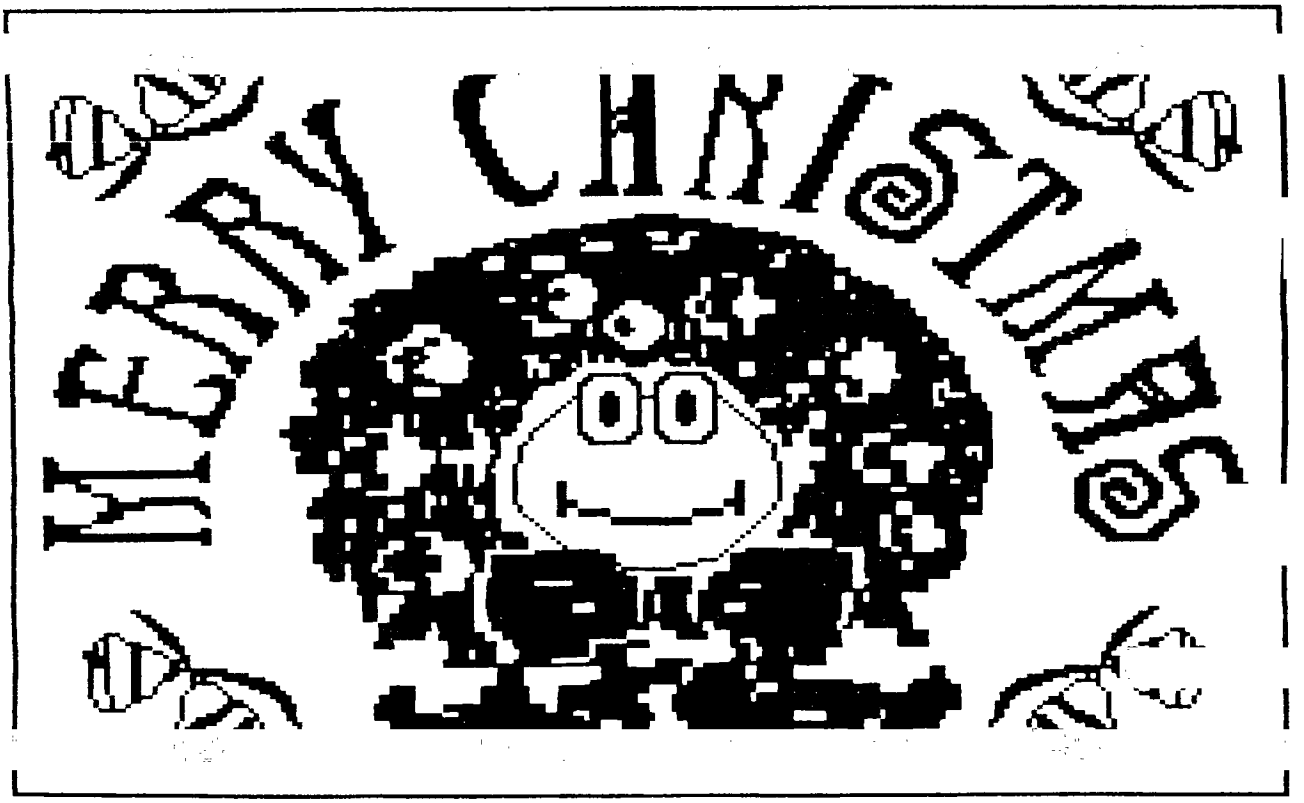
(065) 8912

The Ottawa TI 99/4A Users' Group

The Micro Mail

VOLUME 8 NUMBER 10.....December 1989



DON'T FORGET THE MEETING -- December 5, 1989

AND -- Remember to return your exchange newsletters!

P.O. BOX 2144,STATION D,OTTAWA
*** ONTARIO,CANADA K1P 5W3 ***

COMING EVENTS

December Meeting:	December 5, 1989 7:30 p.m.	Merivale High School
TI-BASE Workshop:	December 19, 1989	Bill Sponchia's home. Contact Bill Sponchia or Tom Bentley for details. No need to call if you were at the first meeting, but if you are a newcomer, please let Bill know so he will have an idea of what to prepare for.
January Meeting:	January 9, 1990 7:30 p.m.	Merivale High School
February Meeting:	February 6, 1990 7:30 p.m.	Merivale High School
March Meeting:	March 7, 1990 7:30 p.m.	Merivale High School
5th Annual TI-FEST	April 28, 1990	Merivale High School. Contact Ruth O'Neill for details, or to volunteer your help.
Newsletter Deadline:	December 15, 1989	(or any time before that!)

EDITOR'S NOTES
from Lucie Dorais

Since the November newsletter was late, everything else will be... Right? I have volunteered to do the December and January newsletters, until we find a permanent editor; I hope to get the January Newsletter around Christmas... Just don't despair! It is not that I don't like the job, it is that with my cover picture and monthly Extented Basic column, I just don't have the time to do everything!

Ruth has been a wonderful editor, and it is a tough act to follow. And I know, because I print the labels, that the stapling, mailing, etc. aspects of the job were always done with diligence, thanks to her and her helpers Charles Earl and Ralph Kuhn.

Since I want this issue to be in your (no doubt anxious) hands as soon as possible, it was done in a hurry; and being my first, I hope you will excuse some breaches to the tradition, like the President's wise words starting on page two instead of three...

THE PRESIDENT'S TWO CENTS' WORTH
by Charles Earl

As you've probably noticed, the November and December issues of the newsletter arrived late. Before you start to grumble, just pause and think about what you've done for your club lately. The newsletter editor's position is still available, even though Lucie Dorais has been kind enough to volunteer and do the December issue. Anyway, this time of year is also renewal time. The executive has gone over the options and decided that the membership price will stay at $25.

Jack McAllister has resigned from the position of Tape Librarian since he will be moving to Kentucky. Stephen Bridgett has stepped forward to take over the position. The cassette library seems to be severly underused, but I hope that is not the case in the future.

The November meeting was very interesting, so it's too bad the turnout wasn't better. Bob Boone gave a review of the Chicago show, and Lucie Dorais gave an excellent demonstration of TI-Artist plus. Michael Taylor showed of Gary Bowser's Diskodex program. Unfortunately, Jane Laflamme couldn't get her color printer working until the week after the meeting, I'm sure we will be hearing more about it in the future.

For the first time, the club offered a choice of two items for the raffle in November. The idea seems popular, and we hope to continue it in the future. The December meeting will offer the choice of TI-Artist plus or Tris. Both are commercial packages, the first from Inscebot and the latter from Asgard Software.

