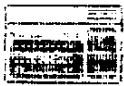




**THE FLUG
TI ROUNDUP**
THE OFFICIAL NEWSLETTER OF
THE FOREST LANE T.I.
USERS GROUP - DALLAS, TX

**USERS OF THE TI 99 AND
COMPATIBLES**

SEPTEMBER 1987
VOLUME 1, ISSUE 12
EDITOR: Richard A. Fleetwood

OFFICERS FOR 1987:

PRESIDENT	Richard Fleetwood
VICE PRESIDENT	Oscar Smith
SECRETARY	Keith Joyner
TREASURER	Ron Kuhlman
CORRESPONDENCE SECT.	Wilson Taylor
MEETING PROGRAM CHAIRMAN	James Carson

Next meeting of the Forest Lane Users Group is:

OCTOBER 4th, 1987-- PROGRAM : Artist programs for the TI 99/4A

IN THIS ISSUE:

UPCOMING SEPTEMBER MEETING INFORMATION	- James Carson
PRESIDENTIAL ROUNDUP - The latest on everything	- Richard Fleetwood
MINUTES OF AUGUST MEETING -	- Roy Willis
TI-bits - From the FLUG program chairman	- James Carson
DALLAS TI FAIRE - Updates on new date in April, committee, etc.	- Richard Fleetwood
ASSEMBLY LINES-Colum 12 - More on ASSY	- Richard DeHamer
A "PC" VOYAGE - part FOUR - more good stuff	- Marc Jensen
LEGENDS - PART TWO - a continuation of the history of TI's own "Real Soon Now" products	- Richard Fleetwood
BASIC COMPILERS - A review	-
THE FLUG LIBRARY - A repeat of information for those wishing to offer programs and disks to the club	- Richard Fleetwood
MODEMS - A short description of types of modems	-
TI WRITER COMMANDS - A very useful list of commands for beginning users of TI Writer	- Tom Kennedy
NUMBER CONVERSION - A short program to help figure out what number you need for your programs	- Barry Traver
PCODE PASCAL - A description of the language and other info.	-
TIBBS UTILITY PROGRAM- For sysops of TIBBS programs, a short program to build a DV80 userlist from your TIBBS userfile.	- Richard Fleetwood
STOP- Mike "Nuke the Whales" Stanfill's reaction to the FCC's plan to charge us more. XB TINYGRAM GARE!	- Mike Stanfill
CONVERTING GRAPHX FONTS TO TI ARTIST FONTS-Helpful hints	- Chris Bobbitt
TI DISK DRIVE INFORMATION - Info you need to know	- Jerry Coffey
ASSORTED OTHER GOODIES....	

SEPTEMBER MEETING INFORMATION

By - James Carson

Below are the planned activities for the AUGUST 2, 1987 meeting of the FOREST LANE USERS GROUP.

YOU ARE WELCOME TO BRING YOUR HARDWARE AND SOFTWARE TO THIS MEETING SO THAT WE CAN PROVIDE ANY ASSISTANCE YOU NEED SETTING IT UP OR FIGURING IT OUT.

- 2:00 - BUSINESS MEETING
 -Officers reports
 -committee reports-TIBBS, NEWSLETTER
 -Library report
 -TI FAIRE news
- NEW BUSINESS
 --Future meeting agendas
 --Monthly Workshop
 --Ramdisk offered to Club-Greg Justice
 --Suggestion of Metro Line by Net99ers
 --December meeting-Christmas dinner, officers elections, library disk party, and more.
- 3:00 - 3:20 - QUESTION AND ANSWER SESSION
- 3:20 - 3:30 -- COFFEE BREAK- 10 MINUTES ONLY
- 3:30 - 3:45 -- AUCTION OF HARDWARE AND SOFTWARE FOR FUNDRAISING FOR TI FAIRE
- 3:45 - 4:45 -- HARDWARE / SOFTWARE HOUR
 Hardware-Fixes for your system
 -Upgrade potentials
 -How to correctly set up your system
 -Hints and ideas for your hardware
- Software-How to load and run programs
 -How to change programs to suit you
 -Programming languages and their uses
 -How to write your own programs
 -the club library

4:45 - 5:15 -- SWAP SHOP AND NEWSLETTER LIBRARY BROWSING

Please help this and all future meetings run smoothly by asking questions at convenient times and not interrupting presentations.

PRESIDENTIAL ROUNDUP

By - Richard A. Fleetwood

Ding-Ding-Ding:

Its school time again, and with Labor Day weekend just around the corner, lots of us either find more time to our selves during the day, or find lots LESS time, due to homework. Our club contains a wide range of ages of our members, and we may see a shift in attendance, which way I don't know yet!

The biggest news this month is that we have had to postpone our proposed Ti Faire, due to unplanned financing problems, and support that fell thru when needed most. I'll have more on the specifics in the Faire column, but for now the Faire has DEFINITELY been rescheduled for APRIL 30th, 1988. There have also been several changes in the committee setup, and those will also be found in the Faire column.

On August 16th, FLUG had a special dinner meeting for its members. We finally brought out our newly finished software library, and had a good time visiting and having a great spaghetti dinner, and playing with lots of programs, as well as providing a few of our members with some stuff to take home and play with. Attendance at this event was rather low, with about 20 individuals in all, but the effort was well worth the outcome. We started something that we can continue in the future to help meet the needs of our membership. My thanks to Ron Kuhlman, his wife Martha, and my wife Annie for providing a great meal and a overabundance of goodies, and to Oscar Smith, Lee Dellenbaugh, and others for helping with setting up the systems and tables and everything else. My greatest thanks to Greg Justice for providing such a great meeting place for us, at no charge to the group. Greg went way out of his way to help us out, and spent th entire afternoon working, after working until 3 am that morning to fix some system problems. He really didn't get to share in the fun of the party, and I think we should do something special for him to show our gratitude.

We are planning another dinner event to happen during the pre christmas holidays, and will probably have it at our regular December meeting. More details as the time comes a little closer.

Prior to the dinner, we had our regular monthly meeting two weeks before. Among things we discussed were the Faire, hardware and software problems, and we had our first official auction. This auction was to start off the club on raising some more funds for our Faire. We raised over \$750 with the auction, with some hot and heavy bidding over lots of great hardware and software. One surprise visitor provided over half the purchases due to equipment needs!

We also had two great demos. The first one was by our surprise visitor, Paul Urbanus. Paul is one of the legendary programmers from out of Lubbock. Among his credits are Parsec, Munchman, and many other cartridge games. Other programs he has written that are just hitting the market are BARRAGE, JUMPY, and SPOTSHOT. These are being marketed by DATABIOTICS, and I highly recommend ALL of them. These are GREAT action games, with action you RARELY see in lots of games today. Contact me for more info on these. Anyway, Paul brought out a brand new device he has been working on for the past few months. You may or may not have seen this thing advertised yet by the people backing this item. The thing is a brand new 512K ramdisk, available in a kit or already built. The name of it is the GRAND RAM. I have seen it advertised already in full page ads in Micropendium, by a company called INNOVATIVE PROGRAMMING. Paul built the hardware under contract for them, and someone else is doing the software for it. The card provides ramdisk, print spooler, and clock functions as well as letting you use John Johnsen's MENU program for the Horizon Ramdisk. It also is going to support two EXTRA ports to allow AIGRAM simulation, and B) hardware adapters for piggy back boards like 4 channel music boards and other things. The demo was short, but interesting, and Paul said he would give us a better one after he gets some of the etched boards back. The board he had was a prototype wirewrapped board. Thanks Paul!

The next thing on the agenda was a demo by our club treasurer Ron Kuhlman about how to disassemble and clean, and even repair some parts of your 99/4A console. Quite a few questions kept this part of the meeting lively. Ron showed the basic components of the motherboard, the three major portions of the console, and told us where the most likely areas would be to look for if our systems started having problems. Ron promises more hardware stuff whenever we are ready.

The club is also planning and setting up a plan to start a loaner library for our members. We are going to make a motion to the membership to purchase used standalone hardware to allow members with minimal systems to experience what an expanded system can offer before they make any purchases. We will be setting up a standalone disk controller and drive with builtin memory expansion, as well as a standalone RS232 and Acoustic modem. These two systems will be an investment in the future for the club, as well as a way for new members to share in the fun of 99er computing. I hope you will support this project.

Future meeting formats will be a little different then the past few months. The executive committee, consisting of all the officers and the committee chairman, have been looking at ways to improve attendance at meetings, and we think that changing some things around we can make for a more diverse, yet infinitely more useful meeting, which we hope will draw more members every month. This month we are going to divide the meeting into two major sections-hardware and software. We will have the club system set up for software help of every kind, and then our other backup system for trouble shooting of members hardware problems, or for taking out and showing how to make things work, or where they go. Anything goes this meeting, so be sure to come and bring ALL your questions.

Finally, heres the latest on the TIBBS. I spent a few dozen hours a couple of weeks ago rebuilding the download library, adding new functions, and otherwise bringing it back up to its former self. It now contains over 80 new downloads, many archived, and more is on the way. I am dumping the VIP section in order to make room for an entirely new concept for our system. The TIBBS is now going to have several major sections. The first will be the main menu, which contains all the setting of parameters, the message base, and accessing the other subboards. I am going to set up this newsletter ENTIRELY in its own section, which will take up quite a lot of disk space, and which of course will be updated EVERY month. I am going to be adding menus for RLE pictures, HARDWARE projects, and INFO files for those things which are good to know month after month. I am also going to change some more things in the download section that will of interest to frequent callers. I'll announce that next month tho.

Hope to see ya at the meeting!

=*af=

Meeting Minutes August 2, 1987

By - Roy Willis

Meeting called to order at 2:06PM by Richard Fleetwood, with TI members present. Richard stated that Wilson Taylor was going to perform his new job as Newsletter Editor. Jimmy Carson, Meeting Program Chairman gave his report of the Programming activities of the club. Ron Kuhlman, Treasurer, was next with his report of the status of the Club Treasury. It stands at about \$800 at the present. Richard Fleetwood, President, was next with the status of the TI Faire.

The report of the Faire status brought about a discussion of the preparations that did not seem to be progressing at the proper speed and the doubt was expressed that we could not get the necessary work done in time to present the Faire in an acceptable manner. Motion was made and seconded, that the Faire be postponed until a later date.

While on the subject of the Faire, another motion was made and seconded that the monies raised for the Faire from all sources, be kept separate, so that we would know how we were progressing toward getting the necessary monies to fund the Faire.

It was decided that at the Club Spagetti Dinner, we would hold a Library Swap to introduce the new and reworked library. It was determined that disks would be available for sale at the Dinner, and that a charge of \$0.50 per disk be charged to copy. For Sale disk would be at \$0.45. If you wished and brought your own disk, it would cost you \$1.00. The Dinner was to be on August 16, 1987, at 3:00PM at TGI in Carrollton.

After a short break, we had a Demo of the Databiotics 512K card by Paul Urbanus. Some of you may recognize the name, he was one of the authors of the TI Game Cartridge, "Parsec". The card demoed was a pre-production model that was wire wrapped and actually more of a proto type. No price was available.

We then had an auction of material donated to the Club to raise funds for the Faire. There were Disk Drives, Modems, Games, Books and other various and sundry items. The bidding for some of the items was quite spirited and fun for all.

Ron Kuhlman then gave a demonstration of how to disassemble the Console and clean the contacts and other necessary items.

Meeting adjourned at 5:30PM.

(Respectfully submitted by Roy Willis in absence of Keith Joyner.)

TI-dbits

By - James Carson

FLUG Program Meeting Chairman

The FLUG newsletter on the TI Internal TIQLR computer system is going to be reduced to a maximum of 3 pages. It has not been updated for several months, and everything in it is quite outdated. I will try to update it weekly to include events planned for those weeks, or future events. It will include all current offices and phone numbers. The TIBBS number and other pertinent information will also be listed.

A summary of the officers meeting and FLUG meeting will be reported. G.G meetings will be reported as they happen. If you have anything you would like listed in this news file, please contact me.

James

THE DALLAS TI FAIRE REPORT

By - Richard Fleetwood

THE NEW DATE FOR THE DALLAS TI FAIRE IS SET FOR

A P R I L 3 0 t h , 1 9 8 8

Please mark IT on your calendar.

The following info. was posted on the club BBS the evening after our last meeting. It was posted because many people needed to know as soon as possible. Announcements were also posted on the DELPHI database system since there was a lot of talk about interest in the venture.

At the AUGUST meeting of the Forest Lane TI Users Group we made some very important decisions pertaining to the proposed TI Faire.

