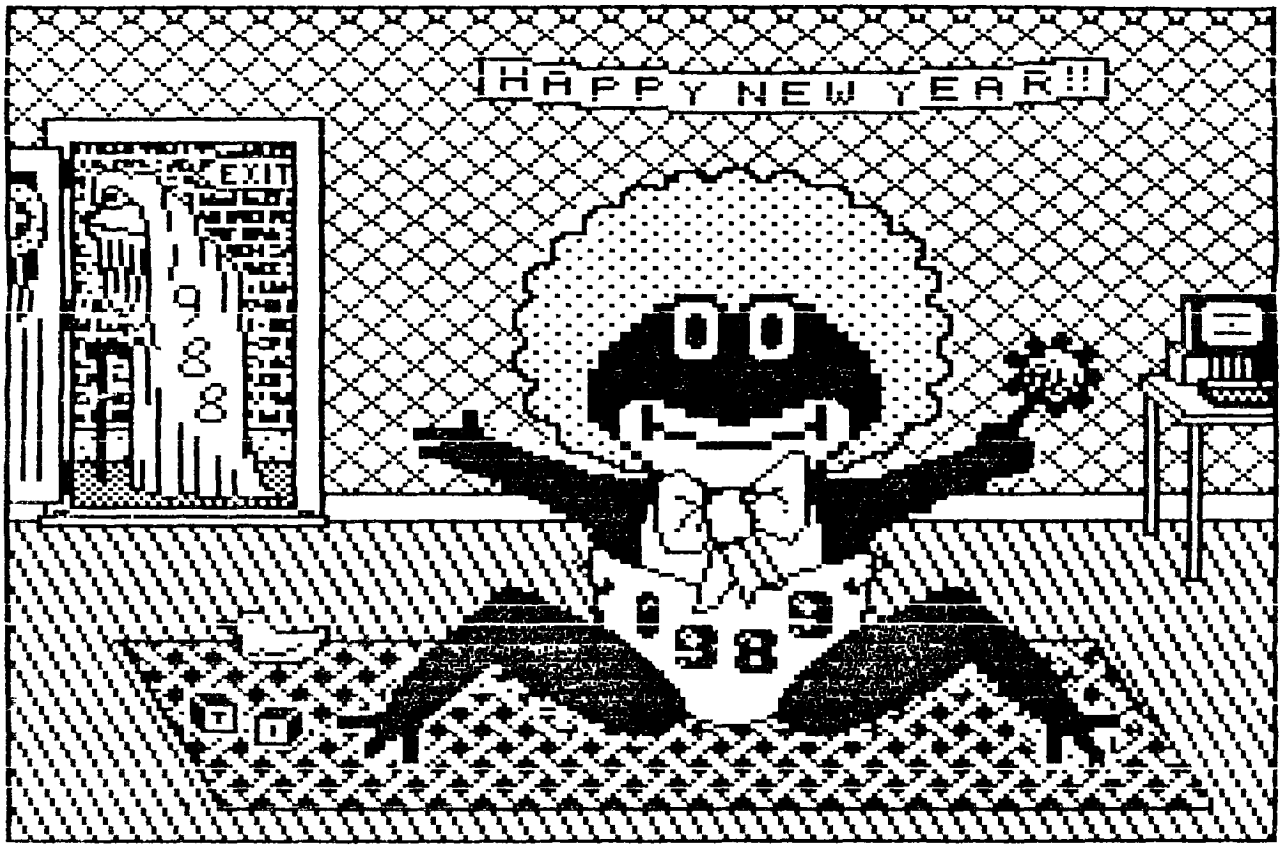
(C-5)
8901

The Ottawa T.I.99/4A Users' Group

VOLUME 8 NUMBER 1.....JANUARY 1989



DON'T FORGET THE MEETING -- January 10, 1989
Special Guest -- Mr. Cheung

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

COMING EVENTS

Beginners' Assembly Language Class	January 7, 1989	Contact Bill Sponchia for more information
January Meeting:	January 10, 1989	Merivale High School Please note that this is the SECOND Tuesday of the month, for this meeting only.
Utilities Workshop: Tom Bentley and Charles Earl will demonstrate their favorite utilities for the /4a.	January 21, 1989 10:00 a.m.	Merivale High School Contact Bill Sponchia for further information.
February Meeting:	February 7, 1989 7:30 p.m.	Merivale High School
4th Annual TI-FEST	April 29, 1989	Merivale High School Contact Jane Laflamme for details or to volunteer your help.
Newsletter Deadline:	January 15, 1989	(or any time before that!)