Well, it looks as if there is serious interest in setting up some new workshops in the beginning of the new year. There will probably be a Compuserve introduction -- the final date will be set at the January meeting. Dick Piche has agreed to do a ramdisk workshop, focusing on the construction of a horizon ramdisk, or adding the RAMBO modification. A programmers' workshop may begin in January too if more people show an interest.

Currently Bill Sponchia is hosting a TI-Base workshop on the third Tuesday of every month. If you are interested in TI-Base, or any other workshop, or even just have suggestions for future workshops, please give Bill a call.

BROWSING THE LIBRARY

with Dave Morrison

The Library's most recent offerings of Disk of the Month were Lucie Dorais' Extended Basic Tutorials (with complete programmes) that were printed in our Newsletters during the period 1987/88 and 1988/89. For those of you who were not inclined (or too lazy!) to type all those programme lines yourself, here is an opportunity not to be missed! More copies of Lucie's disks (SSSD DSSD) will be available at our next meeting or I can send them to you poste haste (?).

The other disk was a great programme by Rejean Felton, of Montreal's CIM-99 Users' Group, which allows the user to design small graphics in the CSGD format; named Graphics Editor, despite the title, it also contains a very fast label printing facility, with or without graphics, plus an option to print those little graphics, up to 10 side by side (very handy for reference!) Graphics Editor was recently demonstrated at one of our meetings and was very well received.

Lucie's programmes are public domain (but the tutorials are copyrighted), and Rejean's programme is Fareware: contributions to the author would not be amiss if you utilize the software. Even if you don't use any Fareware programme, a note of appreciation would be welcomed by the authors! Lucie Dorais and Rejean Felton can be contacted care of our Newsletter.

The Library needs new material for our Disk of the Month offering to continue! We have nothing new to offer for our January meeting, so unless something "turns up" before the first week in January, I will select a number of items from our Library; paste 'em together and we'll have a "mish-mash" to play with!

As the majority of you have (or have had and discarded) most of the Library files in you own Libraries, this is not what you want. So, will some kind soul or souls, please search your resources and try and provide something new very soon! To all of you, a Very Merry Christmas and A Happy New Year!

TI-ARTIST PLUS!
A review by Lucie Dorais

Until now, I was using three programs to design the cover pictures: GRAPHX for its zooming and fast printing functions, TI-ARTIST for its moving/copying functions, and JOYPAINT when I needed to use the "spray can" effect, or to rotate, reduce, or magnify a part of a picture. No more! The latest incarnation of TI-ARTIST, the one with a PLUS!, has changed all of that (except the printing: see below).

TI-ARTIST PLUS! is an update of TI-ARTIST v.2, so if you are familiar with it, you will find the same drawing and converting functions. But: the author, Chris Faherty, seems to have understood my problems, and corrected them all: the cursor can now be slowed to almost nothing, so finally you can do fine pixel work. He added a spray can function, so no more need for JOYPAINT, and he also added what nobody else had dared doing before him: a function to draw ARCS!

These improvements alone could justify switching to the new version, but the author did not stop there: the FONT and PRINT modules have been totally revamped, and two new modules have been added: VECTORS and MOVIE. The only function that was sacrificed was the TEXT mode in the drawing module (remember, you could scale the letters with the CTRL or FCTN digit keys? You don't remember? I never used it and therefore will not miss it...)

The FONT function was taken out of the ENHANCEMENT module (and the INSTANCE function now have its own ENHANCEMENT menu icon). The new module enables you to write up to 14 lines at once, each with its own parameters: Outline (Y/N), Shadow (Y/N), Position (Left/Right/Center).

The PRINT module has also been totally redone, and you can now print in landscape or portrait format (i.e. as is, or rotated), you can print up to three pictures side by side, you can print a picture from memory or from disk. And, if you happen to own a color printer, like the NX-1000 Rainbow, it will print your picture in colors! The results are simply fantastic! There is only one drawback: because it does so many things, the PRINT module takes ages to print even one, "as is" picture... Jane says it takes close to one hour to print a colored one... I am told that the author released a new version of TI-ARTIST PLUS! with improved printing speed. But if you use the original PLUS!, it is still faster to CONVERT your picture to GRAPHX format and print from there. Or use a copy of your earlier TI-ARTIST.

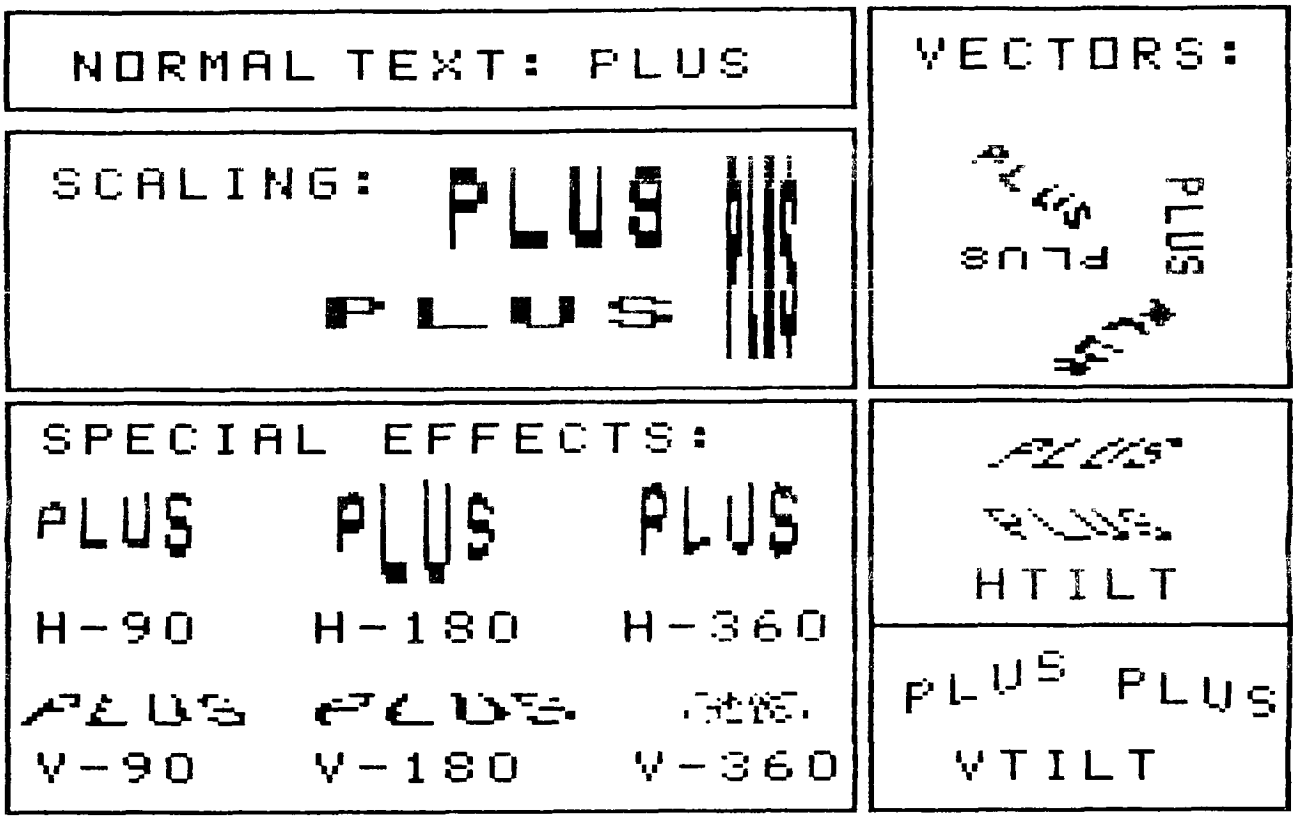
For me, the real PLUS! resides in the added modules: VECTORS and MOVIE. With the MOVIE module, you can create animated pictures almost instantly, provided you have prepared your pictures in advance. Let's say you want to animate my ubiquitous frog: those who have GRAPHX must have realized by now that I use the clipboard CFROG, six pictures of a jumping frog. To give it some realism, I added a pond of water and a diving board. I created the first picture, FROG1, saved it, then modified it slightly for FROG2, etc. When my six pictures were safely saved, I went in the MOVIE module, and with just a few commands, TI-ARTIST contructed a movie file! I found the process much easier to work with than Thomas Opheys' COMIC SHOW. But there is a drawback here too: unlike COMIC SHOW, when you play a movie, the program will play it only once; to see it again, you must wait for the program to re-load the movie...

