status. This is achieved in the first place through the file name extensions. For example, invoke text mode in Extended BASIC with the statement:
    "TEXT80.0000E0002000"

(You may omit the extension in this case, this is the default return code, which is automatically selected by the software if no extension is specified). If you have a console with "false" BASIC (i.e. you have to press the push button) then use the following extension from console BASIC:
    "TEXT80.0700E0F00CF8"

Consoles operating without the aid of the push button use the extension from console BASIC:
    "TEXT80.0000E0000C00"

The extension using Editor/Assembler is:
    "TEXT80.0000E0000E01"

For TI-Writer as printer output use the same as above:
    "TEXT80.0000E0000E01"

The meaning of these digits are:
1 and 2 digits: 16 Kbytes RAM-bank of the VDP RAM,
3 and 4 digits: Value of the VDP Register 0,
5 and 6 digits: Value of the VDP Register 1,
7 and 8 digits: Value of the VDP Register 2,
9 and 10 digits: Value of the VDP Register 3,
11 and 12 digits: Value of the VDP Register 4.

The so far explained extension may be extended by two more bytes, these specify the starting address of the cursor on the screen. For example:
"TEXT80.0000E00020000000"
will start the screen display at the top left hand corner.

The 80 column text mode is opened conventionally by a standard TI99/4A OPEN statement. The OPEN statement switches the screen to 80 column text mode, and all the standard screen commands are effective (such as CALL CHAR, CALL CLEAR, etc.), however these are not executed until the text mode is CLOSEd again.

We can write to the screen using the PRINT #**: expression or accept inputs from the keyboard through INPUT #**: or LINPUT #**:. Execution of the commands will always start from the cursor position to the end of the line. The standard edit keys are active during inputs (DEL, INS, Cursor/Arrow keys). PRINT will automatically execute a carriage return (to the beginning of the next screen line) after the display of each data set. INPUT will return all input characters including any leading and trailing spaces.

The 80 column text mode may be terminated by the CLOSE command. The computer then reverts to its standard mode. Caution: A program interruption during opened TEXT80 mode by a program error could result in incorrect screen display. In such case editing a non existing program line (entering the line: 1 REM) will simply close TEXT80.

An example of the TEXT80 mode using Extended BASIC:

```
100 OPEN #1:"TEXT80.0000E00020000000"
110 FOR X=1 TO 20
120 PRINT #1:"abcdefghijklmnopqrstuvwxyz{[]}1234567890
    ABCDEFGHIJKLMNOPQRSTUVWXYZ(\)~!"
130 NEXT X
140 CALL KEY(0,S,T):: IF T=0 THEN 140
150 CLOSE #1
```

6. Active Control Characters.

A number of control characters in screen print mode (PRINT #) provide some additional functions. These are similar to the general control characters applicable to most printers, even if these, due to the special nature could not be implemented in their original form. These control codes are:

6.1 Turning off the screen.
PRINT #**:CHR$(17) - will turn off the screen.

6.2 Repositioning the cursor in the top left hand corner of the screen.
CHR$(19) will reposition the cursor to the top left hand corner even during displaying data on the screen.

6.3 Invoke blink attribute.
CHR$(18) switches on the blink mode. The text following this command will blink. The blink mode may be switched off within a print sequence by the CHR$(20) command.

6.4 Stepping out of blink mode.
The text printed in the blink mode will remain blinking until the line scrolls off the screen or the TEXT80 file is closed. The character sequence CHR$(27);CHR$(101) will turn off the blinking display. All blinking will stop.

6.5 Cursor positioning.
The control sequence CHR$(27);CHR$(98);CHR$(r); CHR$(c) will position the cursor to any desired spot on the screen. In other words, the printing to the screen will continue at the designated spot or the input is accepted at that screen position. The valid r (row) value is 0 to 25, and the valid c (column) value may be 0 to 79. Control sequence CHR$(27);CHR$(98);CHR$(8); CHR$(40) will position the cursor in row 9 column 41.

6.6 Defining the INPUT string length.
The input string length is generally from the cursor position to the end of the screen line. If so desired the input string length may be defined by the control code sequence CHR$(27);CHR$(112);CHR$(n), where 'n' is the input string length. This may be any number from 0 to 80. Its length however may not exceed the length of the particular screen line. Please note: assuming the string length was defined to be 12 characters (CHR$(27);CHR$(112);CHR$(12)) then subsequent inputs may also be only 12 characters long until the 80 column input length is restored (by CHR$(27);CHR$(112);CHR$(80)).