Decision #1:

FLUG is going to postpone the proposed TI FAIRE until the coming spring. We just do not have enough support from local users groups, both financially and personnel wise, that we need to make the FAIRE worthy of attendance. Our treasury alone cannot support the \$1200 we need up front to reserve the meeting place we have chosen. The MET99ers told us at their last meeting that they probably would not be able to help us financially, and we were counting on their help in this area. With the added time that we now have, they too will be able to raise some funds for the same purpose we are.

Also with the extra time, we will have more time to plan things out the right way, as well as get a much bigger head start on advertising, and lining up vendors.

DECISION #2:

FLUG, as a group, understands that the postponing of this proposed faire is going to hurt our club in the public eye, but it is better for us to hold off throwing a faire and to make it a better event when we do, then to throw a faire and bankrupt our club to the point of disbandment. We above all want to make the faire a reality, but we must garner support from EVERY local group to make it worthwhile for those attending. At this point in time, there is just not enough support to risk the investment by one group.

DECISION #3:

FLUG is going to set up a permanent FAIRE committee to oversee all aspects of this project. Although this should have been done long ago, it will be done now.

To add to the above almost a month after it was written, let me just say that things are looking much better. We have set up a PERMANENT committee to oversee all aspects of this faire, and have appointed a coordinator who has EXPERIENCE in producing national seminars and small conventions. He just happens to be a member of the group also. His name is Roy Willis, and Roy has been a big help both to me and to the club in the past several months. Roy will be taking over the responsibility of this event, from beginning to end, and will be in charge of doing out all responsibilities to those who will FOLLOW THRU. He has started this off by setting up a committee as follows:

FLUG TI FAIRE COMMITTEE FOR 1987/1988 SEASON

Faire Coordinator: Roy Willis 214-231-2168
 Vendor and User group Contact person: Richard Fleetwood 214-328-9257
 -all mailouts and forms
 Reservations - Hotel, vendor tables, Annie Fleetwood 214-328-9257
 -all systems for faire day
 Artwork - For flyers, advertisements, etc. Mike Stanfill

Facilities-Provide sources for systems, tables (vacate, but being supported and all equipment needed, as well as meeting by Roy and Richard for now) -place setup and teardown.

Advertising-for publications, promotional (see above--/)
 -stuff, T-shirts, xeroxing,etc.

Set up Coordinator Locate help, assign projects (see above --)

<<<<<<<<

As mentioned in the ROUNDUP column, we raised over \$350 at our last meeting JUST for the faire. We intend to raise much more. We know that on the outside a faire is a party, a big get together, a place to get to know people, and THOSE are the reasons we are going to do this. But on the inside, a faire requires MONEY to pay for advertising, to reserve rooms and equipment, to pay for food transportation, and lots of other things, that all have to be done before ANY money comes in from attendees. And AFTER the faire has come and gone we have to still have enough to run our club. That is why we have made the decision we did. We want this faire to be a GREAT one, even for a first time event, and we WILL do it.

We expect to see flak because of this decision, and we accept it because we knew it would come. If you are afraid to invest in this event because of this happening, we accept that, but we as a club will stand behind this and make the event come off, by ourselves financially if we have to.

Starting with the next newsletter, you will see this article being written by Roy, as his way of keeping us up to date with all the happenings. I will be putting all my efforts into the background area, contacting everyone by mail or by phone. Please give us feedback on anything you can concerning the faire. We will take EVERYTHING to heart.

Thanks for your support, and make plans for APRIL!

Richard Fleetwood

ASSEMBLY LINES

(Column 12) 08/87

By - Richard DeMaour

Let's get right to it this month. Last time I gave you source code for two files. One for the PAB and the other was the "original" main source code. This month I'm going to list five more subroutine source codes. Save them all to the same disk you saved last month's programs to. Next time I'll give you the final version of the MAIN source and be'll be finished. So, here goes:

```
CLEAR LI R2,768
LI R3,0
LI R1,>2020
CLEAR1 MOV R3,R0
INC R3
BLMP @VSWB
DEC R2
JNE CLEAR1
RT
```

This is an easy one. Save it to the disk as "DSK1.CLEAR". When called by the main program, it clears the screen.

```
STATUS EQU >837C
KEYADR EQU >8374
KEYVAL EQU >8375
SCAN CLR @KEYADR
SCAN1 CLR @STATUS
BLMP @KSCAN
CB @ANYKEY,@STATUS
JNE SCAN1
RT
ANYKEY DATA >2000
```

This one's fairly simple too. Save it as "DSK1.SCAN". When called by the main program, it scans the entire keyboard waiting for a key to be pressed. When a key is pressed, it's ascii value is stored at KEYVAL so the main program can "LOOK" at it to determine which key was pressed.

```
SCROLL LI R2,736
LI R3,32
LI R4,0
```

```

SCROLL1 MOV R3,R0
        INC R3
        BLWP @VSHR
        MOV R4,R0
        INC R4
        BLWP @VSHB
        DEC R2
        JNE SCROLL1
        LI R1,>2020
        LI R2,>32
        LI R3,>736
SCROLL2 MOV R3,R0
        INC R3
        BLWP @VSHB
        DEC R2
        JNE SCROLL2
        RT
    
```

A little more complicated, but not too. Save it as "DSK1.SCROLL". When called by the main program, it will cause the text on the display to "SCROLL" up one line. NOTE: These first 3 routines are written to be "STAND ALONE". That is, you should be able to use them in any other programs you write where the same function is required. (That's why I wrote them as subroutines !)

```

HTOAD  LI R1,BUFF2
        LI R4,>3030
        MOVB R4,*R1+
        MOV @NUM,R4
        CLR R3
        LI R2,100
        DIV R2,R3
        AI R3,>30
        SWPB R3
        MOVB R3,*R1+
        CLR R3
        LI R2,10
        DIV R2,R3
        AI R3,>30
        SWPB R3
        MOVB R3,*R1+
        AI R4,>30
        SWPB R4
        MOVH R4,*R1
        LI R1,BUFF2
        LI R2,>3000
        LI R4,3
HTOAD1 MOVB *R1,R3
        CB R3,R2
        JNE HTOAD2
        LI R3,>2000
        MOVB R3,*R1+
        DEC R4
        JNE HTOAD1
HTOAD2 RT
    
```

This one's a bit more complex. Save it as "DSK1.HTOAD". Unlike the others, this one is for a "SPECIFIC" use and will probably need modification if used elsewhere. HTOAD stands for HEX TO ASCII DECIMAL. The purpose of the routine is to take a hexadecimal number stored at a memory location labeled "NUM", and convert it to a series of ascii bytes that when written to the screen, look like a decimal number. It makes the assumption that the number is less than 1000. Briefly, it works like this: Register 1 is pointed to a buffer area (BUFF2). >30 (ascii for 0) is moved to the buffer and R1 is incremented. The value at NUM is moved to R4. R4 is divided by 100. >30 is added to the whole number result and then moved to the buffer. R1 is incremented. The remainder of the first division is then divided by 10. >30 is added again and the result moved to the buffer. R1 incremented. >30 is added to the remainder (the ones value) and moved to the buffer. Finally, R1 is pointed back to the beginning of the buffer, and a byte by byte compare is done, substituting >20 (a space) for all leading >30's (zeros). This blanks out leading zero's, and right justifies the number in the buffer.

*
 * ROUTINE TO CONVERT RADIX 100 NOTATION
 * TO HEXIDECIMAL. RESULT ENDS UP IN "NUM" !!!!

* PRIOR TO CALL (BL GET), R1 MUST POINT
 * TO FIRST BYTE OF RADIX 100 NUMBER
 * REGISTERS USED - R1,R2,R3,R4,R5
 *

```

GET    CLR @NUM
        CLR R5
        MOVB *R1+,R5
        CI R5,0
        JNE GET1
        RT
GET1   SWPB R5
        LI R2,64
        S R2,R5
        CI R5,1
        JEQ GET2
        CI R5,0
        JEQ GET3
        LI R2,10000
        CLR R3
        MOVB *R1+,R3
        SWPB R3
        MPY R2,R3
        A R4,@NUM
GET2   LI R2,100
        CLR R3
        MOVB *R1+,R3
        SWPB R3
        MPY R2,R3
        A R4,@NUM
GET3   CLR R3
        MOVB *R1+,R3
        SWPB R3
        A R3,@NUM
        RT
    
```

*
 NUM DATA >0000

This is the magic one ! Save it as "DSK1.RADIX". What this routine does is converts a number from RADIX 100 format to a hexadecimal number. To be perfectly honest, I wrote it so long ago, I forget exactly how it works. If you're real gung-ho there's an explanation of RADIX 100 in the ASSEMBLY manual.

That's it for now. Remember, next month the final "MAIN" source, and then we're through. Good luck and have fun.

RICHARD DENAMUR

A "C" VOYAGE, Part 4

By - Marc S. Jensen

Hello out there, faithful (I wonder?) readers! In this fourth part of the C voyage, we are going to learn about one of the singlemost important aspects of the C language. Like most other structured languages (and unlike BASIC), C supports the creation and use of subprograms and/or procedures. In C, these are collectively known as "functions." In all the programs so far, we have used only one function, "main ()". While the main function has no parameters (it is the main program), it is possible to use functions in many different ways: You can pass variables back and forth between functions, a function can call itself in a recursive manner, and a function can be made to return a value (meaning that "x=fcn(a,b)" will make x equal to the result when function fcn() manipulates variables a and b). Let's start off with a simple example. Assume that, in writing a program, you need to format text on your screen, by inserting an appropriate number of blank lines here and there. This could be done by several printf statements, or more effectively using functions.

```

main ()
{
    printf("line one");
    lines(3);
    printf("line two");
    lines(6);
    printf("line three");
}
    
```

```

lines(2);
}
line (n_of_lines)
int n_of_lines;
{
int x;
x=0;
while (x<n_of_lines)
{
printf("%n");
x++;
}
}

```

While this looks cumbersome, it is much easier than having to insert a great many printf statements in a large program. Let's go over this program and see how it works. The main procedure looks much as they did before, except for the "line" statements. These are not actual commands, but they cause the function "line ()" to be executed. The number in the parentheses is used as input for the function. After the brace that closes the main function, you will notice the statement "line (n_of_lines)." This tells the compiler that a new function, called "line" is beginning, and that it has one parameter being passed on to it, namely the variable n_of_lines. The next line, "int n_of_lines," lets the compiler know that n_of_lines is an integer variable.

It is possible to design many types of advanced programs using functions, but before we get to those (which may be several months from now), I want to talk some more about loops. While the "while" loop is easy and straightforward enough to use, it is not very efficient in most cases. The case in mind is the most common use of loops: counting. When programming, it is often needed to repeat a group of instructions a certain number of times, or print a certain number of blank lines. Using the "while" loop, this would take several lines of code, which is not very attractive. Fortunately, we have the powerful "for" loop. The basic format for this kind of loop is

for (initial expression; loop condition; loop expression)

as in the line

```

for ( x=1; x<=100; x++)
printf("This is a loop");

```

This line would print "This is a loop" 100 times, across the screen. Basically, this line can be translated as "Start out with x being equal to one. Then, while x is less than or equal to 100, do the statement(s) following this one.

For each time through the loop, increment x by one." In this same fashion, you could use a statement such as "for(x=50;x<=200;x+=2)" to count by twos from 50 to 200. So much control, in just one line!

Please note that there is no semicolon after the for statement. The "for" statement itself does nothing, except to specify how many times to execute the statement, or group of statements, following that statement. To execute more than one statement in a loop, simply use the braces in the same fashion as in the while loop and the if..else conditional.

As a final word this month (I'm pressed for time!), I want to demonstrate the fact that C programs are "compactable." You don't HAVE to skip all those lines between program segments; You don't HAVE to put the braces on a new line, all those things are only there to make the program more readable. For example, the "line" function presented earlier in this article could just as easily have read something like

```

line(n) int n; {
int x;
for (x=1;x<=n;x++) printf("%n"); }

```

To the compiler, the two are virtually identical. It is, however, much easier to write (and debug) a large program if it has been written in a nice format. Last month, I promised that we'd write a game program this time, but I haven't had the time to even consider doing that sort of thing. But next month, for sure...

LEGENDS...part two

TI 99/4A hardware and software that "never was"

By - Richard A Fleetwood

President-Forest Lane TI Users Group
Dallas, Texas

I've gotten a lot of feedback from readers across the country about my first installment of this series, all of it good! Some of my information mentioned had already been heard about, but not many people knew the good stuff.... I will continue this column for as long as I have information to share with you. I think that knowing things are possible might spark an interest in someone to come up with new hardware and software for all of us to share.

In this installment, I'm going to cover the following items:
THE TI GROM BOX

Next Month I'll cover these:
THE FORTI FOUR CHANNEL MUSIC CARD
THE TI IEEE-488 INTERFACE CARD

The TI GROM BOX, also known as the GROM EMULATOR, was a device that was the forerunner of the GRANKRACKER. It was used to dump cartridges to disk and then allowed you to modify the code using DISKO or some other kind of sector editor. It was used mostly by Q.A.E. (Quality Assurance Engineers) to examine new cartridge based programs and verify debugging before the final products were made.