The second completely new module is called VECTORS, but it is more than that: Vector is only one function in the menu. You can also scale a picture (make it bigger or smaller), tilt it (vertically or horizontally), or play with it in all kinds of ways. The vector function proper will allow you to save a portion of a picture (warning: it disappears from the screen), then reload it with different rotation degrees. The manual says that will even work with three-dimensional pictures, although TI-ARTIST PLUS! cannot create such pictures...

I used the Vector function to write the MERRY CHRISTMAS letters in the cover picture (an up-scaled message written in a TI-ARTIST font, each letter saved as a vector file: if the letter is repeated, you don't need a second vector file), and I used SCALING to enlarge the wreath so that Froggy's head could fit in its center. Due to the vector algorithm, the rotated (to any angle) pictures do

not come out as "clean" as the source one: some reworking has to be done in the ARTIST drawing module, but it is still much faster than if I had to draw each tilted letter! The chart below shows the "raw" results of each function in the VECTOR module.

As usual, Chris Faherty has done a superb programming and designing job: the icon menus are aesthetically very pleasing. TI-ARTIST PLUS! is published by Inscebot Inc. and distributed by Texaments; the price is $24.95 US. Of course, the Canadian dealers mentioned on our back cover carry it, at about $30 CDN. If you own the previous version of TI-ARTIST, you can order the new version by sending the original disk, plus the first page of your current manual, with $14.95 U.S. to: TEXAMENTS, 53 Center Street, Patchogue, NY 11772.




```
**********************************************************************************
*                                                                                *
*         W   Ottawa TI-99/4A Users' Group members to serve in                   *
*         A   various positions -- choice assignments are still                  *
*         N   available. Act Now! This offer will not end until                  *
*         T   you do.  Call anyone on the executive, or the                      *
*         E   former incumbent, who will be happy to tell you                    *
*         D   about the job.                                                     *
*                                                                                *
*         Current positions open:  Newsletter editor                             *
*                                  Advertising Representative                    *
*                                                                                *
*         Please note that preference will be given to applicants                *
*         who are currently members of the human race, but frankly               *
*         we're not too fussy on that point.  The Ottawa TI-99/4A                *
*         Users' Group is an equal opportunity employer.                         *
*                                                                                *
**********************************************************************************
```

ASSEMBLY UTILITY PROGRAMS FOR EXTENDED BASIC PART 2

By David Caron

[Editor's note: due to the length of David's article, the Assembly routine will be published in two parts, this month and next.]

This month's utility is a string dump routine which takes in two, 224 char.-long strings, and sends them to a printer in the form of a high resolution bit stream. The utility is accessed from Extended Basic by:

```
CALL LINK("PRINT",FILENAME$,B$,C$,PC,D,EC)
```

where FILENAME$ is the output device such as "PIO", B$ and C$ are 224 length strings to be printed, PC is the escape code for your printer, and D indicates IBM line spacing or Epson.

B$ and C$ are not normal Extended Basic strings. They must be created using the utility CALL LINK("RSCRN",A$,B$,C$). The documentation and source code for this utility can be found in the October newsletter. Basically, the RSCRN routine will store the top eight lines of the screen in A$, the second eight lines in B$, and the last eight lines in C$. Columns 1,2, 31, and 32 are not stored in the strings.

When two of the three strings are sent to the print utility, they will be printed out exactly as they appeared on the screen, INCLUDING THE CALL CHAR DEFINITION OF EACH CHARACTER PRINTED IN THE PRINTOUT. However, since the printer width has twice the density of the screen, the second string will be printed side by side with the first one, for eight printer lines in all. But you can also use the routine to get a normal screen dump, by defining the right half as empty spaces (see the XB program below).

The high resulution initialization code on most printers is of the form: ESC&CHR$(PC), where PC is an ASCII code. Since this ASCII code can vary from printer to printer, I have made it a parameter accessible from Extended Basic, so that it can be easily changed to suit your printer. The ASCII value for PC (printer code) can be found in your printer manual.

D is an ASCII value which has only two values, D=ASC("I") or D=ASC("E"), where I is used for IBM compatible printers and E is used for Epson compatible printers. The variable D is used by the utility to select IBM spacing or Epson spacing.

