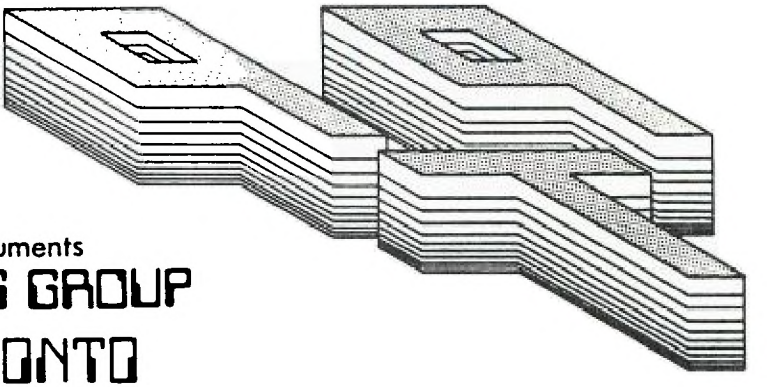

Texas Instruments

USERS GROUP TORONTO

FOR THE TI-99/4A COMPUTER

NOVEMBER/DECEMBER 1984

NINE T NINE USERS GROUP

55 CORDELLA AVE.
TORONTO, ONTARIO
M6N 2J7

CANADA 32

Edmonton User's Group
P.O. Box 11951
EDMONTON ALTA T5J 3L1

EXECUTIVE COMMITTEE

PRESIDENT	**Lloyd Lindsay**	**(743-3868)**
VICE-PRESIDENT	**David Huggett**	**(438-4020)**
SECRETARY	**Mike Mattos**	**(763-3244)**
TREASURER	Gary Willert	(276-1076)
OFFICER AT LARGE	Boyd Brown	(793-3761)

NEWSLETTER EDITOR

David Huggett

LIBRARY COMMITTEE

Vee. Papadimos 491-8602

MEMBERSHIP FEES

FULL MEMBERSHIP	$25.00/year
ASSOCIATE MEMBERSHIP	$12.50/year

All memberships are household memberships. An associate membership is only for those who live beyond the commuting distance of Toronto but who wish to receive our newsletter and have access to our library. You are welcome to visit one of our general meetings before joining the group. If you wish more information contact our secretary in writing at the club address on the front cover or call and leave a message with his answering machine.

NEXT MEETING

The meetings are held on the last Tuesday of each month. Since the last Tuesday of December falls on Christmas day this newsletter is a November/December issue and the December meeting is on the eleventh. The next meeting will be on Jan 29 1985 at Shoreham Public School, 31 Shoreham Drive in Downsview, starting at 7:30 PM. Shoreham Drive runs east/west from Jane street north of Finch. The entrance to the school is a few hundred yards east of Jane on the south side of Shoreham.

COMMERCIAL ADVERTISING

Any business wishing to reach our membership may advertise in our newsletter. The rates are as follows, (width by height):

FULL PAGE (6" X 7.5") $40.00
HALF PAGE (6" X 3.5") $20.00
QUARTER PAGE (3" X 3.5") $10.00

Please have your ads camera ready and paid for in advance. You may contact the Treasurer for more information.

NEWSLETTER ARTICLES

Members are encouraged to contribute to the newsletter in the form of articles, mini programs, helpful tips, jokes, cartoons or even questions. A short article may be submitted in any form but if you don t have TI-WRITER it may be better to type the longer ones into your computer using REM statements, and submit the cassette or disk, which will be returned. We welcome the reprinting of any article appearing in this newsletter providing credit is given to the author and to 9T9. If more information is required, call Gary Willert.

DISCLAIMER

PRESIDENT'S REPORT

By Lloyd Lindsay, CA

Our club is continuing to grow and most of our newer members have found out about us from Texas Instruments or from the Home Computer Magazine.

Dave Huggett, our editor, has recently purchased a business system and will soon be leaving us as he devotes more of his time to his new equipment.
Dave has done a terrific job putting together the newsletter as well as providing us with technical articles and advice. I hope that he will continue to provide us with his technical knowledge after he has left.

With the increase in membership and Dave's departure we must have additional members on the Executive. It is your club and we need your participation if the Club is to continue. Please contact Boyd Brown if you are willing to join the executive. You will learn more about programming and equipment at one Executive meeting than most people learn in a whole year.