As hardware goes, the Grom box was very simple. It consisted of a single board, which was powered by a regular TI 99/4a power supply (both wall unit and console p/s board). The housing (I've seen several different ones) was nothing more than an experimenters box, typical of ones used by electronic gadget makers for all kinds of things. The pc board held several dozen chips, most of which were for addressing. There were also a few PAL chips, several EPROMS, and then a 80K bank of memory, which was the actual space that was used to simulate the grom. Also used was a dip switch to select/deselect the memory chips used.

This unit was hooked up to a regular 99/4A through a cartridge which attached by way of two long cables. However, it was not a normal cartridge. The pboard of the cartridge extended out of the housing about 3/8's of an inch, and the pinout was exactly opposite that of a regular cartridge. In order to use the grom box, one had to cannibalize a working 99/4A console by cutting out an area directly behind the heat vents over the cartridge port. This would be directly over the power plug input. An opening had to be made that allowed you to plug the grom cartridge adapter into the back of the grom port. This was no major problem. I did mine in about ten minutes with a large pair of diagonal wire cutters. Once the opening was made, and before you put the console back together, you had to make a custom grom port system. Using an extra grom port unit, I de-soldered the connector from the small pc board and discarded the board. I then took the connector and soldered it pin for pin onto the back of grom port already in the console. The finished product was a grom port card with a connector facing the front of the console, and one facing the back opening just made. Once this was done, the console was ready to put back together.

On top of the grom cartridge adapter were two switches. These were double throw, multiple pole switches. When the cartridge was plugged into the back of the custom console, the switch on the left activated the grom box memory to the 99/4A operating system. The switch on the right enabled which ever grom port you wanted to use.

Software that I had for the unit were found on disks containing several modules worth of programs. A EA option 5 file (UTIL1) was used to read a DVS0 (CONTROL) file that contained all file information, menu information, and types of grom files(GRAM OR ROM), and what disk the files were to be on. UTIL1 was the same on all disks. CONTROL was different as per the files that were on that disk. In fact, I have found that the UTIL1/CONTROL system will work to load ANY EA 5 files on a regular system, ad not just those grom box programs.

To load any of these programs, you simply inserted your Editor Assembler cartridge into the front port, and the grom adapter into the back port. The left switch always stayed pointed back (away from you). The right switch first had to be flipped toward you so you could use EA to load the loader files. After going thru the EA menus to option 5, simply pressing enter loaded the DSK1.UTIL1 file, which then loaded the DSK1.CONTROL file info into memory. A menu was then displayed which allowed you to select the cartridge on disk you wished to load. After pressing the key, you then pressed <enter> and then the

system went to open that file. After pressing enter, you then needed to flip the right switch to the back to enable the system to load the files into the gram box. After the program was properly loaded, the system returned to the color bar screen, pressing any key displayed the standard TI menu (1. TI BASIC, 2. your cartridge) only now #2 was the cartridge you had just loaded into memory.

Everything was extremely easy to use once you used the system once or twice. I found thru playing with the hardware I could load my GK into the front port, plug the EA cart. into it, then plug the gram box into the back port. The software would load the regular way, and once the gram box was loaded, I could use the GK to dump the gram box contents into regular GK file format. I am not sure of the difference between the two formats, but one would not load into another.

As for software that I got with the unit, well, I can't tell you exactly what, except that I had all my cartridges already dumped and ready to use. I did receive several programs that never made it out the door, and those will be the subject of a future installment of this column. Suffice it to say, I believe this unit was one of the better things that TI did.

TI made this unit ONLY for internal use and testing, and NOT for resale or consumer sales. I know of about a dozen of these and believe that there were only a few dozen of these units made.

I'm not sure about how Craig Miller wound up with enough internal specs on the 99/4A to design the Gramcracker as he did, without knowing something about this device in the planning stages. The two units are very similar in operation, and have to be related somehow. Just a thought.....

Next month I'll cover in detail the FORTI Music card and the original TI IEEE-488 interface card (my serial number is 039!)

As I said last time, if you have any good information you would like to see covered in this column, it would be most appreciated. Please forward to me at the FLUGs address.

Richard

BASIC COMPILER V1.1

by-Herman Geschwind

While there have been many exciting developments in new programming languages for the 99/4A during the last three years (Forth in several variations, "c", Pilot and even a MacroAssembler), very little has happened to provide the TI community with a workable Basic Compiler. Compiled basic is readily available in several excellent implementations for the MS/PC-Dos and C/PM world, for the TI where a basic compiler would do the most good, very little has been available.

Why a Basic Compiler? The great benefit of Console or Extended Basic is that a program can be keyed in and then by issuing the "RUN" command, hopefully, the program will execute. If it should not work as expected, it is quite easy to edit the program and try again.

The penalty for this ease of use is that compared to programs written in other programming languages, both Console and Extended Basic will run very slowly. The reason for this is that Basic is "interpreted", which means that every program statement needs to be converted into machine language that the processor can understand and execute. Once the statement has been executed, the machine language conversion is discarded and the next statement must be converted. The inefficiency of this system is particularly apparent when statements within a loop must be converted over and over again until the loop ends. A compiler (such as an Assembler) makes this conversion only once and then the program is presented to the processor in a form which can be directly executed.

From this digression into the nature of "interpreters" versus "compilers" it should be clear that a system that could be both interpreted and compiled really might be the best of both worlds. As long as a program is still in the development state, it is much easier to test and debug and

edit, if necessary, in interpreted mode. Once the program is finished, it could be compiled for optimal running efficiency.

Ideally, a Basic Compiler should make it possible:

- 1) To use the standard repertoire of Console Basic or Extended Basic keywords, syntax and constructs. There should be no restrictions on the math capabilities, both integer and floating points.
- 2) The compiler should accept the basic program in its normal, executable form without the need to create Merge or List formats.
- 3) The compile process should be fast which means that the compiler itself would need to be an Assembler program.
- 4) The support library that may be required should not be copy protected or encumbered by legal restrictions so that compiled programs can be readily exchanged.

For the 99/4A there are only two compilers (SST and Compiler V1.1). Another product, 9900Basic, is not a true compiler but a product that uses a compiler approach to convert a limited number of Basic statements into Assembler code which then in turn needs to be compiled (assembled).

This review will deal primarily with Compiler V 1.1 by Rytte Data, 210 Mountain Street, Haliburton, Ont. K0M 1S0, Canada, but comparisons to the SST compiler will be inevitable and hopefully useful to put Compiler V 1.1 into perspective.

Compiler V 1.1 is an assembly program with an Extended Basic loader. It will operate with a single disk drive but is much more efficient and user-friendly as a two-drive system. As is to be expected, the system loads quickly and the compile process is lightning fast. By contrast the SST compiler is a basic program and the compile process for even a trivial program can take as long as an hour.

The Rytte Data product will accept normal Console or XB programs either with single or multiple statements per line. Again by contrast, SST will only handle single line statements in MERGE format.

Once the Compiler V 1.1 loads, the user is presented with a menu screen with the options LOAD: 1. BASIC, 2. COMPILER, 3. FP-LOADER, 4. INT-LOADER. Selection option 1. invokes a menu loader which scans the data disk and shows a directory listing of eight files with each file identified by number. Selecting a number will execute the program. Using the combination of SHIFT+NUMBER will load the program without execution and CTRL+NUMBER will delete the program. FNCT+6 will bring up additional files, again identified by numbers 1 through 8. (This menu selection also applies to the "compile" and "fp-" and "int" loader).

Selecting option 2 will invoke the actual compiler, again with "menu-by-the-numbers" selection of the program to be compiled. Once the compile process is finished, pressing ENTER will bring up the main menu again.

Compiler V1.1 comes with two support libraries, one for floating point applications (FP-LOADER) and one for integer only support (INT-LOADER). Except for the fact that the integer loader will not support decimal math and intrinsic functions such as SIN, LOG or SQR, it is otherwise identical to the floating point loader. For applications where floating point math support is not needed, the INT-LOADER will run programs much faster than the FP-LOADER.

So much for the good news. Compiler V 1.1 will compile Basic and XB programs in your library with blazing speed and no restrictions on program format. The bad news is that if you expect assembler-like speed from compiled programs you will be in for a disappointment.

The main gain is that there is no longer a lengthy delay for the pre-scan, programs will execute almost at the instant when the load process from disk is finished. As for actual run time improvements, results can vary widely. For programs with a lot of text, the performance gain is negligible. On the other hand, programs with elaborate graphics and sprites will benefit substantially from the compile process. Music and speech programs should not be compiled unless you like Donald Duck-like renditions.

Here are some examples to illustrate what is to be expected. Consider an empty loop of the type FOR I=1 TO 10000:NEXT I. Extended Basic will take approx. 35 seconds to run through this loop. The same program compiled and run with the integer loader will finish in only approx. 11 seconds. So far so good. But now let us introduce a print routine and see what happens with the loop FOR I=1 TO 10000:DISPLAY AT(15,6):I:NEXT I. In this case XB will be finished in a little over 1 minute, while the same compiled program will need 25 seconds longer to finish! We don't want to embarrass the compiler world by telling how fast a pure E/A program would run! Suffice it to say that one of the drawbacks of compilers is that the machine code which is automatically generated is never as efficient as a hand-coded E/A program. In the case of Compiler V1.1 the code appears to be particularly inefficient. A disassembly of the object code for the empty loop shows that 49 assembler statements were needed, not counting the code that is part of the loader. In the case of the SST compiler an execution time of 1.4 seconds is claimed for an empty loop of 1 to 30,000; but the price for this optimized code is a number of other restrictions.

Other bad news is that since Compiler V1.1 is entirely memory resident, there is a restriction on the size of the program that can be compiled. Again, conditions under which the compiler will abort with the message "Program Too Big" vary. In some cases 20 sector programs refused to compile while in other cases even a 30 sector program would compile and run properly. (A possible solution would be to segment a big program into separately runnable components since the compiler will support the "RUN DSKn.XXXX" command).

Unlike the SST compiler, which requires a major re-write (all variables must be explicitly declared at the beginning of the program, special syntax for many constructs, limited string and math functions), there are only a few cautions that need to be taken with Compiler V 1.1. Keywords that are not supported are DEF, SUB, OPTION BASE 1, TRACE, ON ERROR, CALL LOAD and CALL LINK. The END or STOP statement must be at the actual end of the program. A more serious limitation is that only one file can be opened during program execution.

A quick way to screen a program for these restrictions is to try a compile run. As unsupported keywords are encountered, the compiler will show "Bad Command Line XXX" where the line number refers to the basic program line number. Since the compile process is so fast, it is an easy matter to let the compiler finish and then to select Basic from the menu and fix and save the offending program.

Another disturbing aspect of Compiler V1.1 is that the code generated at times is not free of bugs and glitches. In some cases a compiled program appears to execute properly and then all of a sudden the system crashes or the console will lock up completely. At times a re-compile of the same program will result in better code, in other instances it was found that SAVE, MERGEing a program first would result in more stable code. Additionally there also appears to be a hardware dependency. Use of the GRAM KRACKER with customized Qsys and XB resulted in a succession of spectacular crashes. A plain vanilla black and silver console and XB cart. V110 cleared things up somewhat.

The copyright notice indicates that Compiler V1.1 was written in or prior to 1984. The fact that in two years the author did not issue a more optimized code and eliminate the annoying propensity to crash should give cause for concern. We have faulted many Fairware authors for their rapid succession of updated versions (see DMT000 and FUNLWRITER) but at least they made an attempt to improve their product.

The documentation supplied with Compiler V1.1 is all of 3 1/2 normal typewritten pages. Fortunately the user interface of Compiler V1.1 is accomplished enough that not much more is needed to get the program up and running.

Modifications: The default for the data disk drive is on sector 121, bytes 65 and 81 which on the distribution disk point to DSK1. Changing this byte to a 2 will default the data disk to DSK2 which makes for a much smoother operation with a two-drive setup.

One of the stipulations that we set forth earlier was that any associated support programs and loaders should not be encumbered so that compiled programs can be exchanged readily. Unfortunately both the compiler and the load programs of Compiler V1.1 are copyprotected. So is the SST compiler offering.

To sum things up, Compiler V1.1 certainly is a step in the right direction and compared to SST and 9900Basic deserves high marks for ease of use and user friendliness. Where our quest for the really good compiler for the TI is still unfulfilled is in terms of performance (execution speed) and the rough edges that are evidenced by the many glitches and crashes. With the Fairware market being what it is, maybe an unprotected compiler and loader is too much to hope for. For someone wanting to experiment with a compiler and willing to accept modest performance gains for programs of limited size, Compiler V1.1 might be acceptable for the small price of \$20. For programs of larger size and complexity, you might want to hold on to your money in the hope that someone, someday will come up with a truly acceptable and workable compiler for the 99/4A.