EC is the error code returned by the ROM I/O utility routines. If a valid FILENAME$ was used, then EC will equal 0, indicating no error. The other return values of EC are as follows:

1= Device write protected
2= Wrong open attribute
3= Illegal operation
4= Device memory full
5= End of file error
6= Device error OR function 4 has been pressed if FILENAME$="PIO"
7= Nonexistant file
8= Nonexistant device

A more descriptive list of these error messages can be found on page 299 of the Editor Assembler manual.

And that's all there is to it!

The following Extended Basic program is an example of how a direct (normal) screen dump would be performed:

```
10 BLANK$=RPT$(CHR$(192),224) ! 192 is the EXTENDED BASIC blank
   space character as seen from an assembly program. This is why
   normal strings cannot be used with the print utility.
20 PC=ASC("K") ! K is the high res code for a Star NX-10 printer.
30 D=ASC("I")  ! IBM spacing when my Star NX-10 is in IBM mode.
40 CALL LINK("RSCRN",A$,B$,C$)
```

```
50 CALL LINK("PRINT","PIO",A$,BLANK$,PC,D,EC)
60 CALL LINK("PRINT","PIO",B$,BLANK$,PC,D,EC)
70 CALL LINK("PRINT","PIO",C$,BLANK$,PC,D,EC)
```

Since the source code is very long, I have not bothered to insert extensive comments. Since this utility uses a DSRLNK call, the Extended Basic program which will use the routine MUST be executed directly from the Funnelweb program "LOAD". Use one of the user modifiable menu selections to run the Extended Basic program. The version of Funnelweb used must be compatible to version 3.3D.

If this seems too complicated, then type in the DSRLNK into your program with the CALL LOADs below. Place them BEFORE the CALL LOAD() for the print utility. This DSRLNK is copied from version 3.3D of FunnelWriter.

```
1 CALL INIT
3 CALL LOAD(8194,37,238,63,248)
4 CALL LOAD(9460,37,144,36,252,32,56,37,176,200,62,37,102,83,204,2,224,
  131,224,192,32,131,86)
5 CALL LOAD(9482,4,32,32,40,209,193,9,135,7,6,2,2,131,74,5,128,5,134,129,
  198,19,6)
6 CALL LOAD(8194,43,2,63,232)
7 CALL LOAD(9460,0,0,0,0,0,0,0,0,0,0,0,0,203,20,203,53,203,78,203,231,
  204,71)
8 CALL LOAD(9482,204,150,204,228,205,29,205,75,205,96,33,131,35,253,38,
  184,40,183,41,182,42,195)
9 CALL LOAD(9504,0,0,0,0,0,0,0,0,0,0,0,0,0,100,32,0,46,170,37,12,37,54)
10 CALL LOAD(9526,193,126,83,224,37,46,192,32,131,86,194,64,2,41,255,248,
   4,32,32,40,208,193)
11 CALL LOAD(9548,9,131,7,4,2,2,37,2,5,128,5,132,128,196,19,6,4,32,32,40,
   220,129)
12 CALL LOAD(9570,152,1,37,48,22,246,193,4,19,82,2,132,0,7,21,79,4,224,
   131,208,200,4)
13 CALL LOAD(9592,131,84,200,4,36,252,5,132,168,4,131,86,200,32,131,86,
   36,254,2,224,131,224)
14 CALL LOAD(9614,4,193,2,12,15,0,195,12,19,1,30,0,2,44,1,0,4,224,131,
   208,2,140)
15 CALL LOAD(9636,32,0,19,50,200,12,131,208,29,0,2,2,64,0,152,18,37,49,
   22,238,160,160)
16 CALL LOAD(9658,37,22,16,3,192,160,131,210,29,0,192,146,19,230,200,2,
   131,210,5,194,194,114)
17 CALL LOAD(9680,209,96,131,85,19,9,156,133,22,242,9,133,2,6,37,2,156,
   182,22,237,6,5)
18 CALL LOAD(9702,22,252,5,129,200,1,37,0,200,9,36,250,200,12,36,248,
   6,153,16,226,30,0)
19 CALL LOAD(9724,2,224,37,12,192,9,4,32,32,40,9,209,22,4,3,128,
   2,224,37,12,4,193)
20 CALL LOAD(9746,6,193,215,65,243,224,37,46,3,128,32,56,38,32,2,0,131,
   128,196,48,5,224)
21 CALL LOAD(9768,131,114,2,2,32,0,192,210,2,1,38,66,196,129,200,62,131,
   236,2,224,131,224)
22 CALL LOAD(9790,4,96,0,96,2,224,32,56,216,48,156,2,196,131,216,16,156,
   2,3,128,32,32)
```

This is the utility program itself:

```
        DEF  PRINT
VDPWA   EQU  >8C02
VDPWD   EQU  >8C00
VMBW    EQU  >2024
NUMASG  EQU  >2008
STRASG  EQU  >2010
S       EQU  >2014
V       EQU  >2020
V       EQU  >2028
VMBR    EQU  >202C
NUMREF  EQU  >200C
DSRLNK  EQU  >2532 for Funnelweb < 4/1A)
XMLLNK  EQU  >2018
NEWREC  BSS  32
```

```
STRBUF BSS 225
VDPBUF BSS 32
PAB    DATA >0012,PABBUF,>FF00,>0000,>001C
       TEXT 'PIO.LF.CR                  '
PABBUF EQU  >3EEF
R11BUF DATA >0000
R11BU2 DATA >0000
DEFPAB DATA >0909
STR224 BYTE 224
HPBIT  BYTE 27,75,216,1
IBMLS  BYTE 27,65,8,27,50
EPSON  BYTE 27,65,8,0,0

*****************************************************************
* HEXDEC will take the VDP character definition and create the
* corresponding high resolution code representation for that character.
*

HEXDEC SLA  R0,3          *R0<--CHARACTER NUMBER
       LI   R1,VDPBUF
       LI   R2,8
       BLWP @VMBR         *GET CHARACTER DEFINITION

       LI   R8,VDPBUF
       CLR  R0
       CLR  R1
       CLR  R2
       CLR  R3
       CLR  R4
       CLR  R5
       CLR  R6
       CLR  R7
       LI   R10,128

REPL   CLR  R9
       MOVB *R8+,R9
       SWPB R9
       SRL  R9,4
       CI   R9,8          * SET FIRST COLUMN(R0)
       JLT  SKIP1
       A    R10,R0
       ANDI R9,>0007
SKIP1  CI   R9,4          * SET SECOND COLUMN(R1)
       JLT  SKIP2
       A    R10,R1
       ANDI R9,>0003
SKIP2  CI   R9,2          * SET THIRD COLUMN(R2)
       JLT  SKIP3
       A    R10,R2
       ANDI R9,>0001
SKIP3  CI   R9,1          * SET FOURTH COLUMN(R3)
       JLT  SKIP4
       A    R10,R3

SKIP4  SRL  R10,1
       JNE  REPL

       LI   R8,VDPBUF
       LI   R10,128

REPR   CLR  R9
       MOVB *R8+,R9
       SWPB R9
       ANDI R9,>000F
       CI   R9,8          * SET FIFTH COLUMN(R4)
       JLT  SKIP5
       A    R10,R4
       ANDI R9,>0007
SKIP5  CI   R9,4          * SET SIXTH COLUMN(R5)

       JLT  SKIP6
       A    R10,R5
       ANDI R9,>0003
```

```
SKIP6   CI    R9,2            * SET SEVENTH COLUMN(R6)
        JLT   SKIP7
        A     R10,R6
        ANDI  R9,>0001
SKIP7   CI    R9,1            * SET EIGHTH COLUMN(R7)
        JLT   SKIP8
        A     R10,R7

SKIP8   SRL   R10,1
        JNE   REPR

        MOV   R13,R8
        ANDI  R8,>7FFF
        ORI   R8,>4000
        SWPB  R8              * TRANSFER DECIMAL CHARACTERS TO VDP MEMORY
        MOVB  R8,@VDPWA
        SWPB  R8
        MOVB  R8,@VDPWA
        SWPB  R0
        MOVB  R0,@VDPWD
        SWPB  R1
        MOVB  R1,@VDPWD
        SWPB  R2
        MOVB  R2,@VDPWD
        SWPB  R3
        MOVB  R3,@VDPWD
        SWPB  R4
        MOVB  R4,@VDPWD
        SWPB  R5
        MOVB  R5,@VDPWD
        SWPB  R6
        MOVB  R6,@VDPWD
        SWPB  R7
        MOVB  R7,@VDPWD

        AI    R13,8
        RT
```