EDITOR'S NOTES
from
Ruth O'Neill

Well, I hope everyone had a restful and/or enjoyable holiday, and received some new goodies to use with our favorite computer. There was certainly enough to choose from, this year! If you did get some new programs or hardware, how about telling the rest of the club how you like them in a review here? This newsletter is YOURS, and it's up to you to provide material for it if you want to read it. I have some software coming in to review, and one reviewer lined up, but I could really use more. A number of people have been contributing regularly for some time, now, but this newsletter will be much more interesting if it has new and different kinds of articles each month as well. If you have even the SMALLEST item you'd like me to run, TELL ME! If you're not interested in writing, but the editorial side instead, tell me that, too. If you'd like to be the editor, I'll trade and promise to write something for you every month for a year. Then I'd be free to give Jane more help on the Fest. (Incidently, if Jane doesn't start hearing from volunteers soon, I should imagine she'll start feeling very discouraged about the whole idea.) This is your club. You'll enjoy it so much more if you get involved in it more. I can't stress that enough.

I'd like to thank Mr. Donaldson for bringing his programs to our December meeting. They looked very interesting, and I hope that some of the people who bought programs from him will send in reviews soon. (Yes, folks, for those who missed it, that WAS a hint! <grin>) I hope we'll see him again at a future meeting, although I realize that he has a fair way to travel.

Have a happy and prosperous new year, everyone, and here's to another year of TI-ing!

The President's Two Cents' Worth
by Jane Laflamme

The Newsletter has arrived, and I hope the "hang-overs" of the celebrations have departed! (I wonder if Lucie's Ottawa frog had a hangover...) I have a couple of new year's resolutions for you. Here's the first that you maybe have forgotten. "I promise to do at least one small thing for the Ottawa TI group this year." Well, I'm trying.. But just think of it, if everyone did one small thing, ideas would pass, and others could build on them, and who knows where it would lead?

Last year was a very active one for the TI world in general, and quite an exciting one as far as Ottawa was concerned. TELCO hit the world like a storm in February; we had our third annual TI-FEST; we decided to put TEXLINK on the commercial market; there have been a rash of new commercial releases, such as TI-Base, FirstBase, Batch-It, PrEditor; enhancements for TI-Artist, etc.; there are programs dedicated to the GENEVE such as DISkASSEMBLER, Hypercopy, and Picture Transfer; the long awaited Hard and Floppy Disk Controller card was shipped; at the time of writing, we are all waiting for the release of PRESS (or has it been released by now?) ... and the list can go on and on and I probably will think of many more as soon as this article is published! Our little computer just will not die, and thanks mostly to the users. Each program released is more and more innovative, and doing things that our "parent" would never have dreamed of. And here is the second resolution you can make. "I will donate a fair amount to the fairware author whose program I use this year." Without them, where would we be? I cannot urge you enough; support them!

And now on to 1989... What will happen this year? It's anyone's guess, if in fact, we can guess!!!

Don't forget, general meeting for January one week later than usual as the school will be still closed for the holiday on our usual week. Tuesday January 10th, 1989, 7:30 p.m. in the Merivale High School cafeteria. See you there..

A happy and healthy New Year to one and all!

Jane

A MY-Word Tutorial
by Jane Laflamme

Here's a little tutorial on how to get MY-Word v1.21 running, especial with hard disk. I had a few problems, so I hope I can save you time I lost in going down the wrong "paths".

First and foremost, UNPROTECT all files, or at least the MWG, MW5, and CONTROL, or you will have a time of it trying to SAVE these or any options.

To have MY-Word run from hard disk, or any other device such as a HRD:

Load My-Word from drive 1 using MW5 for option 5 of E/A; or, MWG into GPL title screen.