FLUG SOFTWARE LIBRARY CATALOG

The format for cataloging disks for the Forest Lane Users Group

Filename	Size	Type/No.	14 P	U
ASSORT8001	Free 615	Used 665		
ARCHIVER	32	PROGRAM		U
AUSSIEDAL	12	PROGRAM		U
BASS	36	PROGRAM		U
FLUGMEET	23 DIS/VAR		80	U
FORTHDEMO	309 INT/FIX		128	U
HRD/256K	41 DIS/VAR		80	U
MENU/XBNEW	163 DIS/FIX		128	U
NEWYEAR	35	PROGRAM		U
TINYCAL	8	PROGRAM		U
YAHTZEE	34	PROGRAM		U

The object of this article is to give the volunteers in this project a method to follow in producing a catalog of all available programs, files, articles, documentation, and other disk files that are in our present library. Once we have the basic library complete, we can begin IMMEDIATELY to input more and more programs and files, and it will grow to unbelievable size in a fairly short time. In order to do this though, we have to have a certain set of procedures we MUST follow so that any kind of disk will wind up in the correct library, along with complete documentation and a way that it can be tracked down by any individual in the club who may want it for any purpose.

We must follow four specific procedures in preparing these disks for our club library.

- 1) We must FIRST sort out all programs and categorize them by what kind of file they are. If they are BASIC programs, they go to the BASIC library. If they are ASSEMBLY programs, meaning they are OBJECT CODE or PROGRAM IMAGE files (anything that loads and runs out of the EDITOR ASSEMBLER CARTRIDGE), then they will go into the ASSEMBLY LANGUAGE library.
- 2) Once these programs are sorted out, they must then BE RUN AND VERIFIED THAT THEY WORK PROPERLY. If the file is a TEXT file, then they must be assured of being complete, meaning that they must have an end-of-file marker, so they will be assured of being loadable into TI WRITER.
- 3) After the files have been verified as being complete, then all files of the same library category must be put onto a disk, and the disk must be filled up to within a reasonable amount. We don't need to have disks that are only half, or a third full of programs.
- 4) Now that the disk is pretty much ready to put in the proper library, we need to take care of the final detail. We need to make a TIWRITER file describing the content of each program. This is the easiest, but MOST IMPORTANT part of the catalog. This is the file that will be placed in the HARDCOPY printout catalog that will be given to every member.

Below you will find an example of the type of file you need to make, based on a catalog of an imaginary disk out of my library.

```
DISKNAME="ASSORT8001"-(1)ARCHIVER-version 2.11 of Barry Travers archiving program-(2)AUSSIEDAL-Australian calendar maker from 1509 to 2099-(3)BASS-you are a contestant in a bass fishing contest-(4)FLUGMEET-dv80 file describing plans for the January 1987 FLUG meeting-(5)FORTHDEMO-public domain Forth language demos, released by TI-ARCHIVER WITH "DCOPY"-(6)HRD/256K-dv80 file of how to upgrade the Horizon Ramdisk to 256k-(7)MENU/XBNEW-The newest release of
```

the P.D. MENU program for the Horizon ramdisk-ARCHIVED WITH "ARCHIVER"
 - (B)NEWYEAR-music program that plays AULD LANG SYNE for a new years eve party-(9)TINYCAL-prints out very small calendars on EPSON printers-(9)WANTZEE-play the classic game

that is all there is to it. Please be sure you follow all the above procedures, and then before you know it, we'll have an ever-growing library.

BELOW ARE THE SUB-LIBRARIES, AND NUMBERING SYSTEM GUIDELINES...

CATEGORY	DISK NUMBER	CATEGORY	DISK NUMBERS
	UTILITIES	GAMES	UTILITIES GAMES
1. BASIC	1000-1499	1500-1999	6. GRAPHICS -6000-6999 xxxxxxxx
2. EX. BASIC	2000-2499	2500-2999	7. COMMUNICATION -7000-7999 xxxxxxxx
3. ASSEMBLY	3000-3499	3500-3999	8. MISC. FILES -8000-8999 xxxxxxxx
4. C/FORTH/PAS.	4000-4499	4500-4999	9. CASSETTE PROGRAMS-9000-9999
5. TEXT FILES	5000-5499	5500-5999	

MODEMS: Some Basics

BY- William Gregory - Greater Orlando 99ers

A modem is a translating unit between the computer and telephone system. A direct connection from the computer and telephone system is not possible because the output of a computer are digital pulses. The telephone system is designed to transmit human speech. By converting the digital output into a series of audio tones, information can then be transmitted over the telephone lines. Converting digital pulses into audio tones is called modulation. Going from audio tones back to digital pulses is called demodulation. A unit that can do both functions is called a Modulator/Demodulator or Modem for short.

Since digital information is composed of logic "one" and logic "zero", we use two different frequency tones to indicate this. In a bi-directional telephone transmission, there would be a problem with only one set of tones. Data can be sent both ways over the same set of wires, but it wouldn't work if the same set of tones were used to transmit data in both directions. By having a Modem use two sets of tones, this problem is solved.

One set of tones, (1070) HZ for "zero" and (1270) HZ for "one" is used for originating data. Another set of tones (2025) HZ and (2225) HZ, is used for receiving data. The receiving data of (2025) HZ would be for logic "zero" and the receiving data (2225) HZ is used for receiving logic "one".

When data is transmitted over the telephone system, noise on the line will interfere with the communication if the data is transmitted too fast. By sending data at 300 bits per second maximum, this noise problem is solved. This rate is also called 300 Baud, and it results in a transfer speed of 30 characters per second. Higher Baud rates are now available. Special lines and equipment are needed for these rates.

The two most popular types of modems in use today do not require unusual equipment. One is the acoustic-coupler, and the other is the direct-connect. The acoustic-coupler is cheaper, but more likely to pick up noise interference. So, take your pick!]

EDITOR'S NOTE

This article is about 18 mos. old. Don't know about those special lines and with so many 1200 Baud modems, I don't think of them as all that special either any more. Wonder where we'll be in another 18 mos.]

TI-WRITER COMMANDS

EDITOR COMMAND	FUNCTIONALITY	EDITOR COMMAND	FUNCTIONALITY	EDITOR COMMAND	FUNCTIONALITY
BACK TAB	1 1 T	INS. BLANK LINE	8 1 D	QUIT	1 = 1
BEGINNING / LINE	1 V	INSERT CHARACTER	2 1 B	REFORMAT	1 2cmR
COMMAND/ESCAPE	8 9 1 C	LAST PARAGRAPH	1 16cmR	RIGHT ARROW	1 D 1 D
DELETE CHARACTER	1 1 F	LEFT ARROW	1 S 1 S	WROLL DOWN	1 4 1 A

DEL. END OF LINE	1 1 X	LEFT MARGIN REL.	1 1 Y	WROLL UP	1 6 1 B
DELETE LINE	1 3 1 N	NEW PAGE	1 19cmR	SCREEN COLOR	1 1 1 3
LINE #'s(on/off)	1 0 1	NEW PARAGRAPH	1 18cmR	TAB	1 7 1 1
DOWN ARROW	1 1 X 1 A	NEXT PARAGRAPH	1 14cmR	UP ARROW	1 E 1 E
DUPLICATE LINE	1 1 5	NEXT WINDOW	1 5 1	WORD TAB	1 17cmR
HOME CURSOR	1 1 L	WOOPS!	1 11cmR	WORD WRAP/FIXED	1 1 0

LOAD FILES = LF (enter) DSK1.FILENAME (load entire file)
 LF (enter) 3 DSK1.FILENAME (merges filename with data in memory after line 3)
 LF (enter) 3 1 10 DSK1.FILENAME (lines 1 thru 10 of filename are merged after line 3 in memory)
 LF (enter) 1 10 DSK1.FILENAME (loads 1 thru 10 of filename)

SAVE FILES = SF (enter) DSK1.FILENAME (save entire file)
 SF (enter) 1 10 DSK1.FILENAME (saves lines 1 thru 10)

PRINT FILES= PF (enter) P10 (prints control character and line numbers)
 PF (enter) C P10 (prints with no control characters)
 PF (enter) L P10 (prints 74 characters with line numbers)
 PF (enter) F P10 (fixed 80 format)
 PF (enter) 1 10 P10 (prints lines 1 thru 10)

Note: if your printer uses RS232 switch P10 with RS232. To cancel the print command press FCTN 4.

DELETE FILE= DF (enter) DSK1.FILENAME

SETTING MARGINS AND TABS (16 tabs maximum)
 L - Left margin R - Right margin I - Indent T - Tab
 use ENTER to execute or COMMAND/ESCAPE to terminate command.

RECOVER EDIT= RE (enter) Y or N

EDIT = E (enter) (enter edit mode)

LINE MOVE = M (enter) 2 6 10 (moves lines 2 thru 3 after line 10)
 M (enter) 2 2 10 (moves line 2 after line 10)

COPY = Same as move except use C instead of M.

FIND STRING = FS (enter) /string/ (will find string)
 FS (enter) 1 15 /string/ (will find string in lines 2 thru 15)

DELETE = D (enter) 10 15 (deletes line 10 thru 15)

Base Conversion Program

BY- Jim Folz

I found this in a newsletter the other day and I had to have it. I am pretty sure it is not in its original form since I had to go in and fix it. It works now.

```

10 ! By Barry Traver
100 CALL CLEAR
110 INPUT "NUMBER?":N0
120 INPUT "FROM BASE?":F
130 INPUT "TO BASE?":T
140 CALL NUMCON(N0,F,T)
150 PRINT "THE ANSWER IS ";N
$
160 CALL KEY(O,K,S)
170 IF S<>1 THEN 160 ELSE 10
0 :: REM INSERT YOUR PROGRAM
HERE
3000 SUB NUMCON(N0,F,T)
3010 D=0
3020 L=LEN(N0)
3030 FOR I=1 TO L
3040 P=POS("0123456789ABCDEF",SEG$(N0,I,1))-1
3050 D=D*P+F*(L-I)
    
```



```

3060 NEXT I
3070 NS="":
3080 Q=INT(D/T)
3090 R=D-T*Q
3100 NS=SEG$(1*0123456789ABCD
EP",R+1,1)&NS
3110 IF Q=0 THEN 3140
3120 D=D-Q
3130 GOTO 3080
3140 SUBEND

```

A GUIDE TO THE p-System on the TI 99/4a

[Note: this is the first draft of this document...please make comments via MUSUS, the TI Forum or EMAIL (my # is 72406,2754), or US Mail, or via TI-Writer disks. My address is:

1540 Florida Ave. #205
Modesto, Ca 95350

I especially request help and comments at the places where I have placed percent signs (ZZZ):

Introduction

When TI discontinued the machine in October, 1983, few of the over two million 99/4 owners were using the p-System; if all 99/4 owners felt like orphans, we p-system users were doubly so. But however isolated we seem there are advantages: the p-System on the 99/4a is a full implementation of a "serious" operating system and as such we are able to do much more with it than TI-BASIC. Also, we can get help from any p-System user, regardless of their hardware. Finally, when the day comes when our machines either die or are outgrown, our programs will be portable to virtually any other machine on the market.

The purpose of this guide is to orient the TI 99/4a user to the p-System in general, and point out the special considerations about its implementation on our machine. It is hoped that this guide will be of value to all who are interested, from the serious programmer to those who are considering purchasing the p-System.

Overview of the p-System

The p-System is an operating system — a set of programs to run the computer and allow us to write and run programs. It is not a language; other languages might be available under the p-System. While only PASCAL is currently available to us, other users of the p-System can use BASIC, FORTRAN, and MODULA.

This concept is confusing to many TI 99/4 users because TI Console and Extended BASIC have the operating system built in — they include text editors, disk management programs, etc — all in the ROM containing BASIC. In the p-System one addresses all these functions separately, and most are on disk.

The essential part of the p-System for the TI99/4a is the p-Code card (or the old p-Code peripheral). The p-Code card contains some of the system programs in ROM. You must also have the 32K memory expansion. There is currently no way to use the extra 96K RAM available in the 128K card manufactured by Foundation. A disk is not absolutely necessary, as programs can be loaded and run from cassette. However, I do not know of any cassette programs, nor do I know of anyone who has used a cassette with the p-System.

The p-Code card includes, in ROM, the p-Code interpreter. This program accepts the user's and system programs in p-Code and executes them by interpreting them

into 9900 machine instructions.