[The second part will be published next month.]

HOT BUG

While perusing other newsletters in search of inspiration to fill this third of a page, I found the following item in the Hunter Valley 99ers (Newcastle, NSW, Australia!) Newsletter Aug. 1989, itself quoting Gary Taylor in the May issue of the Pittsburgh UG Newsletter... (or how information always comes back, like a boomerang):

"[Charles Earl] has had to develop some debugging tools of his own during the development of PRESS and decided to release "Hot Bug" as fairware. It is a new "pop-up" debugger offering step or realtime execution of programs. It comes complete with a Hex oriented calculator, and will support remote debugging from another TI! It will load into a Supercart or Gram Kracker, leaving a full 32K for your program.

"It has a Fairware price tag of $20 and can be purchased from:

Charles Earl
34 McLeod Street
Ottawa, Ont.
K2P 0Z5 "

FAST EXTENDED BASIC

LUCIE DORAIS

Just when I was despairing, unable to invent new, if not exciting, programs, light came from Regena in MICROpendium: she liked a program (written for another computer) so much, that she translated it for the TI. With that exemple from above, I decided to dive into my pile of photocopied programs, collected since 1983.

So here is the dreaded BUG, written by 7th-grader Brian Leibowitz many years ago in a very simple Basic: no arrays, one statement per line, and most of all no CRT terminal: you are warned that "If you elect to see all the pictures, this program has the ability of consuming well over six feet of terminal paper per game. We can only suggest recycling the paper by using the other side." (David H. Ahl, Basic Computer Games, Microcomputer edition, p. 30). Now we have monitors, and Basic has advanced enough to let us write a shorter and much more attractive program.

The game, basically a random dice game, should appeal to young children. But the program itself, translated into XB, is not for children. Be warned: we enter the glorious world of DEFinitions, arrays, and we perfect our knowledge of "relational expressions".