In command mode, F9, enter "SP" for Set Path. It will ask you for current path which will be "DSK1.MWG" or "DSK1.MW5", whichever one you loaded. When prompted for "new" path, type in "WDS1.[DIR.SUB-DIR]." (I am unable to test which is right for the HRD, but would guess "DSK6.MWG".) Note the period at the end and no filename for the second option. It would not accept DSK1 or DSK1. in the "current" but did give me an error message on the bottom of the screen (I presume) - so fast I was unable to read it. I found I had to add the filename. Therefore, I assumed incorrectly, that you needed a filename in the second....

You will note that it saves the pathname to disk and thus re-writes your loader (ONLY the one you used). I suggest that you have a backup before going through this exercise. (Always a good practice anyway...) I got myself into a few troubles trying to get it working and needed to start from scratch!

Here's a little tip, especially convenient when running from a hard disk or a

Horizon RAM disk/s. With the following, you can exit GPL, with shift shift control pressed at the same time, and then go directly into MDOS with control, alt, del pressed at the same time. When you type in "MW" [cr], it will automatically load GPL, then the loader for MY-Word. All you have to do when entering TI mode is press option 2 for MY-Word. I have a little batch file for most of my Cartridges or GRAM loaders. Although the MENU program taking up the last two files of GPL is a great program, I find it somewhat erratic with every update of MDOS. The following method is quite fast now that I am hard disk based.

In MDOS, using "COPY CON MW" [cr], or in MY-Word using Print File - (PF [cr]), stripping control characters with the C in front of the device, (C DSKn.MW [cr]), type in the following:
"E:GPL WDS1.CART.MWG" (or whatever path you require) or "E:GPL MWG" if your Hard Disk or HRD is assigned as "E" drive and GPL and MY-Word are residing there.

To assign your hard disk, in MDOS type "ASSIGN hdS1=E:" for the above example. It can be any letter you wish...

I would love to hear from others, and some of the tips and ideas they have. Write to me at the address on the Newsletter, phone me at the numbers listed at the back, or leave me a message on the Ottawa BBS.

Enjoy....

BROWSING THE LIBRARY
--with DAVE MORRISON

It is a little strange, writing a newsletter article for the January newsletter, even before Christmas. I am supposed to hope that you all have had a very enjoyable holiday whilst yet awaiting mine! Anyway, you all did have a good holiday and I wish you the best for a Happy and Prosperous New Year.

For January, I will be offering a disk by our own Art Green (RAG SOFTWARE). This is the latest version (V4.0) of T.I. Writer and is described by the author as "Fareware Improvements for TI Writer Word Processor and Editor". This programme now happily resides in my RAMdisk. One of these days, I hope to see an article written about Art and his accomplishments. I might point out that it is mainly through the good offices of members such as Art Green and Lucie Dorais that I have been able to provide our members with new programmes each month. Sure, I can always fall back on our Library but, as a I have written previously, most of the programmes in the Library are already in the libraries of our members.

Another Version 4.0 is also available - This is "COMIC SHOW" by Thomas Ophey of Duisberg in the Federal Republic of Germany. This disk was provided by Lucie Dorais, who also edited the disk and added one of her own programmes. Mr. Ophey is not asking for any financial compensation for his efforts, but he would appreciate receiving your opinions, preferably on disk. Please don't disappoint him!

You will note that I have not attempted to describe "COMIC SHOW". It is an unusual graphics programme which really is beyond my capacity to explain! Perhaps Lucie can be persuaded to either demonstrate or explain how this programme functions at our next meeting.

Art Green's disk is single-sided and "COMIC SHOW" will be available in both single and double-sided format.

See you on the 10th - one week later that our regular "first Tuesday of the month" date.

Dave

P.S. Having said that Art Green's T.I. WRITER Improvements is now on my RAMdisk, the Editor will probably mildly comment about the fact that I have obviously not yet learned how to use it!

EXPANDING EXTENDED BASIC'S POWERS,
WRITING ASSEMBLY ROUTINES: PART 4
By David Caron

For some queer reason my last article was labeled PART 2 when it should have been PART 3.