Sandro Lorenzen, a technical representative from Texas Instruments arrived at our November meeting as promised and did supply us with some useful information on TI 99/4A hardware. He will be attending as many future meetings as possible, and will keep us informed of happenings in the TI 99/4A hardware field. Remember that Sandro is not a software specialist, so please, no questions in this area. The swap meet went down very well, albeit a little less organised than last years. Nevertheless, most people got what they wanted and those of us who were selling had a profitable evening. This was an excellent opportunity for purchasing hardware, firmware and software at prices that were for the most part very reasonable. I beleive that at least two copies of Extended Basic were traded. At present it is almost impossible to purchase this item in the Toronto area. Extended Basic availablily is considered to be essential for the survival of this computer. Sandro has told me that there is a good supply of Extended Basic Modules available at;

Triton Products
P.O.Box 8123
San Francisco, California 94128

The price is US$ 99.95 and the company has a toll free number; 1-800-227-6900

LIBRARY PROGRAM SPOTLIGHT

Multiplan

by Lloyd Lindsay

Our library has a new version of Multiplan which works much better than the original. The program diskette must be named TIMP before it will run.
With this version your cursor will continue to move from cell to cell as long as you keep the function and arrow keys depressed. This speeds up the movement of the cursor and you now have a spreadsheet program as powerful as most of the ones used with other microcomputers.
To speed up the entry of data you should turn off the automatic Recalculation. Otherwise, you will waste your time watching the program recalculate the spreadsheet each time you make an entry.

HERE'S HOW

by GARY WILLERT

BRANCH??? I DON'T SEE ANY TREES!

There are a few areas that seem to be perennial favourites for questions. I don't mean that the same people ask questions about these areas repetitiously (although some do), but rather that there always seems to be SOMEBODY that needs help in dealing with these areas. Three that come immediately to mind are branching, formatting and sorting. Therefore, in the upcoming months, this column will deal with these subject areas starting this month with branching techniques.

Since the purpose of the "HERE'S HOW" column is to provide advice for beginning to intermediate programmers, all of the program examples that I use are in BASIC or Extended BASIC and most of the references used are also to those languages. However, it is important to remember that BASIC is not the only language in the computer world. Indeed, one of the most controversial topics today among educators is the choice of language to use in the classroom.

One of the features of BASIC is the unstructured approach that is permitted. By this I mean that the language does not FORCE you to write your program in a certain way; in BASIC, as long as you don't make any technical errors in your use of each specific statement and follow a few simple rules such as closing your loops (even this is not always necessary), your program will RUN. The problem may be that it will not run very efficiently and may not even do the job you want if you have been lax in your organization. So, since the language does not force organization upon the programmer, he must exercise a degree of self-discipline to ensure that his programs run as well as possible.

It is no secret that I, along with a great number of other programmers, favour TI Extended BASIC among BASIC languages. I am not going into a diatribe about the features of the various BASIC languages - that is outside the scope of the column - but it is worth noting in passing that the version of BASIC that we use is among the best. (Yes; I know it is slow, but I am talking about features.)

One of the features that makes Ext. BASIC so desireable is the availability of user-written SUBPROGRAMS. Let us, then, take a look at SUBPROGRAMS along with DEFINITIONS and SUBROUTINES.

If you were paying attention at the beginning, you will have noted that I promised to deal with branching. Strictly speaking, this is not true. I am certain that anyone who has read his manual knows how to use GOTO and GOSUB. Therefore, I do not intend to give instruction on how to use these commands but rather to illustrate that there are choices to be made about when to use them.

_GOTO!

GOTO, although much used and readily understood, is really a controversial command. A number of programmers feel that the command should not even be included in any programming language. This seems a little extreme, and I must admit that I use GOTO in my own programs, but it is a command that should not be used indiscriminately. The problem, you see, is that excessive use of GOTO makes it hard to trace a program when you are trying to debug it and time is wasted making unnecessary branches. A large number of GOTO commands in a program usually indicates a lack of planning. Instead of allowing the program to flow naturally forward, some programmers write haphazardly, putting sections of program in wherever tey happen to be when they think of them and then correcting the flow by branching to and from that area of the program with GOTO. Each such diversion necessitates the use of at least 2 GOTO statements.