The object of the game is for each player to build a BUG: each rolls a dice; to each dice number corresponds a body part: 1=body, 2=neck, 3=head, 4=tail, 5=feelers (two) and 6=legs (six). But you cannot get a body part if you don't already have the part it attaches to: no head without a neck, no feeler without a head, and of course nothing if you don't have a body yet.

```
100 ! ** BUG ** L.Dorais / Ottawa UG / Nov. 1989
110 OPTION BASE 1 :: DIM AL$(2),FL(2),F$(2),GE$(2),LG(2),L$(2),
    NO$(2),P$(2,6),PR(6),PT(4,2),RL$(2,2),TP(2),W(2),W$(2)
120 CALL CLEAR :: CALL SCREEN(11) :: A$="      " :: B$=A$&"1"
    :: E=11 :: F=15 :: GOSUB 700
130 DEF SR(X)=X-12*(P=2) :: DEF CN(X)=X-8*(P=2)
140 DATA you`already`have,you`get`,you`don't`have,two`feelers,
    six`legs,PRESS ANY KEY TO,@ROLL YOUR DICE@,YOU@WIN
150 GOTO 170 :: A,C,D,K,P,PL,R,S,T,X :: CALL CHAR :: CALL HCHAR
    :: CALL CHARPAT :: CALL COLOR
160 CALL SPRITE :: CALL MAGNIFY :: CALL KEY :: !@P-
170 DATA a`body,a`neck,a`head,a`tail,a`feeler,a`leg
180 DATA ALREADY HAVE,GET@,DON'T HAVE,TWO FEELERS,SIX LEGS,
    press`any`key`to,`roll`your`dice`,win
190 DATA A BODY,A NECK,A HEAD,A TAIL,A FEELER,A LEG,1,1,2,1,3,1
200 FOR X=1 TO 2 :: READ AL$(X),GE$(X),NO$(X),F$(X),L$(X),
    RL$(X,1),RL$(X,2),W$(X)
210 FOR T=1 TO 6 :: READ P$(X,T) :: NEXT T :: NEXT X :: FOR X=1
    TO 6 :: READ PR(X) :: NEXT X
220 FOR X=9 TO 13 :: CALL COLOR(X,2,11) :: NEXT X
230 CALL CHARPAT(39,A$,63,B$) :: CALL CHAR(64,"",123,A$,124,B$)
240 FOR X=64 TO 90 :: CALL CHARPAT(X,A$) :: CALL CHAR(X+32,A$)
    :: NEXT X
250 A$="FFFFFFFFFFFFFFFF"&"0000000F0F0C0C0C"&"000000FFFF000000"&
    "000000F0F0303030" :: B$="3C3C3C3C3C3C3C3C"&"3C3C181818181818"
260 CALL CHAR(128,A$,132,B$,136,A$,140,B$)
270 CALL D(23,1,RPT$("-",28)&" HOW MANY PLAYERS? (1-2) 1") ::
    ACCEPT AT(24,26)SIZE(-1)VALIDATE("12")BEEP:A$ :: PL=VAL(A$)
280 IF PL=1 THEN A$="I@" :: B$="i`" ELSE A$="YOU@" :: B$="you`"
290 AL$(2)=A$&AL$(2) :: GE$(2)=A$&GE$(2) :: NO$(2)=A$&NO$(2) ::
    W$(2)=B$&W$(2)
300 ! ** game **
310 P=1 :: CALL CLEAR :: CALL SCREEN(14) :: CALL HCHAR(1,1,96,384)
```

```
    :: IF PL=1 THEN CALL D(24,F,"computer")
320 CALL MAGNIFY(2) :: CALL SPRITE(#1,32,2,17,97,#2,32,2,113,97)
330 IF PL=1 AND P=2 THEN 350 ELSE CALL D(SR(10),E,RL$(P,1)) ::
    CALL D(SR(11),E,RL$(P,2))
340 CALL KEY(0,K,S) :: IF S=0 THEN 340
350 CALL ER(P) :: RANDOMIZE :: D=INT(6*RND)+1
360 FOR X=1 TO 12 :: CALL PATTERN(#P,X+48+6*(X>6)) :: NEXT X ::
    CALL PATTERN(#P,D+48)
370 B$=P$(P,D) :: CALL D(SR(3),F,B$) :: T=PR(D) :: IF D>1 AND
    PT(T,P)=0 THEN A$=NO$(P) :: B$=P$(P,T) :: GOTO 590
380 A$=AL$(P) :: IF D>4 THEN 400
390 IF PT(D,P)THEN 590 ELSE PT(D,P)=1 :: GOSUB 690
400 ON D GOTO 420,450,470,500,530,560
410 ! ** body **
420 R=SR(6) :: C=CN(128)
430 CALL HCHAR(R,5,C,3) :: CALL HCHAR(R+1,5,C,3) ::
    CALL HCHAR(R+2,5,C,3) :: GOTO 600
440 ! ** neck **
450 CALL HCHAR(SR(5),6,CN(132)) :: GOTO 600
460 ! ** head **
470 R=SR(3) :: C=CN(128)
480 CALL HCHAR(R,5,C,3) :: CALL HCHAR(R+1,5,C,3) :: GOTO 600
490 ! ** tail **
500 TP(P)=TP(P)+1 :: R=SR(9) :: C=CN(132)
510 CALL HCHAR(R,6,C) :: CALL HCHAR(R+1,6,C+1) :: GOTO 600
520 ! ** feelers **
530 IF FL(P)=2 THEN B$=F$(P) :: GOTO 590
540 GOSUB 690 :: CALL HCHAR(SR(2),5+2*FL(P),CN(133)) ::
    FL(P)=FL(P)+1 :: TP(P)=TP(P)+1 :: GOTO 600
550 ! ** legs **
560 IF LG(P)=6 THEN B$=L$(P) :: GOTO 590
570 GOSUB 690 :: T=(LG(P)>2) :: R=SR(LG(P)+6+3*T) :: X=3-5*T
    :: C=CN(129-T)
580 LG(P)=LG(P)+1 :: TP(P)=TP(P)+1 :: CALL HCHAR(R,X,C) ::
    CALL HCHAR(R,X+1,C+1) :: GOTO 600
590 CALL SND(500,500,250) :: CALL D(SR(6),E,A$) ::
    CALL D(SR(8),E,B$)
600 IF TP(P)<9 THEN P=2+1*(P=2) :: GOTO 330
610 ! ** end **
620 W(P)=W(P)+1 :: A$=STR$(W(1)) :: B$=STR$(W(2))
630 CALL ER(1) :: CALL ER(2) :: CALL SND(1500,1500,1500) ::
    CALL SND(1500,1500,2200)
640 IF PL=1 THEN A$="@@YOU@"&A$ :: B$="@@ME@"&B$ ELSE
    A$="@@PL1@"&A$ :: B$="@@PL2@"&B$
650 CALL D(SR(6),F,W$(P)) :: CALL D(E,E,A$&B$&"@@") ::
    CALL D(14,E,"another`game|`y")
660 ACCEPT AT(14,25)SIZE(-1)VALIDATE("YNyn"):A$ :: IF A$="N"
    OR A$="n" THEN END
670 FOR X=1 TO 2 :: PT(1,X),PT(2,X),PT(3,X),PT(4,X),FL(X),
    LG(X),TP(X)=0 :: NEXT X :: GOTO 310
680 ! ** gosubs **
690 CALL SND(2000,1400,2200) :: CALL D(SR(6),E,GE$(P)&B$) :: RETURN
700 DISPLAY AT(1,3):"THE WINNER IS THE FIRST":"  PLAYER TO COMPLETE
    HIS": :TAB(13);"BUG"
710 DISPLAY AT(6,3):"EACH PLAYER ROLLS A DICE": :"  EACH DICE NUMBER
    STANDS":"  FOR A PART OF THE BUG:"
720 DISPLAY AT(11,5):"DICE  PART    NEEDED":"    ----  -------  ------"
730 DISPLAY AT(13,7):"1   BODY"&B$:A$&"2   NECK"&B$:A$&"3   HEAD"&B$:
    A$&"4   TAIL"&B$:A$&"5   FEELERS   2":A$&"6   LEGS"&A$&"6"
740 DISPLAY AT(20,1):"THE PLAYER CAN GET THE PART ONLY IF HE ALREADY
    HAS THE  PART IT ATTACHES TO..." :: RETURN
750 !@1+ ** u-d subs **
760 SUB D(R,C,A$) :: DISPLAY AT(R,C):A$;:: SUBEND
770 SUB ER(P) :: CALL PATTERN(#P,32) :: T=(P=2) :: C=96+64*T ::
    R=2-12*T
780 FOR X=R TO R+9 :: CALL HCHAR(X,13,C,18) :: NEXT X :: SUBEND
790 SUB SND(A,B,C) :: CALL SOUND(100,A,2) :: CALL SOUND(100,B,2)
    :: CALL SOUND(150,C,2) :: SUBEND
```

BUG gives us the opportunity to use a two-color screen, something I wanted to

do for a long time: player one plays on a yellow background, player two (or the computer) on the magenta screen. If you don't like this color combination, just change the colors in line 220 (top screen) and 310 (screen color, used for bottom screen.)