Anyone serious about the p-System will have to have at least one and preferably two disk drives. The p-System will support double sided disk drives. If you are going to make use of the p-System for writing programs, then you will find that in a single drive system that your drive had better be double sided; two drives are better and two double sided drives are best. I see little use for a third drive. ZZZ Does anyone know if the Corcomp double density controller allows double density disks on the p-System (360 single sided, 720locks double sided)?ZZZ

A printer is certainly useful for the serious p-System user and can be supported through the RS232 (and PIO) card. ZZZanybody had experiences with other RS232 and PIO cards, peripherals, etc -- by 3rd party manufacturers?ZZZ

There are several programs that are important parts of the p-System. One is the PASCAL compiler, the program that takes PASCAL programs and translates them to p-Code, so the p-Code card can run them. This program comes on disk and is necessary to write your own programs but not to run someone else's programs.

Another disk contains two important programs: the Editor and Filer. The Editor is used to edit documents; you only need this to write your own programs although it can even, to some extent, be used for word processing. The Filer manages the disks; virtually any user of the p-System needs this program. Also on this disk are a number of utility programs which are of much value to most users.

The last disk contains the Assembler and Linker, both of which are necessary to do assembly language programming.

Learning to use the p-System

The TI manuals that come with the system components are pretty good as manuals go, but are not designed to teach the basic use of the system. There are many books about the U.C.S.D. p-System and PASCAL. Two very useful books are: "Introduction to the UCSD p-System" by Grant and Butah (Sybex), and "The UCSD PASCAL Handbook" by Clark and Koehler (Reward Books). The former is an excellent introduction to the use of the p-System. This guide will describe the unique aspects of our implementation by comparing the TI 99/4a system to that described in this book. The latter book is a relatively technical guide to UCSD PASCAL with very good examples. There are a number of good books that teach PASCAL; a discussion of them is beyond the scope of this guide.

USUS (the U.C.S.D. p-System users' Group) is a good source of help in the use of the system. Many of its members can easily be reached via the MUSUS SIG or Comuserve (PCS-55). The beauty of the p-System is that it, at least in theory, acts exactly the same on all systems; thus, you can get help from any knowledgeable p-System user; MUSUS is a good place to find such people.

The Filer

The Filer works pretty much as described by Grant and Butah. The major exceptions are that some of the prompts are shorter and the TI 99/4 only displays 40 columns at once. To see the whole 80 column display one uses the FCTN-7 and FCTN-8 keys (screen left and right).

It should be noted that disks may be physically in the drives but cannot be addressed by their volume names unless they were present at startup of the p-System, or reinitialization (I command) or by invoking the V (Volumes) command of the Filer. This command doesn't just list the on-line diskettes, but formally recognizes them so that they can be accessed by volume name in user programs. If you want to use the FILER but didn't have it in a disk drive at start up, then you have a problem. You cannot invoke the FILER V command to get the system to recognize the disk - because you can't run the FILER. The solution to this catch-22 is to use the system I command and restart the system. On restart, the p-System looks anew at the disk drives, and will then recognize the diskettes loaded in them, and know where to find the FILER, EDITOR, etc.

The Z (zero) command sets up a new (or zeros an old) disk after it has been formatted by the DFDRMAT utility program (or the Disk Manager Cartridge). Single sided disks have 180 (512 byte) blocks, double sided disks are twice that size. Using duplicate directories consumes 4 extra blocks but will often avert disaster when disk failure occurs.

The Editor

The Editor for the 99/4 is that which Grant and Butah call the "Screen Oriented Editor" — the Editor that is discussed at length in their book. In general the Editor works just like that of Grant and Butah with a few differences that could probably be best described as bugs. The "S" (Same) option on the "F" (Find) command does not work. The copy buffer seems to be emptied at times for reasons that are not obvious. After repeated insertions and deletions of a given page on screen, often the screen image is not correct. The actual text is different. As a result, it is best to use the "V" (Verify) command when in doubt. It re-displays the current screen page.

The 99/4A's small ram size allows only a little more than 12k characters in the text buffer. If you need to break down a larger file the AUTOPSY program in the USUS library can be of help. Programs can be written in several parts and strung together with the PASCAL compiler \$!f (include file) command.

Compiler

The TI 99/4a PASCAL compiler is a full Version IV.0 implementation. It seems to be identical to the compiler described by Clark and Koehler. In its TI version, the Compiler appears to be in four parts: 1) the start-up code; 2) the code to process the Declarations, 3) the code to process the body of procedures, and 4) the code to produce p-Codes. Parts 2 and 3 are swapped in and out for each procedure in the program, making compilation a good time for a coffee break. The swap parameters described by Grant and Butah seem to be inoperative on the TI; I suspect that the compiler is set for maximum swapping to accommodate the TI's limited RAM. Making a listing causes the already slow compiler to run much slower still, so one should be sparing in using this function. Also, if errors are found by the compiler it may not make a listing, even if the user keeps continuing compilation to the end. It may even hang with the error:

```
STACK OVERFLOW#REBOOT
```

At this point, one must turn the console off and on again and to restart.

The TI 99/4 version of PASCAL includes a number of extensions that allow the use of the special capabilities of the machine. These extensions are in the form of procedures included in the "TEXAS INSTRUMENTS UNITS", which are in the system LIBRARY. These include the routines to change screen color, use graphic modes (bit map, multi-color, etc), sprites, sound and speech. The use of these routines will allow the sophisticated user to do in a high level language what otherwise was previously only possible in Assembly. The new TI FORTH possesses these features as well. Please note, however, that the use of these procedures will make the program machine dependent, and thus not portable. One cannot, for example expect a program using sprites to run properly on an APPLE, because that machine lacks the hardware to produce them. To keep a program portable, one should avoid using the facilities in the TEXAS INSTRUMENTS UNITS.

Among the routines included in these UNITS, I have only used the procedures for random number generation and screen color definition. These procedures work as specified. I welcome input from others who have used the other facilities.

For program development, it is desirable to be able to quickly go from compiler to editor, so it is good to have both programs on one disk or both on different disks on different drives (in a multi-drive system). Also, put the file SYSTEM.PASCAL on the ROOT volume so that meaningful error messages are displayed.

Utilities

There are a number of valuable programs included with the Editor and Filter disk. This guide will discuss the most useful of these programs.

DIFORMAT formats new disks (or reformats old ones). It serves the same function as the Disk Manager module in this function. It can format double sided disks but usually fails (I was told by TI that they knew of this bug) but works fine on single sided disks. For double sided disks, one can always use the Disk Manager 2 module if available. The Zero command in the FILER must still be used in either case before a disk is usable.

MARKDUPDIR and COPYDUPDIR are useful in dealing with duplicate directories on disk, whose use is highly recommended.

MODRS232 alters the device definition associated with REMIN: and REMOUT: (these use the same definition) and PRINTER:. TI has kindly included the source code

for these programs. If you use a printer whose definition is different from the default (RS232/2.BA=9600.DA=7.PA=0.EC — this probably isn't what you're using), you can alter the source code to make the printer definition without user prompting and run it as SYSTEM.STARTUP. The rules for describing the RS232 or PIO communication setup are the same as used by BASIC.

RECOVER is useful if an both directories are blown (or you foolishly used only single directories). It can help restore lost files. PATCH will allow repair of individual files. You have to be a very expert user to use PATCH, however.

SETLTYPE allows one to force a program to reside in RAM rather than VDRAM. Its stated purpose is to declare whether a program is in p-Codes or native code (machine language). Machine language programs must reside in RAM; p-Code programs may be in RAM or VDRAM as the p-System selects (usually in VDRAM). The system views both VDRAM and RAM as one large space to be filled with programs and data. It will store data down from the high end of RAM and programs up from the low end of VDRAM until RAM or VDRAM is filled; then it will overflow into the other space (i.e. programs will overflow in RAM or data into VDRAM). One can use SETLTYPE to force a program or segment to be in RAM. If this is done it will run somewhat faster, but may impinge on (and thereby limit) the area for data. As mentioned above, assembly language programs must be put in RAM with SETLTYPE.

Using the p-System

Several notes can be made about using the p-System on the TI99/4a. First, you cannot use the TI LOGO cartridge on a system with the p-Code card in place. The p-System will work fine; LOGO will, in certain circumstances, loose control to the p-Code card, destroying what you were doing with LOGO. If your p-Code card has an on/off switch, all will be well if you turn the card off before using LOGO.

When a 99/4 is powered up with the p-Code card is in place (and turned on) the p-System initializes and then displays the p-System greeting. The p-System remains active until the user stops it; thereafter it will not restart without using the technique below (or turning off the system). To get from the p-System to the BASIC environment is simple — use the p-System 'R' command. To go from BASIC to the p-System, you need to have one of the following cartridges in place on the console:

1. Extended BASIC
2. Editor/Assembler
3. Mini Memory

From console or Extended basic, execute the following command:

```
CALL LOAD(14536,0,0)
```

and then type fctn= (Quit) or the BASIC command 'BYE'. The p-System will then operate as if you had turned the system off and then on again. Some programs and cartridges seem to have the same effect as the CALL LOAD listed above. Exiting TI-WRITER, MULTIPLAN, or the COMPANION word processor will lead to restart of the p-System. Likewise, some of the commands in the DISK MANAGER will reset the p-System so that it will become active again upon leaving this module.

Speed is an issue of interest to all potential users of the p-System. If computing speed is the reason for the TI BASIC programmer to consider the p-System, the following facts apply: Because the p-System is interpreted, it is relatively slow compared to compiled PASCALS (such as TURBO PASCAL on MS-DOS and CP/M systems). Compared to other machines, the p-System on the 99/4 is quite slow — I believe it is the slowest implementation of the p-System around. TI BASIC, however, can be benchmarked with a sundial, so it is easy for the p-System to beat. Number crunching programs PASCAL can run up to 60 times faster than their BASIC counterparts, especially if put in RAM (see SETLTYPE). Programs doing mostly screen I/O can be slower than BASIC, however. Disk I/O seems a little faster with the p-System — perhaps because of the simpler file system, leading to less searches for the next block, etc. I have no experience with FORTH, but know that it too is faster than BASIC. The programmer desiring speed above all else should compare the p-System to FORTH.

On running the V command in the FILER program, you will notice that there is a device #14 called 'OS:'. If you do not have a disk in drive one (the usual root device), one will find that the 'PREFIX' and 'ROOT' devices are defined as 'OS:'. This device includes files that are in ROM and are used for booting the system. You can read and examine these files but not, of course, modify them. To change the PREFIX definition, use the FILER P command; to change the ROOT definition you must use the system I (warn start) command.

Compatibility with other Machines

The p-System on the 99/4 is very compatible with other p-System implementations. Source code developed on the 99/4 can and does run without modification on most other machines if you avoid using the TEXAS INSTRUMENTS UNITS included with the PASCAL Compiler. The 99/4 is less able to run programs from other machines due to it's relatively small RAM size. If the program is properly commented, however, there will be no problem. The PASCAL compiler, for instance, is 99 blocks long -- 49.5K. It runs on the 99/4 (with a lot of disk swapping). The only other problem in running program from other machines on the TI is the presence of a 40 (rather than 80) column display. The TI will display all 80 columns with the SCREEN RIGHT and LEFT keys, but you will probably want to modify the display (if possible) to fit the screen. In the file SCREENOPS.CODE on device #14 (see above) is a description of the program in the case of the TI that the screen is 40 columns. Clever programs will adapt themselves to the size of the screen.

A larger problem is physically transferring programs to and from the 99/t any functional communications program at this time. Files can be exported from the 99/4 easily by using the FILER to transfer to RENDOUT.

TIBBS UTILITY PROGRAM

By - Richard Fleetwood

This program was found in an old 1985 issue of the Bayou Bytes newsletter from Louisiana Users Group. Most of the program was written by Roger Hickerson, the Sysop of their TIBBS. I used the program as is for a few months then started playing with it making it easier to use. What the program does is take a standard TIBBS program userfile, which happens to be an INTERNAL, RELATIVE, FIXED 85 file, and reads the pertinent, although not private, information, and writes a DV80 file to disk, which can then be used as a listing of users on the system for other callers to recognize. If a caller needs the exact name of a user to leave a private message, or wants to know if one of his friends frequents the board, then he can use this list to tell.

My changes to the program include printing both to the disk file and printer at the same time, letting you enter the date and starting and ending user number, and a few other things. The listing is relatively simple, and can be easily modified for different printers, file formats for other bbs systems, or whatever.

If you have any questions about it, or further enhancements, please contact me thru the FLUG TIBBS, at (214)328-4880, or thru the Forest Lane TI Users Group mailing address.