6.7 Changing the screen colours.
The default colours are white text on black background and black text on green background for blinking text. These colours may be altered at will by the control sequence CHR$(27);CHR$(99);CHR$(n);CHR$(b), where n represents the normal colours and b the blink colours. The colour values are as follows:

Text colours		Background colours	
0	Transparent	0	Transparent
16	Black	1	Black
32	Mid green	2	Mid green
48	Light green	3	Light green
64	Dark blue	4	Dark blue
80	Light blue	5	Light blue
96	Dark red	6	Dark red
112	Cyan	7	Cyan
128	Mid red	8	Mid red
144	Light red	9	Light red
160	Dark yellow	10	Dark yellow
176	Light yellow	11	Light yellow
192	Dark green	12	Dark Green
208	Magenta	13	Magenta
224	Gray	14	Gray
240	White	15	White

To determine the n or b code number, the text and background colour values have to be summed.

6.8 Defining the blink frequency.
The time interval of the blinking text, which is displayed first in normal colours followed by the display in the selected blink colours may also be determined by the user. This is accomplished by the control sequence CHR$(27); CHR$(102);CHR$(t)
The approximate blink frequency (the t value) may be calculated from the timing table shown below. The t value is the sum of the normal and blink colour timing values.

Timing values:			
Normal colours		Blink colours	
Value	Time (s)	Value	Time (s)
0	0	0	0
16	0.16	1	0.16
32	0.33	2	0.33
48	0.5	3	0.5
64	0.67	4	0.64
80	0.83	5	0.83
96	1.0	6	1.0
112	1.17	7	1.17
128	1.34	8	1.34
144	1.5	9	1.5
160	1.67	10	1.67
176	1.84	11	1.84
192	2.0	12	2.0
208	2.17	13	2.17
224	2.34	14	2.34
240	2.5	15	2.5

I have concluded that the disk controller I am using with the RAMdisks is not compatible with the RAMdisks but there are many unanswered questions there. The solution is to get another disk controller, which is in hand waiting to be installed. The hard disk controller is another problem which I will address in a Bug Report next month, in which we will be highlighting the Myarc hard disk controller and hard disks. As for the poor hard disk with corrupted sectors, I was just starting to look at writing a disk fixer type program for the hard disk when I read two articles by Garry Christensen of Brisbane about the hard disk controller. The first was a program (Assembler) for copying the first half of a hard disk onto the second half and back again. The second one showed how to use Debug (or Super Debug) to enter a small program segment into memory which allows segments to be read and written to the hard disk. Very useful programs and ideas, Garry, thank you very much. Using the second one I read some of the sectors and found that the disk has some sectors which are no longer readable. One was in the subdirectory descriptor records sectors of the root directory which was easily circumvented. At least two others were in sectors used for the bit maps (sectors 1 to 31). Myarc's hard disk file system is somewhat inflexible here as these sectors cannot be changed for others if they are not usable. This means that the disk manager program reads the directory structure and then tries to read the bit map to see how many sectors are in use, which it cannot do and so this hangs up the program. This means that if a hard disk has a bad sector in this region, it is not usable by Myarc's hard disk controller. Bad sectors in other areas can be removed from use by setting their usage bit to 1.

****

Now on to something I am reluctant to write about. First let me say that I believe that Peter Schubert has done wonders for us all with his innovative approach to hardware expansion systems. I have or have had both a mini PE system and a multifunction AT card and think that the concepts are great. Unfortunately I have had problems with both pieces of hardware and I believe that the problems are not just with the particular items I possessed. I know of at least 3 other systems which have given similar problems. I think that the problems are to do with the software in the DSRs, as pointed out by Tony McGovern, as well as some hardware problems of a subtle nature like timing problems. If you have a mini PE system on its own or a multifunction card as a disk controller then things work reasonably smoothly. The only problems I experienced were the occasional writing of one sector of information to the wrong place on the disk. When editing many files on a DSDD disk, one or two would become corrupted with a sector of another file in the middle the wrong article. That was not too bad and no one else seemed to be having similar problems. When I used a RAMdisk with it I would sometimes find it very difficult to reload the RAMdisk. However when I used the mini PE system with the RAMdisk, I gave up before I loaded the RAMdisk as my floppy disks were being destroyed track by track. The disk drives were types where the heads were always down and that may make things like that more likely to happen. This would also be true if you chose to have the heads load on motor running rather than on the disk select. Now I have received a call for help from a member with a mini PE system and an AT RAMdisk for it with problems exactly the same as mine using Horizon RAMdisks. The AT RAMdisk has other problems of continually losing its ROS which I shall try and overcome, but I would like to warn others that there are possible problems with this hardware. As I said at the beginning, I do not want to stop this particular hardware from being available, but I do not think that it is fair for people to buy products without being told about possible problems with it. If anyone has a mini PE system with RAMdisks which works as well as they could hope for could they please write and tell me so I can give some good news. A hardware repairer only ever hears about the bad systems! o