IN GENERAL, any section of a program that is only accessed from one statement should be written as part of the main program.

_GOSUB!

GOSUB is the most common branching command. The advantage over GOTO lies in the RETURN statement that ends the subroutine. It is faster to RETURN to a stored branch point than to GOTO a line number. This same feature makes the subroutine able to be accessed from a number of different points in the program and to RETURN faultlessly every time.

IN GENERAL, any section of a program that is accessed from more than one statement can be written as a subroutine. In some cases, usually following a test, it is also advantageous to use a subroutine even though the subroutine is only accessed from one statement.
Subroutines are treated as part of the main program; any variables used are consistent with their use elsewhere.
_SUBPROGRAM!
Subprograms are one of the most poorly understood concepts among learning programmers. They are literally set apart from the rest of the program. In reality, what distinguishes subprograms is that very fact: they are NOT part of the main program.
Subprograms come in a variety of forms: built into TI BASIC is access to a number of assembly-language subprograms, or users can write and use subprograms either in Extended BASIC or in assembly language. The features of subprograms are perhaps illustrated by looking at one that is built in. Let's use KEY.
KEY illustrates most of the features that characterize subprograms:
1) It is accessed by "CALL ...". This characteristic is self-explanatory.
2) It has parameters that can be passed both in and out. The parameters in this case are _key-unit!, _return-variable! and _status-variable!. The key-unit is passed into the subprogram and can therefore be either a constant or a variable. Since the return-variable and status-variable are both being passed out, variables must be used to accept the passed values. It is possible to use a variable that will pass information both in and out of a subprogram; this will be illustrated below.
3) It can be accessed from any point in the program, using whatever variables are convenient or necessary. This point is related to the one above, but I put it separately because it represents, from the programmer's point of view, the most significant difference between subprograms and subroutines.
4) It resides outside the main program. This seems self-evident when it is applied to a feature that is built into the language, but the point actually applies to all subprograms: assembly-language subprograms are, of course, stored in the memory expansion and Ext. BASIC programs are found at the end of the main program. Although this does not give quite the convenience of building a library on disk and loading those segments required, it is almost as good: you can write a library of subprograms and store it on disk with the MERGE option. Then, when you require a subprogram in your work, merely merge the required entry at the end of your program. Subprograms may be placed in any order (since they are accessed by name) as long as they all come after the main program. REMARK statements may follow subprograms.
_DEFINITIONS!
Definitions are easy to ignore. They arenever referenced by name in the manual and sometimes it is difficult to see how the DEF command has any application.
Definitions are used when it is necessary to perform exactly the same action on a variable in more than one place in the program. Since only one statement is involved, it impossible to perform tests on the results. If tests are required, then you should be using a subprogram.

INTERESTING TRIVIA

by Boyd Brown

BOOKS for the TI 99/4A

Many of our new 99/4a console owners out there will have read the Beginners Basic book supplied with their sets, at least I hope they have, as this book is a well set out and informative document. At the back of this book, on page 143, there is some information on a book by Herbert D.Peckham on Basic Programming. The cost of this book is quoted as being $10.95 U.S. If this

item is purchased from a Canadian book store it could cost as much as $26.95. While it is a very well written and thought out book it has a lot of inaccuracies in the programme examples which can prove to be very frustrating to the computer tyro.
Other books are available and are on the shelves of most of the book stores in Toronto.
USING AND PROGRAMMING THE TI-994A by Frederick Holtz, Tab Books Inc. #1620 $14.50
This book not only has chapters covering Basic Programming, but it also extends to the history of computing, the architecture of the TI-994a, TI-994A Graphics, other programming languages. There is an overview of TI-994A hardware and software and a chapter on the conversion of other computer basic to TI basic. All in all a very useful and relatively inexpensive book.
I have also located two other books on the shelves of our book stores, both of these contain ready to run programmes which are accurately transcribed and should run after being keyed in.
THE TEXAS INSTRUMENTS HOME COMPUTER IDEA BOOK by David Ahl, Creative Computing Press at $11.00 Cdn.
Only one small chapter in this book is set aside for games, the rest of the book is dedicated to useful programmes in the field of science, probabilities, geometry, problem solving, compounding etc. A very useful book for the grade 8 and up student who has no programming skills.
36 TEXAS INTRUMENTS PROGRAMMES FOR HOME SCHOOL AND OFFICE by Len Turner, ARCsoft Publishers at approx $12.00 Cdn.
This book contains a number of ready to run progammes which are so simple that most of us should have been able to write them after digesting the information supplied with the basic console. However, for those of us who just want to read and type, they include programmes for the home, classroom and buisness. For example, there are programmes on, Salesmans Commissions, Hourly Wages, etc., etc. Add it to your collection of usefull trivia.
Finally, for information on just about anything to do with the TI-99/4A I strongly suggest a subscription to the new Home Computer Magazine.