FOREST LANE TI USERS GROUP
P.O. BOX 743005
DALLAS, TEXAS 75374-3005

```
50 CALL CLEAR
55 DISPLAY AT(5,5):"TIBBS USERFILE LISTING"
56 DISPLAY AT(6,12):"PROGRAM"
57 DISPLAY AT(8,5):"By Richard Fleetwood"
58 DISPLAY AT(9,2):"FOREST LANE USERS GROUP"
59 DISPLAY AT(11,1):"based on a program written"
60 DISPLAY AT(12,1):"ROGER HICKERSON,BAYOU 99ERS"
61 FOR D=1 TO 1000 :: NEXT D :: CALL CLEAR
62 DISPLAY AT(5,3):"insert disk containing:";"USERFILE into drive 1:";"":then pre
```

```
55 ENTER"
63 ACCEPT AT(20,1):V$
64 CALL CLEAR
65 DISPLAY AT(10,3):"ENTER TODAY'S DATE:"
66 DISPLAY AT(12,3):"08/28/87"
67 ACCEPT AT(12,3)SIZE(-8)BEEP:DATE$
100 !CONVERT TIBBS USERFILE
110 !TO 819/VAR 80 USERLIST
120 !BY ROGER HICKERSON
130 !9/29/85
140 OPEN #4:"DSK1.USERFILE",INTERNAL,RELATIVE,FIXED 85
150 OPEN #5:"DSK2.USERLIST",VARIABLE 80
151 OPEN #6:"P10"
152 PRINT #5:DATE$ :: PRINT #6:DATE$ :: PRINT #5:"FLUG TIBBS USER LIST=" :: PRIN
T #6:"FLUG TIBBS USER LIST"
153 PRINT #5:RPT$( "-",60):: PRINT #6:RPT$( "-",60)
154 PRINT #5:"Last Name First TI? Location Level"
155 PRINT #5:RPT$( "-",60)
156 INPUT "LOW USER NUMBER ?":LN
157 INPUT "HI USER NUMBER ?":HN
158 PRINT #6:"Last Name First TI? Location Level"
159 PRINT #6:RPT$( "-",60)
160 FOR K=LN TO HN
165 DISPLAY AT(20,5):K
170 INPUT #4,REC (K:TS,US,V$,W$,X$,OAS,OB$,OD$,OE$,ALT$,PHONES,LV$
175 LV=ASC(LV$):: LV$=STR$(LV)
180 IF ALT$="Y" THEN ALT$="YES " ELSE ALT$=" NO "
190 IF T$="" THEN 240
200 LV$=USERPTS(" ",(13-LEN(US$)))
210 T$=ALT$
220 FN$=VALRPT$( " ",(8-LEN(V$)))
225 W$=VALRPT$( " ",(20-LEN(W$)))
230 PRINT #5:LV$;" ";FN$;T$;W$;" ";LV$
235 PRINT #6:LV$;" ";FN$;T$;W$;" ";LV$
240 NEXT K
250 CLOSE #4 :: CLOSE #5
260 END
```

THE FCC STOMP

By - Mike Stanfill

(EDITORS NOTE: Mike uploaded this program to our TIBBS a few days ago, and I wanted to share it with you. It somehow also expresses my feelings about what I think about the FCC and their stupid proposals brought on by money grubbing lobbyist for phone services. Hope you enjoyit is fun to play! =raf=)

```
100 !****"THE FCC STOMP"**** *****A TINYGRAM***** ***BY MIKE STANFILL
*** *****MEMBER DTIHCUG*****
101 CALL CLEAR :: PRINT " THE FCC STOMP:";"":;"A RESPONSE TO THE ATTEMPT:";"B
Y THE FCC TO FURTHER:";"RESTRICT OUR 1ST AMENDMENT:";"LAWS!"
102 PRINT "";"USE THE SD KEYS TO MOVE:";"YOUR STOMPER LEFT AND:";"RIGHT, AND USE
THE A KEY:";"TO STOMP THE HEADLESS FCC:";"REPRESENTATIVES!"
103 PRINT "";"PRESS ANY KEY TO START!"
104 CALL KEY(0,K,S):: IF S=0 THEN 104
110 CALL CLEAR :: B$(0)="41413E0B087F4141" :: B$(1)="7F49490914141436" :: CALL C
HAR(129,"000001030FDFFFSE",131,"FFFFFFF3E" *B$(1))
120 T=2 :: CALL MAGNIFY(3):: Y=16 :: CALL COLOR(14,S,5)
130 CALL KEY(1,K,S):: CALL SPRITE(81,128,5,1,Y*8-15):: IF K=1 THEN 150 ELSE Q=(K
=2)-(K=3):: IF Q*Y>1 AND Q+Y<32 THEN Y=Y+Q :: DISPLAY AT(1,1):J
140 CALL CHAR(132,B$(T*(T-3)*-1)): T=T+1 :: T=T+(T=5)*3):: IF SP(T)=0 THEN CALL
SPRITE(81,132,2,185,1,0,RND-9):: SP(T)=1 :: GOTO 130 ELSE 130
150 CALL LOCATE(81,177,Y*8-15):: CALL VCHAR(2,Y,136,22):: FOR Z=2 TO 4 :: CALL C
OINC(81,BZ,8,H):: IF H THEN CALL DELSPRITE(WZ):: SP(Z)=0
160 J=J+H :: CALL SOUND(-399,110,Z*6,-5,Z*7):: NEXT Z :: CALL VCHAR(1,Y,32,23)::
CALL LOCATE(81,1,Y*8-15):: GOTO 140
```

Converting GRAPHX to TI-Artist

=====

I recently read an article in the Hoosier User's Group's excellent user group newsletter on converting GRAPHX to TI-Artist with interest. I too had faced this dilemma some time back in trying to transfer our popular GRAPHX Companion series of products over to TI-Artist (a project which regrettably still has never been completed due to time limitations and a rather low priority). In any case, I thought the procedure we worked out may be of interest as well.

After playing with both programs for a while, we hit upon the solution offered by Mr. Robert Coffey. As a matter of fact, on our now discontinued Artist Companion disk there is a font that was converted over in just such a manner. It was so time consuming that we soon gave up.

The matter stayed dropped until 9 months or so ago when we were preparing Font Writer for release. I asked Peter Hoddie, the author, if he knew of a way to convert files over, and he said that Font Writer could be used.

Later that week, I sat down, and 4 hours later I had my first font. I chose a very elegant Times Roman, with a complete upper and lower case alphabet, from GRAPHX Companion IV (which was then in the editing phase) for the experiment.

The process is rather simple, actually.

Step one involves getting the fonts to TI-Artist format. If they are stored in a clipboard, as our GRAPHX Companion series fonts are, this involves first pasting them onto a screen (leaving plenty of room between characters, and then saving that screen to disk. As mentioned in Mr. Coffey's article, it is a very good idea to use an empty disk for this.

Next, enter TI-Artist and select the conversions section. Convert the screen from GRAPHX to TI-Artist format (load it in as a GRAPHX screen and save it under TI-Artist).

Next, enter the TI-Artist section from the menu, and select disk options. Load the screen into memory. Leave TI-Artist and go to the Enhancements option. The screen you loaded in TI-Artist will be in memory. Next, enter the Slides menu and select the Save Instance option. The filename you give should be the ASCII character that the picture you are saving represents (i.e., if the picture is of an "A" the filename should be A). Next, the screen will appear. Move the cursor to the upper left corner of the character picture you are saving, press the fire button, and move it to the lower right side of the picture, boxing in the character. Press the fire button again and the picture will be saved to disk. Do this over and over until the whole font is saved as individual instances.

When you are done, exit TI-Artist and load your copy of our Font Writer program. Enter the Font Editor option.

When that portion of the package is loaded, go to the menu options, and select the option for opening a font for output. Do NOT select the Append font option. Next, enter the Instances selection from the same menu, and load in the first character (A or whatever). After it loads, you will be dropped to the graphic window.

At this point, it is a good idea if you establish a "baseline" first. The baseline is the bottom line that you will use (not physical) for placing the characters. Characters with descenders (which are the hardest to center), can be easily line up if you adjust them according to that line (remember such characters make look odd, what with all that empty space above them, but that is the way they should look - TI-Artist and Font Writer look at a font from the upper left hand corner).

Using the Move Picture keys of the editor, you can easily move the picture left, right, up and down to center it on that imaginary line (use the block boundary markers to avoid confusion). After the picture of the character is centered, enter the menus again, and again select the font options. Then select the option to save a picture in a font. The Editor will ask you what ASCII character it represents, and then a white cursor will appear on a screen showing the picture. Position this cursor, with the arrow keys, on the lower right corner and press Enter. The program will automatically save it to disk in the font file you specified as the ASCII character you specified. Do this over and over until all your instances are converted to a font.

The advantages of this system over the one mentioned by Mr. Coffey (which, while still a good system and not requiring Font Writer) is that it is much faster, and you can center the characters in the font a lot easier since Font Writer's editor has tools for it.

Converting regular clipart from GRAPHX to TI-Artist, of course, is a much simpler procedure since all you really have to do is paste it on a screen, convert the screen to TI-Artist, and save each individual picture as an Instance or Slide.

The reverse process, converting TI-Artist to GRAPHX, is very simple. All you have to do is get whatever you are converting onto a screen, save it to disk, convert the screen to GRAPHX (again using TI-Artist's conversions utility) and then paste it into a clipboard from a GRAPHX screen.

Regarding the legality of converting art from GRAPHX to TI-Artist; I'm not sure what the policy of other manufacturers is, but ours is that once you buy the stuff, it's yours. You can convert it to any format you like. However, remember that the works in our GRAPHX Companions and Artist instances series ARE copyrighted (they are in fact the product of literally thousands of hours of work - a single font may take up to 10 hours to draw with GRAPHX!), and you can't give them out to anyone else. You can convert them for your use, but no one elses.

Font Writer is also copyrighted to J. Peter Hoddie and is manufactured and distributed by Asgard Software. The use of it as described here is only one of the many functions of the product.

Copyright 1987 - Chris Bobbitt
May be reproduced freely if unaltered
All Rights Reserved

Disk Controllers - from TI to MYARC

Copyright Jerry Coffey, January 1987

The views expressed in this article reflect the author's personal experience with TI, Corcomp, and Myarc disk controllers. Technical data has been verified wherever possible, but is not publicly documented in some instances. Please bring any errors to the attention of the author.

The disk capacity of the TI99 has increased in just a few years from less than 80K (a single one-sided 35 track drive) to almost 2.9 megabytes (four double-sided, double-density, 80 track drives). The early standalone was replaced by the PEBox system which would support three double-sided 40 track drives (540K). Corcomp introduced their four drive double-density system (1440K), followed by Myarc's similar system with two double-density formats (1280K and 1440K). Then in 1986, Myarc offered its 00 track upgrade which doubled capacity again. Even as capacity was increasing rapidly, the TI and Corcomp controllers differed only modestly in I/O speed. When MYARC introduced its fast DSDO controller, few reviewers did justice to its speed advantage. Early comparisons were done at the standard TI or Corcomp interface, but the big speed gains required taking advantage of the much tighter sector interlace possible with the high-speed MYARC card. To understand how this works we need to take a look at the way a disk drive

Disk Drive Fundamentals

A floppy disk drive writes information in concentric rings called "tracks" on a thin plastic disk coated with a film of magnetic particles. Each track in turn is divided into blocks of information called sectors. A blank disk has one (or more) index holes used to synchronize the process of writing to and reading from the disk. The type with many holes are called "hard sectored" since each sector has its position fixed by an index hole. The type of disks used by most computers have only one hole and are called "soft sectored". In this system the computer must write magnetic signposts on the disk to mark out each sector in a process called "formatting" or "initializing" a disk. These signposts take up a substantial fraction of the space on a track since they include not only sector numbers but buffers (filler bytes) that allow the computer to get into synchronization to read or write sectors of data and to prevent the sector identifier from being overwritten by a drive operating at a slightly different speed from the drive that formatted the disk.

The typical 5.25 inch disk drive has a "stepper motor" capable of moving the drive's read/write head(s) in or out along a radius of the disk in steps of

1/48 of an inch (thus the terminology "48 tpi" = 48 tracks per inch). Since the inner tracks have a smaller circumference, they crowd the bits of information together. Magnetic coatings on a floppy disk are rated by their capacity in bits per inch at standard magnetic flux for the write head. This figure is usually over 5000 bpi for modern floppies, but was somewhat lower a few years ago. The circumference of the inner track of a 40 or 80 track disk is about 10 inches -- which allows about 6250 bytes to be written on the track without exceeding 5000 bpi. For comparison, the Corcomp double density format requires over 6400 bytes per track. Media limitations were the reason that some early 5.25 disk drives only used the outer 35 tracks. The 16 sector (by 256 bytes/sector) format recommended by most drive makers requires only 6250 bytes per track and includes several hundred additional "buffer" bytes to compensate for differences in drive timing.