(P. Powers); Fibonacci Numbers; Factorial (P. Powers); Linear Correlation; Linear Programming (A. Rahe); Math Aid, Math Help (G. Hitz); Matrix Inversion (C. Whitelaw); Projectile Problems (M. Wade); Quadratic Equation Solver (R. Jayavant); Regression Analysis (J. Wicklatz); Nth Order Regression, Matrix Multiplication, Matrix Inversion, Mohr's Circle.

781. HIGHER MATH #2 (228)
Triangular Numbers (P. Powers); Trigonometry (D. Morin); Trigonometrical Solution of Triangles (R. Jayavant); Using Logarithms (B. Falkin); 3x3 Linear Equations (T. Roberts); 3N+1 Problem (P. Powers); Surveyor (P. Wisselberger); Triangles, Trig-Trix; Trinomial Factoring (R. Jayavant); Simultaneous Equations (P. Wisselberger); Table of Trigonometrical Functions (T. Roberts); Binary/Decimal Test; Solution and Graphs; Solving 3 Variable Equations (J. Webb)

810. TYPING PRACTICE (350)
Tigercub Typing Tutor, Typezapper, Keyzap (J. Peterson); Type Invaders, Typ-ette Timer, Typing for Accuracy (Regena); Type Man (Towers/Johnson); Typo (Romox); Typo-Blast (Hallmark); Keyboard Test (J. Behnke); Typing Skill, Typing Teacher; Typo II (K. Siberz

815. MORSE CODE TEACHER (155)
Morse Code Introduction, Tutor, and Practice, by E. Dohmann; Morse Code Generator (T. Kruse); Morse Code Teacher; Morse Keyer (A. Minton); Morse Coder (B. McFadden) requires Editor Assembler module.

820. HEALTH and THE HUMAN BODY (354)
Biorhythm; Biorhythm #2 (Nahigian); Biorhythm #3; Biorhythm #4 (T. Roach); Biorhythm Compatibility (J. Volk); Biorhythms and You; Calorie Cop; Daily Nutrition (D. Shamet); Diet Rite; Name that Bone (Regena)

821. HEALTH #2 (145)
Self Evaluation; Computer Medi-Alert (D. Fischer); Coronary Risk Analysis (R. Tamashiro); Safety Awareness Program (S. Moore Jr); Teeth Wisdom (Regena); Wind Chill Factor (K. Wentzel)

830. PHYSICS (111)
Science Friction (McQuade Ent.); Windchill Factor; Temperature/Humidity Index (K. Wentzel); 4-Stroke Engine Demonstration; Jet Engine (P. Yorke)

840. NATURE (277)
Climate Pollution (Software Netherlands); Genetics; Lightning (G. Schechter); and Life (R.F. Kirmse) requires Editor Assembler module; Hurricane Tracker (Mineo/Mouledoux)

850. CHEMISTRY (277)
Chemical Symbols (J. Valling); Grunge on Chemistry; Hydrogen (Quon-Dobbs); Laboratory Calculator (K. Romstedt); Periodic Tables (P. Pruszinski); Peptide HPLC (P. McPhie); Table of Elements (J. Peterson); Protein Predict (Robson/McPhie)

860. ASTRONOMY (342)
Astronomy; 14 Constellations (K. Martin) 57 Constellations (Burgess/Volk); Elliptic Orbits, Parabolic Orbits (P. Severance); Jupiter's Moons (K. Jamieson); Planets; Phases of the Moon; Planetarium (R. Schenk); Skyscape (P. Parrish)

861. ASTRONOMY #2 (304)
Comet Halley (R. Browne); Mr. Bill on the Moon (C. de Marti); New and Full Moon Dates; Voyage to the Planets (S. Shouse); Watch the Planets Move (J. Priser); Planetary Comparison (Gary Jones); The Solar System (K. Williams); A Space Holiday Begins (L. Preece); Stars (D. Baker); Sunrise-Sunset (D. Wentzel); Star Travel (D. Thorpe)

870. RELIGIOUS PROGRAMS (910)
Bible Quiz; Bible Trivia (S. DeGeare); Daniel, with speech (E. Moss); Logic Flow of Christianity, Logic of

Regional Group Reports

Banana Coast   13/08/89 Sawtell
Carlingford   16/08/89 Carlingford
Central Coast   12/08/89 Toukley
Glebe   10/08/89 Glebe
Illawarra   21/08/89 Keiraville
Liverpool   11/08/89
Northern Suburbs 24/08/89
Sutherland   18/08/89 Jannali

BANANA COAST Regional Group
(Coffs Harbour area)

Regular meetings are held in the Sawtell Tennis Club on the second Sunday of the month at 2 pm sharp. For information on meetings of the Banana Coast group, contact Kevin Cox at 7 Dewing Close, Bayldon, telephone (066)53 2649, or John Ryan of Mullaway via the BBS, user name SARA, or telephone (066)54 1451.