The four Extended Basic routines I am about to discuss are what make Assembly and Extended Basic such a great team. Using these routines, you can transfer up to sixteen variables between Extended Basic and Assembly. Any of these variables could be entire arrays if you wish. Just imagine! You could make up an unlimited number of new Extended Basic procedures, like reading in a whole line from the VDP screen to a string all at once instead of using CALL GCHAR. Today, however, we will use just use these routines to print something on the screen. The four routines are:

NUMREF (NUMber REFerence, BLWP @>200C)
NUMASG (NUMber ASsiGn, BLWP @>2008)
STRREF (STRing REFerence, BLWP @>2014)
STRASG (STRing ASsiGn, BLWP @>2010)

All of these routines are loaded from the Extended Basic module into low memory along with VMBW, VMBR etc. when CALL INIT is executed.

NUMREF is a routine which copies a variable from extended basic into assembly. Let's take an example. Say you executed your assembly routine with: CALL LINK("START",X). X is an Extended Basic variable which I will let be equal to 5. Just before you execute BLWP @NUMREF in assembly, you load a value into R0 and R1. The value in R1 tells NUMREF where in the parameter list the variable is. In our case, the variable is the first parameter in the list, so R1=1. For simple non-array variables like X, always set R0=0.
IF X had been an array like DIM X(10) and you wished to select any of the ten values from Extended Basic, then the call link would look like CALL LINK("START",X()). In such a situation R0 is used to tell NUMREF which array element you want passed. If you wanted to pass X(4) then R0=4. IF you wish to know how to access multidimensional arrays, read section 17.2.1 in the Editor Assembler manual. For our purposes, knowing how to access single dimensional arrays will be sufficient.

Once R0 and R1 have been set, BLWP @NUMREF can be executed. The variable in the parameter list will be copied to CPU addresses >834A to >8352, yes 8 BYTES! not two. I neglected to remind you that all of Extended Basic variables are in FLOATING POINT notation and take 8 bytes to represent them. I will also add that they are always in a form of scientific notation, so it would be difficult to convert a simple number like 5.0000000000000E0 (x=5) to a Word (16-bit) number. Fortunately the TI home computer comes with a bunch of great little routines like CFI (Convert Floating point to Integer). This routine is accessed with:

```
BLWP @XMLLNK    * where XMLLNK=>2018
DATA >12B8
```

This Very Handy routine will take that awful 8 byte floating point number and convert it into the 16-bit number five. It will then place this number at the address >834A. If you do something crazy like making X=1.465838734 then CFI will simply round the floating point number and place a 2 at >834A. CFI even handles negative numbers! What more could one ask? Do not ask me what happens if X=1.0E99 or X=1.0E-99. Remember that -32768<=x<=32767 for a possible conversion to 16-bit format. The only reason I am making such a big deal out of this is because I went to the trouble to make a routine similar to CFI, and then learned of its existence in the console!

NUMASG is identical to NUMREF except that you place the 16-bit number you want sent to Extended Basic, in >834A. Notice however that before you can execute BLWP @NUMASG, the number must be in floating point notation. CFI will not work, but CIF will! (Makers of the TI-99/4A evidently thought of everything) CIF, as you may have guessed, stands for Convert Integer to Floating point. This console routine can be accessed using:

```
BLWP @XMLLNK  *where XMLLNK =(>2018)
DATA >20
```

and presto! there you are. All that is needed now is BLWP @NUMASG. If R1=1 then the Extended Basic variable in the first parameter of CALL LINK will be set to

the original integer at >834A. If the call link statement is something like CALL LINK("START",1) instead of CALL LINK("START",X) then NUMASG will return you to Extended Basic and issue an error message.