WANTED DEAD OR ALIVE

A 100 Micro-Farads reward has been offered for the capture of Hopalong Capacity and his side-kick Eddy Current (possibly armed with electron guns), who escaped from their Weston Primary cell, where they had been clapped in ions. These men have not been seen faraday or so. They were charged with stealing Joules from a Volt and also with the induction of an 18-year old coil called Milli Henry who was found half-choked. They were last seen riding Megacycles over the Wheatstone Bridge. These Megacycles were of low frequency, so they robbed a man of an A.C. motor and returned ohm by a short-circuit. The man offered no resistance.
STOP PRESS...They eventually fell foul of the cunning Constable Mill-Amp of the Electromotive Force who brought them down to earth.

FOR SALE

One complete 99/4A system including, Console, Panasonic tape recorder

Perhipheral Expansion Box, TI Printer, Extended Basic, Multiplan, TI Writer

Memory Card, RS-232, One Disk Drive, all in like new condition

Make an offer on all or part, call 820-6762

EASY TO ASSEMBLE

by David Huggett

In the last article we introduced the beginning beginner to assembly language. We showed how to print our name to the screen and then how to print it to all of the screen locations, making it appear to move. I suggested to try and move your name from the top down and the bottom up simultaneously. The significance of this is to remember where the last screen location was as we switch from top to bottom. As is usually the case in Assembly, there are many ways to do this. Since we are learning to work with registers we will use them to store this information. Also, the computer stores and retrieves information faster from these registers than anywhere else in the memory. Looking at the sample program below, after the label START, you will see that we load these "memory" registers with the appropriate starting locations. We can also call them "buffer" registers, or anything you wish, they are just locations in memory that store information. But we have three buffer registers and only two starting locations, and the first register's value is twenty. I used that number so that when the names met in the centre of the screen they would meet on one line instead of between two lines. When we print from the top down we must erase the first character of the word on the screen, increment one space, then print the word again, in this case our name. This keeps the screen clear except for our name. When we print from the bottom up we must erase the last letter of our name, space to the left the number of letters in our name, then print it. That's why we need two memory registers for the bottom location. The rest of the program is self explanatory and when it is fully understood you may wish to try the counting program that follows this one. When you are comfortable with that one you will have initiated yourself into Assembly and should have less trouble continuing on from here. Good luck.

```
        DEF  START          name of program
        REF  VMBW,VSBW,KSCAN  utilities used
START   LI   R4,20          top starting location
        LI   R5,768         the end of the bottom starting location
        LI   R6,7           the beginning of the bottom starting location.
LOOP    MOV  R4,R0          move R4 into R0.      (text prints from left to right)
        LI   R1,32          load R1 with the space character
        BLWP @VSBW          write a space to the screen
        INC  R0             increment R0
        LI   R1,NAME        load R1 with text at the label NAME
        LI   R2,7           the length of NAME
        BLWP @VMBW          write NAME to the screen
        LI   R3,0           load R3 with zero to start delay loop
DELAY   INC  R3             \          increment R3 by one
        CI   R3,700          \ delay   compare R3 with 700
        JLT  DELAY           /         if less, jump to DELAY
        MOV  R0,R4          if not, store R0 in R4, (to retain last top position)
        MOV  R5,R0          put contents of R5 in R0 (now do the bottom position)
        DEC  R5             now decrement R5 (bottom moves to the left)
        LI   R1,32          put space character in R1
        BLWP @VSBW          write space to the screen (erase last letter of NAME)