The trick to work with two screen colors is to use two alphabets; the drawback is that each text data has to be duplicated, but fortunately BUG does not use too much text. Some is written in uppercase, some in lower: this is because, in line 240, we will copy the uppercase alphabet into the lowercase one; char. 64 (@) and 96 (`) will be our "spaces". With two colors, the text displayed in the upper half of the screen will be black on yellow, and the lower half black on magenta; when we need to really attract a player's attention, for ex. when Tex asks to roll the dice, we use the contrasting color: look at DATA lines 140 (placed before the pre-scan) and 170-190 to see what I mean. All that data is read into arrays, one for each player (look at the DIMensioning in line 110), in lines 200-210.

Most of the arrays in BUG have only one dimension, and most will hold only two values, one for each player. PR, DIMed to 6, will hold the prerequested body parts; actually, we fill only PR(2 to 6), since the body does not need a part to attach itself to. Three arrays have two dimensions: P$(2,6) are for the six text body parts for each player; PT(4,2) will hold the flags for the four body parts that are unique (body, neck, head and tail: the two feelers and the six legs counters will be FL(2) and LG(2) respectively). The RL$(2,2) array will hold the two parts of the "roll your dice" message. The data is read by Tex while you read the instructions: in line 120, we have GOSUBed to them, lines 700-740. The variables defined right before are some spaces used by the instruction display; E and F will hold two often used numbers, 11 and 15, mostly used for the display columns.

The characters used to draw the bug are defined in line 250: here is an easy way to define characters, using the 64-char. possibility of XB, but splitting it into four shorter definitions for better legibility. There are only six characters defined, so our BUGS will be a bit square, but Tex does not have enough characters left once we use both upper and lower cases. The top screen bug, black on yellow, will be in set 13, and the bug left "au naturel" (black on transparent) will be in set 14, 8 characters higher. The two screens each have 12 lines, so the messages and bug parts for player 2 will always be displayed 12 rows below the corresponding row for player 1. Which brings us to line 130 (Jane, do you follow me?)

This line has two DEFinitions, a very useful function: in the first one, which deals with Screen Rows, it means that each time Tex will encounter an SR() expression, it will not read it as an array element, but as a pointer to an already defined expression. Same thing for CN(), which deals with the Characters' ASCII Numbers. As you can see, we can use a relational expression, (P=2), in a DEF statement. Here, P is the current player variable, 1 or 2. If it is player 1's turn, the expression (P=2) has a value of 0, so the value of X in the DEFinitions will remain X; otherwise, if P=2, the expression between ()s takes the value of -1: the row will be 12 rows down, and the drawing character will be 8 characters more than the value of X sent by the program statement. Look at line 420, which calcultates the row and character needed to draw the body: R=SR(6) sends Tex to the DEFinition in line 120, with X having a value of 6 (starting row to draw player one's bug body); if it is player two's turn, the row will be equal to 6-12*(-1), or 6+12, or 18. Still in line 420, the character used to draw the body is either 128, for the top screen, or 128-8*(-1), or 128+8=136, to draw on the bottom screen. As you can guess, these functions will be used over and over in the program, which is why it was much easier to DEFine them once and for all.

Back to running the program: by now you have read the instructions, so Tex asks how many players (line 270); if you answer one, you will play against Tex, so in line 280-290 we prefix all player two's messages with a very self-centered "I", in upper or lower case, depending on the message we want to modify. If there are two human players, the messages are prefixed by a more polite "You". Whatever the case, the player using the top (yellow) part of the screen always starts first, but this does not give him more chances: each player cannot get very far until he rolls a one, to get a body. (for simplicity, I will use the masculine for the player, but that does not mean the game is for machos...)

So the first thing we do when we start the game proper, in line 310, is to set P=1, then build the screen: we use the character "", defined as empty earlier. Line 320 is used to put our two dice on the screen; we make them transparent by using character 32, space. Since we do not have enough characters left to draw real dice, we will use the characters 49 to 54, i.e. the digits 1 to 6, with a magnification factor of 2: this will show them big! And because sprite color is totally independent from the character color, they will show as black over any background color (this is the "2" found after "32" in the two sprite CALLs; change it to any color you wish).

Line 330 will display the "roll dice" message for player one, and for player two if he is not the computer; the display row uses the DEFined SR() function; you can see that a function can be treated the same way as any variable. This is the only time in BUG that the player can interface with the game; otherwise, Tex does everything for you: it rolls the dice in line 350 by selecting a random number between one and six; line 360 displays the digit sprites from one to six twice, then the sprite takes the pattern of the D random value; there is a relational expression here: if counter X is higher than 6, we subtract 6 from X, to display the digits one to six a second time.

The reasoning portion of the game starts right after; the first version I wrote had the ON D GOTO right here, and each body part was dealt with separately. Then I grouped together all the duplicate functions, which saved quite a lot of code. As soon as the dice has stopped rolling, B$ is given the name of the body part correspondng to the dice value, kept in the array P$(Player,Dice), and displays it besides the dice, at row 3 or 15. Then the variable T takes the value of the prerequested body part, to check if the player has it or not, i.e. if the flag PT(body part,Player) is set to one; if not, A$ takes the value of NO$, the "don't have" message and B$ takes the value of the missing body part; Tex is then sent to line 590 to display the message and sound some sad tones.

Even if you have the prerequested part, most often you will already have the part corresponding to the dice value just played, so A$ becomes the "already have" message; if the dice value is higher than four, i.e. you rolled a feeler of a leg, you are sent automatically to the ON D GOTO line because these two parts, not being unique, need a personalized checking. If the body part rolled is unique, line 390 checks its flag; if it is on (i.e. not zero), Tex goes to line 590 to display the "already have" message. Third case, you don't have the part yet, so you Get it: the body part flag is set to one, and we GOSUB 690 to display the "get" message before RETURNing to line 400 to send Tex to the proper drawing module.

The four unique parts modules (body, neck, head and tail) work the same way: Tex calculates the display row and the character depending on the player, and draws the part by using CALL CHARs. In most cases the row and char. are determined before the CALL, for better legibility, but in the case of the neck, which is just one CALL, I used the DEFs as variables in the CALL itself. Before drawing the tail, we encounter a new array, TP(Player); it holds the total "ending parts", to warn Tex that a bug is complete when it has one tail, two feelers and six legs; this variable is checked each time the player's turn has ended, in line 600. If the total parts is less than nine, P gets the value of 2 if it was 1, or vice-versa, before going back to line 330 for the next player.