STRREF is similar NUMREF except that a string is passed from Extended Basic to Assembly instead of a number. R0 and R1 function the same way. R2 is used however to tell STRREF where you want the string. The usual procedure is to allot some memory before the actual start of your assembly routine. This memory is alloted in the same way as was done for the user workspace registers in the last article. You need only allot LEN(string)+1. The additional character indicates how long the actual string is. This is why strings cannot be any longer that 255 bytes -- that is the largest number possible for a byte to represent.
EX. STRBUF BSS 25
This string can be no longer than 24 bytes. R2 is assigned the address STRBUF. Now only one more thing must be done. If, for example, the string in Extended Basic was 255 characters long instead of 24, guess what would happen? STRREF would simply copy the string starting at STRBUF, then continue copying over your workspace register and much of your assembly routine. Such a situation would likely result in unpredictable results on the part of the TI computer. Fortunately there is a built in safeguard against this. When STRREF is called it will check the byte at the address indicated by R2 and check to make sure that the Extended Basic string is no longer than the value of this byte. If it should be, STRREF will return execution to Extended Basic and issue an error. When BLWP @STRREF is executed, the actual string starts at STRBUF+1, not STRBUF.

STRASG: Well, there is not much to say here. All you do is set R0=0, R1 to the string position in the CALL LINK parameter list, make up the string somewhere in CPU memory, set the byte immediately in front of that string equal to the length of the string, set R2 equal to the address of that byte, execute BLWP @STRASG, and the string variable in extended basic gets assigned the string in CPU memory.

Now I will rewrite the assembly routine from my last article using the above routines. This assembly routine can be accessed from Extended Basic using CALL LINK("START",X,S$) where X is the starting address of the string S$ being written to the screen. 0<= X <=767.

```
        DEF  START     *This places the word "START" in the def table
                       *along with its start address.
VSBW    EQU  >2020     *The Extended basic assembly environment has
VMBW    EQU  >2024     *no ref table so the assembler directive EQU
VSBR    EQU  >2028     *(equate) must be used to define the constants
VMBR    EQU  >2030     *VSBW, VMBW, VSBR, and VMBR. Take a look at
                       *page 415-416 of the Editor/Assembler manual.
NUMREF  EQU  >200C
NUMASG  EQU  >2008
STRREF  EQU  >2014
STRASG  EQU  >2010
XMLLNK  EQU  >2018

STRBUF  BSS  256       *This is where strings will be transferred
                       *between assembly and Extended Basic.
NEWREG  BSS  32        *This is where our workspace will be. 32 bytes
                       *are reserved since 16 registers X 2 = 32
START   LWPI NEWREG    *This instructs the computer to use the memory
                       *locations indicated by NEWREG as the work-
                       *space registers.
        CLR  R0        *loads R0 with 0
        LI   R1,1      *loads R1 with 1 (first parameter)
        BLWP @NUMREF   *gets the value of X from CALL LINK
        BLWP @XMLLNK   *converts X from a floating point number to an
        DATA >12B8     *integer which is in >834A
        MOV  @>834A,R3 *Saves the integer X into R3 since the word
                       *at >834A may be altered by STRREF.
        LI   R1,2      *loads R1 with 2 (second parameter)
        LI   R2,STRBUF *address of where S$ should be copied to.
        LI   R0,256    *Loads R0 with 255 (largest possible string)
```

```
       MOV  R0,*R2    *makes @STRBUF=255 This allows S$ to be up to
                      *255 characters or less.
       CLR  R0        *restores R0 to 0 since S$ is not an array.
       BLWP @STRREF   *copies S$ into STRBUF starting at STRBUF+1
       LI   R2,STRBUF+1 *tells the routine below where to find S$

       MOV  R3,R0     *tells the routine below where to start
                      *writing the message to the VDP screen

*The rest below is identical to the last article in the December
*newsletter

LOOP   MOVB *R2+,R1   *The asterisk in this case does not mean a
                      *comment is starting but rather, now read
*carefully: The BYTE at the address indicated by R2 is COPIED
*into R1, so the value of R1 becomes 83 X 256.(ascii code of S,
*the X 256 means that 83 is in the most significant byte of R1,
*this is how MOVB works ). The + sign after R2 means that the
*value in R2 will be incremented by 1 AFTER the MOVB operation
*has been performed so the value in R2 is now STRING+1.  This
*operation is what is known as: Indirect Auto-Increment
*Addressing. See page 58 of the Editor Assembler manual.

       AI   R1,>6000 *adds hex 60 or decimal 96 to the most
                     *significant byte of R1.
       BLWP @VSBW    *Sends the ascii code in R1 out to the screen
       CI   R2,STRING+4  *compares R2 with the address of STRING+4
                     *and sets the STATUS bits to indicate less
                     *than equal to and greater than conditions
       JLE  LOOP     *If the STATUS bits indicate a less than
                     *(really a low) or equal then goto LOOP.
       CLR  R0       *this code simply returns you back to
       MOVB R0,@>837C *Extended Basic.
       LWPI >837C
       RT
```