Timing is EVERYTHING

With soft-sectored disks, the integrity of the read/write processes require critical timing. The disk rotates at 300 rpm within a small margin. This means there are about 250 thousand magnetic pulses (bits) passing beneath the head each second. In single density format, the majority of these pulses are timing or filler bits -- in double density, many of the timing bits are suppressed in order to double the rate of data bits. In a typical sector read the drive must bring the disk up to speed, recognize the index hole, step out to track zero (to get its bearings), determine single or double density, verify its position, step in to the target track, verify the track number (written in the format operation), detect the sector identifier as it flies past, then immediately read the 256 data bytes into memory. Five of these operations require accurate reading of the magnetic pulses whizzing by at over 250K bits per second.

If you do some quick arithmetic (256 bytes/sector = 2048 bits/sector into 250K bits/second)... hmmm... Why can't the drive read a 125 sector file in one second? Well first many of those bits are not data bits, they are overhead to keep things synchronized and allow for timing variation between drives. Second, some time is used moving the head from one track to the next when more than one track must be read. Third, 250K is the instantaneous read rate and the computer must take time to do other things like move the last sector out of its buffer to make room for the next one. In the standard TI protocol for reading a disk, the data is moved into VDP ram (so the drive could be used without the memory expansion) before it goes to the expansion memory. All this thrashing eats great chunks of the time available for reading data. By the time one sector is safely tucked away in the 32K card, several sectors have already passed by the drive's read head. If the sectors were written consecutively on the disk, we would have to wait a full revolution (0.2 seconds) before the next sector would pass under the head. To avoid this inefficiency, the consecutively numbered sectors are spaced out around the disk so that they are separated by just enough time to take care of other business. The actual pattern in which the sectors are scattered is called the "interlace". The idea of the interlace is to spread the sectors out to match the timing needs of the hardware -- both the time needed to slash each sector and the time needed to step from one track to the next and get the the head settled down for some serious (250K bps) reading.

Interlace and Head Step Times

Life was simple with the TI disk controller. Both the interlace and the head step time were locked into the controller's PROM (that's the programmable chip that contains the control programs for the card). The head step time is the built-in delay between step signals to allow the stepper motor to move the head one "click" in or out. The TI settings are very conservative (read "slow") to allow for slow drives. The step time is 20ms -- if you step from track zero to track 39, it takes 20x39=780ms, almost four revolutions of the drive. The TI interlace lays the sectors down on a track in the order 075318642. This allows all sectors to be read in four revolutions of the disk though the slow head step lets another revolution go by between tracks. Thus the maximum read rate is about 9 sectors per five revolutions (= one second) or 2304 bytes per second.

When Corcomp designed its double density disk controller, allowances were made for the increased speed of later drives by permitting the step rate to be set with DIP switches for each drive. The step rates available are 30, 20, 12, and 6ms (the faster values quoted in the CC manual are referenced to

the wrong clock speed). They also provided a choice of interlace options, though only a couple of them are practical. The default interlaces are labeled "7" for single density and "10" for double density. The single density interlace is the same as TI's, but with a faster step setting the head can be moved without losing a revolution and thus reads 20% faster than the TI controller. The double density interlace allows 18 sectors to be read in five revolutions, but it doesn't leave enough margin to slash the last sector and step the head in time to catch the zero sector of the next track (that's why the sector number "hangs" for 0.2 seconds each 18 sectors while verifying a formatted disk -- you are seeing the extra revolution needed to acquire the first sector of the next track). Thus the maximum read rate is 18/1.2 or 15 sectors per second, about 67% faster than the TI controller. Users of the CC controller have probably noticed that it loads its own MANAGER program faster than this. In this case a special loader bypasses VDP and loads directly to CPU RAM -- this faster handling of the data allows the stepper motor to be activated sooner and saves one revolution per track (so the 98 sector file can be read in about 5.5 seconds). This provided a foretaste of the speed that MYARC would achieve with its double density controller.

The MYARC controller bypasses VDP RAM to load directly to CPU RAM. This technique coupled with a buffer RAM chip on the controller card provided a quantum jump in disk I/O speed. The MYARC card reads the TI single density interlace at 11.25 sectors/second (the same as Corcomp) and reads the CC 18 sector/track interlace at 18 sectors/second (the same speed Corcomp reads its MANAGER program), but this is only the beginning. Since the hardware empties its sector buffer faster, consecutive sectors can be placed closer together allowing a track to be read in fewer revolutions, i.e., it supports a faster interlace. With fast drives, the 9 sector/track single density format can be read at interlace "2". (NOTE: In the MYARC terminology, the interlace number represents the number of disk revolutions required to read a track.) This works out to 22.5 sectors/second compared to 9 for the TI and 11.25 for the CC controller. The MYARC 16 sector format can be read at interlace "3", 26.67 sectors/second -- 3 times as fast as the TI controller and almost twice as fast as Corcomp double density. The Corcomp 18 sector format can be read at interlace "3" or "4", but the data rate is the same in either case, 22.5 sectors/second. Interlace "4" is smooth but requires a very quick head step, interlace "3" reads the track in 3 revolutions but forces an extra revolution for the step from track to track because sectors 17 and 0 are adjacent on the disk. Though both interlaces have the same data rate, interlace "3" is safer if you are uncertain about the speed of your stepper motor.

In order to read and write both double density formats, the MYARC system must insert an additional step in some I/O operations -- sector zero must be read to determine whether a double density disk has 16 or 18 sectors per track. This datum is needed to convert the the logical sector numbers used by the TI operating system into track and sector-within-track addresses for the floppy disk controller chip. The TI and Corcomp controllers do not need this step because they do not use the full potential of the TI disk I/O protocol. Once this step, accessing sector zero, is added to the various disk operations, it opens the system up for using more than two formats -- including 80 track formats.

Beyond Double Density

A two format system can be managed using only the floppy disk controller's inherent ability to sense single and double density recording patterns. To get beyond this limitation, the additional data stored in sector zero must be read, stored, and used to modify the special binary commands sent to the FDC (floppy disk controller) chip. Fortunately the TI99/4A system design already provides for such innovations through the Device Service Routine concept and standard "GPL" calls. The system doesn't care what hardware is attached as long as it plays by the rules -- an interface program stored in a memory chip (PROM) on the peripheral device does the trick. This program handles calls for I/O operations from other programs such as TI Writer or the Basic Interpreters. Another set of rules controls the way disk and file information are saved on a disk. Disk parameters are stored in sector 0, while sector 1 must have a two byte "pointer" (a hexadecimal sector address) for each block (one sector) containing the bookkeeping data for a file. It is these blocks that are scanned in order to display the disk directory.

Since the Myarc controller must read sector zero to determine the number of sectors per track, the other parameters in that sector are available to control other variables such as number of tracks. But there were other

limitations to overcome. The number of files on a disk is limited by the space available for pointers. 256 bytes at 2 bytes per pointer would give 128 files — except the pointer list must end with a null word (>0000) so directory routines know where to stop — so we get 127 files per disk. The pointer itself can address sector numbers as high as 65535, so this is no problem. The real limitation is the bit map in sector 0. It begins at byte 56 leaving only 200 bytes or 1600 bits available to map the disk. Since a bit must be turned on for each sector used, the 1440 sector DSDD 40 track disk is already near the limit. The answer devised for the 80 track DSDD system is to map two consecutive sectors with each bit. It wastes some space but no more than systems that use a standard 512 byte sector.

Making the Quad System Work

So now lets say we have new code in the disk controller EPROM (an "erasable" version of the PROM chip used by TI) that does all the proper tricks with the bit map and has the FDC commands to control the new 80 track drives we have added to the system. We still have to tell the controller which drives are 80 track and find a disk manager program that can use the new commands. The collection problem can be taken care of using the DIP switches on the card (but in the process you lose their original function — setting step speed). Since the Eprom responds to standard GPL calls, most functions can be handled by the TI Disk Manager 2 cartridge. The exception is the disk formatting process — the formatting works OK, but the initial data written into sector zero is for the standard bit map. (This can be fixed by changing byte 56 from >03 to >01 with a sector editor.) Read/write operations from XB or TI Writer work fine since they use the GPL protocols. Myarc has an excellent disk manager program that works beautifully with 40 track drives, but it has suffered from a number of subtle bugs in 80 track mode. This program, like many others designed for high speed I/O, uses assembly language code to handle the FDC — bypassing some of the routines in the EPROM. Differences in bit map handling, even slight differences in execution times can affect the performance of 80 track drives. The code in the 80 track EPROM has had a lot of attention to proper timing — the price you pay for higher performance.

Fine Tuning the Myarc Disk System

Before you start using the Myarc system routinely, there are some experiments that can get maximum performance from your drives. Use the Myarc disk manager to try different interlace settings — first with your 40 track drives, then with the 80 track drives. Watch for hesitations as each formatted disk is verified, then use the Test option to read the sectors you have layed down. Look and listen for "retries" — when the sector number pauses with a head seek noise. Use the best disks you have and note the combinations that test smoothly. With fast drives in good condition, you should be able to run 9 sector (single density) format at interlace 2 and 16 or 18 sector double density format at interlace 3. Don't worry if 18/3 pauses at the end of each track — this is just the extra revolution forced by having sectors 17 and 0 adjacent on the disk.

When you try this with 80 track drives, don't be surprised if the results are different. The time required for the head to settle into a wide standard track may not be adequate to get it reading properly from the narrow tracks on the quad drive. Such subtleties as erase delays and disk quality are also more critical on the skinny, low power tracks. My Mitsubishi 4853s (96 tpi) will support both 16/3 and 18/3 but are unreliable at 18/4, while my TEAC 558s support all three at 48 tpi. Don't take chances with any setup that is marginal. The error rate may be low, but it always seems to happen to a file that isn't backed up.

Hot Rodding

If you want to try for a little more speed, there are two more tricks you can use. The faster WD1772 FDC chip is pin compatible with the standard WD1770 supplied by Myarc. It will try to step the head at 2ms rather than the 6ms setting of the standard chip. (The 80 track EPROM automatically uses the fastest step speed available.) Many of the latest drives can step at 2ms or 3ms even though they are conservatively rated at 4ms or 5ms. The change is noticeable but may not be worth the high price of the WD1772 (it is not a

commonly used chip and is rarely discounted). The second fix is cheap and very useful for producing large quantities of copies. The FDC chip's automatic "write verify" function can be defeated by shorting one pin on the controller card to ground. This is best done with a switch so the verify can be enabled for normal operations. The effect of this modification is equivalent of the "turbo" option on the Corcomp controller and should be used only after testing.

Interface Patterns

Notes: The configurations marked * and ** are the standard interface patterns for TI and Corcomp formats. The end-of-track intervals are only approximate since the 9 and 16 sector formats include more buffer space than the 18 sector format.

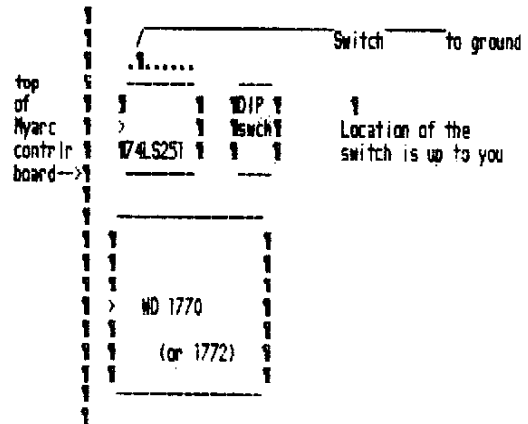
Sect/trk	Interlace	Pattern (dashed line is time available for head step)
9	4 *	0 7 5 3 1 8 6 4 2
9	2	0 5 1 6 2 7 3 8 4
18	5 **	0 11 4 8 15 1 12 5 9 16 2 13 6 10 17 3 14 7
18	4	0 9 5 14 1 10 6 15 2 11 7 16 3 12 8 17 4 13
18	3	0 6 12 1 7 13 2 8 14 3 9 15 4 10 16 5 11 17 none
16	5	0 13 10 7 4 1 14 11 8 5 2 15 12 9 6 5
16	3	0 11 6 1 12 7 2 13 8 3 14 9 4 15 10 5

16-sector patterns are not precisely to scale

Disk Controllers — an Addendum March, 1987

I mentioned a "turbo" modification to lock out the "read after write" (write verify) routine usually performed by the controller. Here are the details:

Find the 74LS251 chip at the top center of the controller board, above the DIP switches and beside the large FDC chip (marked WD1770). Solder a wire from the number 2 pin of the 74LS251 through a switch to ground (e.g. the wide trace of the DIP switches or any trace connected to that wide trace). It looks about like this from the bottom (non-component side) of the board.



As always you proceed at your own risk. (One person has told me this did not work on his 40 track system, but I haven't verified that.) You can tell it is working if your controller writes as fast as it reads

(normally the write takes twice as long).