Since a normal bug needs two feelers and six legs, the feeler and leg modules first check the flags FL(Player) and LG(Player); if they are equal to the maximum number, B$ takes the value of the F$ and L$ strings, i.e. "two feelers" and "six legs"; A$ remains the "already have" msg, and Tex goes to line 590 display it. If you have less than the maximum, you get the part, one at a time of course. The feelers are drawn by line 540, which also adds one feeler to the flag, and one to the total part flag. The feeler row is DEFed as 2 or 14 by SR(2), and the character as 133 or 141 by CN(133). The column is calculated by "5+2*FL(P)"; if we have no feeler yet, FL(P)=0, and the column will be 5; if FL(P)=1, the column will be 7.

The draw legs module is a bit more complicated, since we need six legs, three on each side of the body. In line 570, T gets the value of a relational expression: if LG(P)>2, that means if we already have at least three legs, T will become -1; if we have less than 3 legs, T will be 0. We then use the value of T to get the row, using the DEF SR(), sending as a value for it the

result of LG(P)+6+3*T; the variable X will be the column, either 3 or 8 depending on the value of T, and finally the left character (legs need two char. side by side) will be either 129 or 130, again depending on T. To summarize:

LG(n) PL1	LG(n) PL2	T= (LG(P)\2)	row PL1	row PL2	col	characters PL1	characters PL2
0-2	0-2	0	6-8	18-20	3	129-130	137-138
3-5	3-5	1	6-8	18-20	8	130-131	138-139

When the game ends, Tex gets to line 620, where the number of games won by each player is kept in the array W(Player); A$ and B$ take those current values. Line 630 erases both halves of the screen and sounds the victory bells. This very sophisticated game then chooses the final display: if you play against the computer, Tex will use "YOU" and "ME" to display the number of games won by each player; but if there are two human players, it will use "PL1" and "PL2"... The message is always on row 11 (variable E), i.e. in the yellow screen; to make it stand out we use the upper case, which will show this message in black on magenta. Remember, all those "@"s and "`"s are there to put spaces on the screen. The "play again" message in line 650 will be displayed on row 14, therefore in the magenta part of the screen, so we use lower case to make the text black on yellow for contrast. If we want to play again, line 670 resets all the counters and flags to zero.

The user-defined subs are now familiar to you: CALL D saves a lot of DISPLAY AT code. SUB ER(P) erases the part of the screen for player P=1 with char. 64 and the screen for player P=2 with char. 32; again, T is used for the truth or not of a relational expression, used by both the char. statement C and the row statement R. I found that we cannot use a DEFined function in a used-defined sub, so R equals a statement, instead of a reference to SR(2). This sub also makes the sprite temporarily absent by giving it the pattern of the space character. The SUB SND simply plays three sounds.

This is all, have fun during the holidays! If the children make too much noise, give them a few BUGS to play with...

IS THIS OUR LAST YEAR???

by Lucie Dorais

While the whole world is heralding a new decade, the TI community is slowly dwindling down, and seems more underground than ever. More and more, every TIer should invest something into the community: if Garbage In means Garbage Out, it follows that Good Stuff In means Good Stuff Out.

In the September issue, Ruth O'Neill had asked us to write to COMPUTER SHOPPER, to ask them not to drop their TI FORUM column, but it has not reappeared; worse, they never published an explanation, even less an apology, while the Timex-Sinclair and the Adam still have a regular column. How can they justify it? If, like me, you feel like mourning when you check the Table of Contents of CS, please write to them (I did), you never know what will happen! The address is: Bob Linstrom, Editor-in-Chief, Computer Shopper, 1 Park Avenue, New York, NY 10016.

Fortunately, MICROpendium is still there, as strong as ever, and of course many fine newsletters, including ours. But nothing comes free in this world, and I would like to make a special appeal to all of those who still have not renewed: we would like to hear from you as soon as possible, so that we can continue to publish our own newsletter.

HOTLINE NUMBERS

The executive has expressed a desire to assist all members should you have some problems or questions, want to do some library swapping or borrow a book. This will be the place to look. Listed here are the members of the executive, committee heads, and others in the group willing to help in their specialized areas. Of course, if you wish to be placed on the list, just give me a call. I know there is a lot of expertise within our Group, so I hope to add to this list. Please respect normal hours unless you specifically know that someone doesn't mind a call at 3a.m., or use the BBS to leave a message at 738-0617, 24 hours a day, 7 days a week.

CHARLES FARLEY.....PRESIDENT.....(613) 231-3651

JOHN CLARKE.....VICE PRESIDENT.....(613) 838-2081

MICHEL GOSSELIN.....SECRETARY.....(819) 684-3983

RALPH KUHN.....TREASURER.....(613) 236-2182

JANE LAFLAMME.....PAST PRESIDENT...(H) (613) 837-1719 or (W) 745-2225

PETER ARPIN.....SYSOP.....(613) 523-0017

BILL SPONCHIA.....WORKSHOPS.....(613) 523-0878

TONY HOPKINS.....(FORMER) ADVERTISING REP.....(613) 746-4463

DAVE MORRISON.....LIBRARY CHAIRMAN.....(613) 737-4889

STEVEN BRIDGETT.....CASSETTE LIBRARIAN.....(613) 521-3631

HENRI MONAT.....ARCHIVES.....(613) 824-0941

LUCIE DORAIS.....MEMBERSHIPS, (INTERIM) EDITOR.....(613) 232-0393

BOB BOONE.....HARDWARE/SOFTWARE.....(705) 476-9391

ART GREEN.....ASSEMBLY HELP.....(613) 837-1955

DICK PICHE.....TECH.....(613) 521-8667

DAVID CARON.....TECH, EXTENDED BASIC, ASSEMBLY HELP..(613) 837-1397

CLUB BBS.....SET MODEM TO 8N1.....(613) 738-0617

1990 RENEWAL $25.00

NEW MEMBERS $25.00

NAME _____

ADDRESS _____

CITY _____ PROVINCE/STATE _____

POSTAL/ZIP CODE _____ TELEPHONE (____)_____

Please make cheque payable to the Ottawa TI-99/4A Users' Group and send it, along with this form, to the address shown on the cover page -- or better still, bring both to a meeting.

P.O. BOX 2144,STATION D,OTTAWA
*** ONTARIO,CANADA K1P 5W3 ***

EDMONTON 99er USER'S GROUP

EDMONTON, ALBERTA