        S    R6,R0          subtract 7 (R6) from R0 (length of name)
        LI   R1,NAME        load R1 with the text
        LI   R2,7           and R2 with length of text
        BLWP @VMBW          and write it to the screen
        C    R0,R4          see if the names have met
        JGT  LOOP           if not, start again and move both names another space
SCAN    BLWP @KSCAN         key scan routine
        MOVB @>837C,R0      has a key been pressed?
        JEQ  SCAN           if not keep on looping
        JMP  START          if yes, start again at the beginning
NAME    TEXT 'MY NAME       the text at the label NAME
        END
```

```
*                    COUNTING PROGRAM BY DAVID HUGGETT
*
       DEF  COUNT          put COUNT (the program name) in the definition table.
       REF  VSBW,VMBW,KSCAN  put these utilities in the reference table.
COUNT  LI   R4,>3000  ---tens      \          (together called the REF/DEF tab-->
       LI   R5,>3000  ---hundreds  \ buffer registers where we store the valu
       LI   R6,>3000  ---thousands / of what is in these three columns.
ONES   LI   R0,>172      location of ones column on the screen.
       LI   R1,>3000     zero character. VSBW only sees MSB, >30 = 48 (ASCII 0)
LOOP   BLWP @VSBW        write to the screen.      (MSB = most significant byte)
       AI   R1,>0100     add one, only MSB (the left two numbers) are affected.
       CI   R1,>3A00     have we gone past nine? >3A = 58 which follows nine.
       JEQ  TENS         if yes, go to TENS.
       JMP  LOOP         if not then jump to LOOP and repeat the process.
TENS   AI   R4,>0100     add one to the tens register.
       CI   R4,>3A00     compare R4 to the number following nine, ASCII 58.
       JEQ  HUND         if the comparison is equal then jump to the HUND label.
       LI   R0,>171      if not then load R0 with the screen location, and
       MOV  R4,R1        load R1 with the number in the tens register and
       BLWP @VSBW        "bullwhip" it to the screen.
       JMP  ONES         go do the ones again.
HUND   LI   R0,>171      tens column location on the screen
       LI   R4,>3000     reset the tens buffer.
       LI   R1,>3000     load the zero character.
       BLWP @VSBW        put zero in the tens column on the screen.
       AI   R5,>0100     add one to the hundreds register.
       CI   R5,>3A00     compare it to the colon (greater than nine).
       JEQ  THOUS        if equal go to the THOUS label.
       LI   R0,>170      screen location for hundreds column.
       MOV  R5,R1        load register one with register five.
       BLWP @VSBW        write the single byte to the screen.
       JMP  ONES         now do the ones again.
THOUS  LI   R0,>170      *
       LI   R5,>3000     * we could just use >30 but that is poor programming
       LI   R1,>3000     * practice as we will see at later levels.
       BLWP @VSBW        *\
       AI   R6,>0100     * \
       CI   R6,>3A00     *  \ same as above but in the thousands column.
       JEQ  STOP         *  /
       LI   R0,>16F      * /
       MOV  R6,R1        */
       BLWP @VSBW        *
       JMP  ONES         *
STOP   LI   R0,>16E      area where  (after 9999 is reached)
       LI   R1,NUM       10000 is put on the screen.
       LI   R2,5         R2 is for the number of bytes to be written as
       BLWP @VMBW        this is a multiple byte write command.
       LI   R0,>205      screen address where PROMPT begins.
       LI   R1,PROMPT    load into register one, PROMPT.
       LI   R2,23        number of bytes of PROMPT.
       BLWP @VMBW        display to screen.
SCAN   BLWP @KSCAN       *
       MOVB @>837C,R0    * similar to CALL KEY in basic.
       JEQ  SCAN         *
       LI   R0,>16E      screen address to start CLEAR from.
       LI   R1,>2000     equals ASCII 32 (the space character).
CLEAR  BLWP @VSBW        put one space character on the screen.
       INC  R0           goto next screen position.
       CI   R0,>21D      is it equal to the position of the "T" in REPEAT.
       JNE  CLEAR        if not keep putting space characters to the screen.
       BL   @COUNT       screen is cleared, start over again.
NUM    TEXT '10000'
PROMPT TEXT 'PRESS ANY KEY TO REPEAT'
       END
```