CARLINGFORD Regional Group.

Regular meetings are normally 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.

Regular 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.

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 Lou Amadio on (042)28 4906 for more information.

LIVERPOOL Regional Group

Regular meeting date is the Friday following the TIsHUG Sydney meeting at 7.30 pm. Contact Larry Saunders (02)644 7377 (home) or (02)708 5916 (work) for more information.

All demonstration programs are subject to Air Mail from USA. Some are still on the way
*** ALL WELCOME ***

August Meeting, 11th August. Demonstration of Press.

NORTHERN SUBURBS Regional Group.

Regular meetings are held on the fourth Thursday of the month. If you want any information please ring Dennis Norman on (02)452 3920, or Dick Warburton on (02)918 8132.

Come and join in our fun. Dick Warburton.

SUTHERLAND Regional Group.

Regular meetings are held on the third Friday of each month at the home of Peter Young at Jannali at 7.30pm. Group co-ordinator is Peter Young, (02) 528 8775. BBS Contact is Gary Wilson, user name VK2YGW on this BBS.

Following the recent Tutorial Day at Woodstock Community Centre, the Sutherland Regional Group has now obtained an up to date copy of the Library catalogue from Terry Phillips. It is amazing to see the sheer volume of the software inventory and even more difficult to select some new programs for evaluation. This new software will form the basis of upcoming activities at future regional meetings.

Many thanks to Les Andrews for his efforts in instigating a TI Artist special interest group, which has provided the impetus at a regional level for the greater use of this program.
Peter Young

TIsHUG in Sydney

Monthly meetings start promptly at 2pm (except for full day tutorials) on the first Saturday of the month that is not part of a long weekend. They are held at the Woodstock Community Centre, Church street, Burwood. Regular items include news from the directors, the publications library, the shop, and demonstrations of monthly software.

If you did not attend last month's meeting, you missed seeing a demonstration of Myarc's Geneve computer, the TI Artist SIG in full flight (they were still going at 5:45pm!), a tutorial on strings and some photographs from Ross Mudie's assembly weekend which included one gem of a rather embarrassed director. If you missed all this, do not worry, yet more is to come in the following months:

August 5 - Buy, swap and sell day. Have any hardware, software, books, etc., that you no longer use? Bring it along to the August meeting and sell it to other members who are looking for the particular item. There will also be the first part of a series of three lectures on effective use of TI-Writer's transliterate command as well as the TI Artist SIG.

September 2 - Ross Mudie will demonstrate how to exchange programs and mail between two computers via modems. First a TI99/4A will be connected to a TI99/4A, followed by a TI - IBM connection. The TI Artist SIG will meet and the second part of the transliterate lecture series will be presented.

The remaining meetings for 1989 will be held on October 7, November 4 (Full day workshop) and December 2.

Craig Sheehan (Meeting co-ordinator).

For Sale

TI-Writer manual (original, not a photocopy) $18
TI Editor/Assembler manual (not a photocopy) $28
TI Multiplan manual (photocopied) $10
Navarone Widget (in as new condition) $35
Music Maker module (includes demo disk/tape) $10
LOGO II Curriculum Guide (original manual) $10
Personal Record Keeping module (suitable for cassette systems without 32K memory expansion). Complete with 2 manuals detailing the extra CALLs available. $12
TI Extended BASIC manual only $8
TI Terminal Emulator II module, gives unlimited speech capabilities with speech synthesizer. Comes with full documentation. $30
Phone (042)84 2980 up to 10:30 pm

Christianity (K. Hubbard); Journeys of Paul, Parts A and B (D. Cheathsam)

871. RELIGIOUS PROGRAMS #2 (42)
      Bible Books (C. Cooley); Who Am I?

890. TEACHERS' HELPERS (186)
      Grade Average Program (L. Hughes); Grade Book, Drill (K. Romstedt); Grade Point Average (R. Stickle); Grading Program; Teacher's Helper (J. Peterson); Teacher's Pet (L. Preece)