Since I wrote the article on disk controllers, I have discovered some surprising facts about my own system. All of the Myarc timings in the article were done on an 80 track system with the fast WD1772 controller chip (stepping at 2us). With some help from Paul Charlton and Richard Roseen, I recently customized Paul's Eprom to step at 3us using the WD1772. (Richard's drives were making errors at the faster speed.) I used a Mechatronics Eprom programmer to download the Eprom code to disk, changed the FDC commands with a sector editor, and wrote the altered code back to a fresh Eprom. The process is simple (and cheap) once you decide what code you need in the Eprom.

The slower step speed made it possible to notice some slight differences in the performance of the WD1772. The first thing I noticed was that interface 4 on 18 sector tracks was no longer smooth - it was missing the first sector after a track seek and forcing an extra revolution of the disk. This was the first clear indication of how close this format is to the "ragged edge". The reaction to the small change in step speed implies that this interlace comes within 5% of the minimum time required to step and settle the head. Thus the likelihood of read/write errors is relatively high with this interlace. It will occasionally detect the sector 10 and begin to read or write before the head has completely settled. This interlace should definitely be avoided - 18/3 is both faster and more reliable.

The Eprom modification itself was an interesting experience. I patched the new FDC commands into some unused text bytes and patched addresses into the code to point to the new locations. The Mechatronics Eprom Programmer is an excellent piece of equipment. It will burn (program) a 2764 (8k) in about 90 seconds using the fast algorithm. I have talked to Jim Horn and Jeff Guide about offering an Eprom service to the customers of Disk Only Software. There are many possibilities this technique opens up. There is the 80 track modification for the TI controller worked out by Andy Cooper. And many Myarc owners are still using old Eproms that have never been upgraded (though this situation has improved since In: Phillips increased his production capacity). The fix we developed for Richard's controller can provide the optimum step speed (3us or 5us) for different disk drives using the WD1772 FDC chip. And any enterprising programmer can get his tailor-made code installed in nonvolatile memory.

Last Minute Addition

Hardware for Sale

- 1- CC40 system - (BK CC40 with Hexbus Watertape, RS232, Modem, and Word processor Cartridge) \$250.00
 - 1- Hexbus 5 1/4" Disk Drive Controller and drive. For use with CC40, 99/8, or 99/4A with Hexbus Adapter. \$175.00
 - 1- Horizon Ramdisk - Doublesided 180K Version Already Built and Ready to Run - \$200.00
 - 1- Mechatronics Gramkante - 128k w/ Software and Manual \$225.00
 - 1- TI PLODE Card w/docs \$100.00
- contact Richard at (214) 328-9257

THIS SPACE FOR RENT

PUT YOUR ADS HERE IN THE FLUG TI NEWSLETTER!



FULL PAGE - 14 DOLLARS
 HALF PAGE - 10 DOLLARS
 QUARTER PAGE - 6 DOLLARS
 Personal ads are FREE (MEMBERS ONLY)

FLUG MEMBERS GET HALF OFF!
 All funds support newsletter!



SALE PRICES GOOD THRU Sept. 30, 1987

Armado BYTES

Box 980921--Dallas, TX--75398

Summer Specials

Hardware

	REG. PRICE	SALE PRICE
MYARC 9640 COMPUTER	445.95	439.95
MYARC DISK CONTROLLER	145.95	139.95
MYARC RS232 INTERFACE	79.95	76.95
MYARC 512K MEMORY CARD	239.95	234.95
AVATEX 1200 BAUD MODEM	95.00	90.00
MECHATRONICS		
128K GRAMKARTE W/SOFTWARE	199.95	189.95
128K STANDALONE W/PIO	129.95	119.95
EPROMMER	129.95	119.95

Software

TI ARTIST	19.95	17.95
MG DISKASSEMBLER	19.95	17.95
FONTWRITER	24.95	22.95
TOTAL FILER	24.95	22.95
THE PRINTERS APPRENTICE	22.50	19.95
TPA FONTS DISK #1	11.50	10.00
RAPID COPY	14.95	13.95

Used Equipment

QUAN.	DESCRIPTION	PRICE EACH
1	TI EXPANSION BOX	\$ 130.00
1	99/4A CONSOLE	45.00
4	TI DISK CONTROLLERS	75.00
1	TI PCODE PASCAL CARD	100.00
1	ORIGINAL TI PRODUCT-NEVER RELEASED- IEEE-488 INTERFACE CARD - ONE OF A KIND	250.00

Now Available

REPAIR SERVICE - 99/4A CONSOLES /
MOST REPAIRS - ONLY \$25 - CALL FOR
MORE INFORMATION

Call Us At 214-328-9257

ADD 4 % OF TOTAL ORDER FOR SHIPPING AND HANDLING

-SORRY, NO CREDIT CARD ORDERS-
TEXAS RESIDENTS ADD 7 1/4 SALES TAX

Free Catalog

FOREST LANE USERS GROUP--INFORMATION

Officers for 1987

PRESIDENT	RICHARD FLEETWOOD
VICE PRESIDENT	OSCAR SMITH
SECRETARY	KEITH JOYNER
TREASURER	RON KUHLMAN
CORRESPONDENCE	WILSON TAYLOR
PROGRAM MEETING CHAIRMAN	JAMES CARSON
TIBBS SYSTEM OPERATOR	RICKEY MORGAN

Committee members for 1987

NEWSLETTER COMMITTEE	
Resource Manager	ROY WILLIS
Newsletter Editor	WILSON TAYLOR
Printing and Formatting	RICHARD FLEETWOOD
Mailout Chief	TERRIE MORGAN
SOFTWARE LIBRARIAN	RON SCHVAB
MEMBERSHIP CHAIRMAN	TERRIE MORGAN
NEWSLETTER LIBRARIAN	WILSON TAYLOR

PLEASE SEND ALL CORRESPONDENCE AND EXCHANGE NEWSLETTERS TO THE CLUB AT:

The Forest Lane Users Group
P.O. Box 743005
Dallas, Tx 75247

FOREST LANE USERS GROUP TIBBS INFO.
(214) 398-7162-24 HOURS A DAY
300 or 1200 BAUD

THE "ULTIMATE" TIBBS--Possibly the most advanced and useful bulletin board in the country for the serious TI 99/4A user. We support QUAD DENSITY with two quad density/doublesided drives and two double sided/double density drives, based around our own designed disk controller card. We also have one HORIZON RAMDISK card, and the TIBBS Clock Card. The program itself is possibly the most advanced, most changed TIBBS around, and it makes use of all TI features--Graphics, Sound, and Color. We also support XMODEM transfers, with over 130 files to download at this time, with many more to come. To log on, just call the above number, with your modem, hit 'N' and press <ENTER> then follow the prompts. After you have gotten your user number, you will be upgraded to regular user level within 24 hours.

MEMBERSHIP INFORMATION

A Forest Lane Users Group membership includes a monthly newsletter mailed to your home, full access to the FLUG TIBBS, access to the FLUG software library of public domain and FAIRWARE programs, and several other benefits. Annual membership is \$15 a year. For more information please contact the members below whose phone numbers are listed.

MEMBER NEWSLETTER INPUT

Please send all newsletter input, articles, copy, pictures, files, or news to RICHARD FLEETWOOD at the FLUG PO BOX. Disk files are appreciated, or files may be uploaded to the TIBBS. NOTICE--all input must be received no later than TWO weeks before the meeting in order to make that month's issue.

FLUG ADVERTISING RATES

All personal classified advertisements for members will be printed at no charge. Commercial advertisements will be charged the following rates--1/4 page minimum only--Fee per issue--

	<members>	<Non-members>
1/4 page ad	\$ 4	8
1/2 page ad	\$ 5	10
Full page ad	\$ 7	14

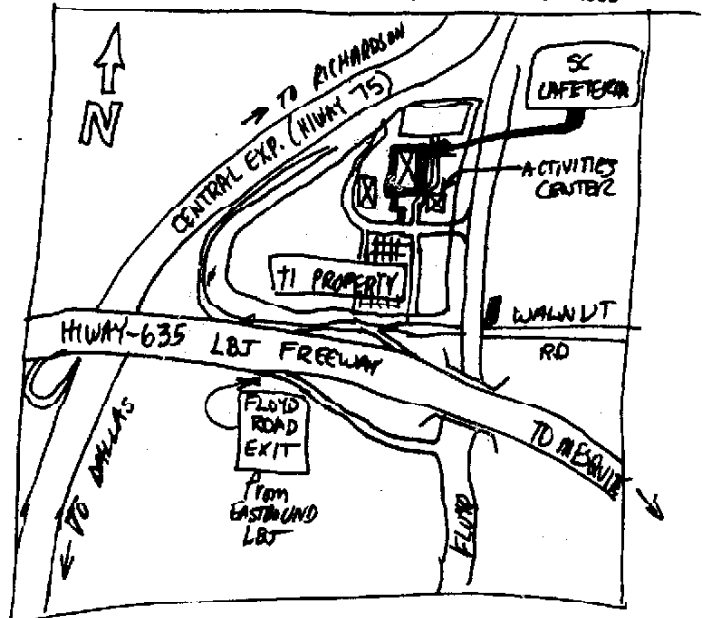
ANY ARTICLE APPEARING IN THIS NEWSLETTER MAY BE REPRODUCED FOR PUBLICATION IN ANOTHER TI USERS GROUP NEWSLETTER, AS LONG AS THE AUTHORS NAME AND USERS GROUP NEWSLETTER AFFILIATION ARE KEPT INTACT. WE ENCOURAGE EXCHANGE NEWSLETTERS FROM OTHER USERS GROUPS. PLEASE FEEL FREE TO DUPLICATE THIS NEWSLETTER, BUT ONLY IN ITS ENTIRETY IF YOU WISH TO MAKE COPIES AVAILABLE TO YOUR MEMBERS. ALL OPINIONS EXPRESSED WITHIN THIS NEWSLETTER ARE THOSE OF THE AUTHOR OR AUTHORS, AND DO NOT NECESSARILY REFLECT THE OPINION OF THE FOREST LANE USERS GROUP, ITS OFFICERS, OR ITS MEMBERS. ALL INFORMATION PROVIDED WITHIN IS EXPRESSLY FOR THE USE OF EXTENDING THE USEFULNESS OF THE T199 TO AS MANY PEOPLE AS POSSIBLE. NO WARRANTY OR GUARANTEE IS MADE OR IMPLIED AS TO ANYTHING YOU MIGHT USE. WE ARE NOT RESPONSIBLE FOR ANY DAMAGE OR PROBLEMS CAUSED BY MISUSE OF INFORMATION FOUND WITHIN THIS NEWSLETTER.

FLUG MEETINGS

FLUG meetings are held on the FIRST SATURDAY of every month at the main TI plant on CENTRAL EXPRESSWAY and LBJ Freeway in North Dallas. To get to the Plant, take LBJ EAST from Central Expy, and take the first exit, which will be FLOYD ROAD. Follow FLOYD ROAD around to the STOPLIGHT at WALNUT and FLOYD. Go NORTH thru this light, and then take the FIRST ENTRANCE to the LEFT into TI. You will pass the TEXINS ACTIVITY CENTER on your right AFTER you have turned off on FLOYD. After about 100 FEET you will come to a STOP SIGN. Go straight ahead (DUE WEST) and you will see a covered walkway on your immediate RIGHT. A sign hanging off the END of the covered walkway will say EMPLOYMENT CENTER. Just park across the street from the covered walkway, and then follow the walkway back into the trees. You will come to the SC CAFETERIA building, which also contains the TI HEALTH CENTER, and the TI EMPLOYMENT CENTER. Take the door on the left, then follow the signs to the conference room in the BACK of the CAFETERIA. The meeting starts promptly at 2 PM, so try to get there early.

MEMBERS TO CONTACT FOR MORE INFORMATION

Richard A. Fleetwood (6 to 10 pm) (214)328-9257
Oscar Smith (6 to 10 pm) (214)227-3259
Ron Schvab (6 to 9 pm) (214)234-4553



Meeting this coming
SUNDAY
at 2 PM. - Please be there!

NOTICE

THERE IS GOING TO BE AN
AUCTION
OF TI 39/4A COMPUTER HARDWARE
AND SOFTWARE AT THE

SEPTEMBER MEETING
OF THE
FOREST LANE TI USERS GROUP

PLEASE DONATE YOUR UNUSED
OR SPARE equipment to help
raise money for the TI FAIRE
to be planned for October 19
PLEASE SEE INSIDE FOR LOCATION/TIME

If Unavailable, Return to:

**FOREST LANE TI
USERS GROUP**



**P. O. BOX 743005
DALLAS, TEXAS**

75847-3005

PLEASE DELIVER TO:

**MIKE STANFILL
2330 JONESBORO
DALLAS, TEXAS
75228
(1/88)**

TIBBS-214-328-4880