With this assembly routine you can control where and what your message is written to the VDP screen. Keep in mind that unlike DISPLAY AT, if you let x be 767 and S$ is 256 bytes long, screen wrap will not occur with S$ being printed at the top of the screen, but rather, S$ will be written to the VDP being used for the first 31 character definitions, ascii 0 to 30. Since the memory for these characters is being used to hold some of Extended Basic's system variables, queer things could occur.

Hmm. I have not yet decided exactly what will be in my next article. Any suggestions? If you have any questions do not hesitate to call me at 745-4618.

NEW LOCATION FOR CDU!

Computer Download Unlimited at last is settled in a new location:

Computer Download Unlimited
8 Talon Street
North Bay, Ontario
P1A 1N5

Phone: (705) 476-9391 (Voice)

(705) 476-3043 (BBS 300/1200)

All the best in your new home!

FAST EXTENDED BASIC

LUCIE DORAIS

Last month (year!), I promised to give you a program to transform the flakes created with SNOWFLAKE into TI-Artist Instances. I did better: a User-Defined Sub that you can merge to any XB program; all you have to do is to add a line (CALL INSTANCE) to call it from wherever you want in your own program. And I give you two examples: the Flake Utility, plus a picture of the Mayflower ship from last May's Newsletter cover. So, people who don't have TI-Artist can still get something to do this month, as the Ship program can be run by itself, to show a pretty picture (feel free to add music to it...)

We start with the Flake utility:

```
100 ! FLAKES>TI-ARTIST INSTANCE / L. Dorais / Ottawa U.G. / Dec. 1988
110 !
120 GOTO 130 :: A$,FN$,IN$,C,K,KN,N,R,X :: CALL CHAR :: CALL HCHAR ::
    CALL COLOR :: CALL SCREEN :: CALL INSTANCE !@P-
130 DISPLAY AT(1,2)ERASE ALL:"FLAKE FILE TO READ:": :" DSK1.": : :
    " INSTANCE TO CREATE:": :" DSK1."
140 ACCEPT AT(3,5)SIZE(-12)BEEP:FN$ :: FN$="DSK"&FN$ :: ACCEPT AT(8,5)
   SIZE(-12)BEEP:IN$ :: IN$="DSK"&IN$
150 IF POS(IN$,"_I",1)=0 THEN IN$=IN$&"_I" :: DISPLAY AT(8,2):IN$
160 DISPLAY AT(13,10):"READING..." :: OPEN #1:FN$,INPUT ,VARIABLE 65
170 INPUT #1:N :: FOR X=1 TO N :: INPUT #1:SD$(X) :: CALL
    CHAR(4*(25+X),SD$(X)) :: NEXT X
180 CLOSE #1 :: DISPLAY AT(13,10):""
190 ! display the flakes
200 K=104 :: KN=K/4-25 :: FOR R=13 TO 16 STEP 3 :: FOR C=10 TO 22 STEP 3
210 CALL HCHAR(R,C,K) :: CALL HCHAR(R+1,C,K+1) :: CALL HCHAR(R,C+1,K+2)
    :: CALL HCHAR(R+1,C+1,K+3):: IF KN=N THEN 240
220 K=K+4 :: NEXT C :: NEXT R
230 ! call the sub
240 CALL INSTANCE(13,10,17,23,IN$,64)
250 ! continue program
260 DISPLAY AT(24,8)BEEP:"ANOTHER ONE? Y" :: ACCEPT AT(24,21)
    VALIDATE("YN")SIZE(-1):A$ :: IF A4="Y" THEN 130 ELSE END
```

The program first asks which Flake file to read from (created with last month's program, SNOWFLAKE), then the filename you want for your Instance; if you forget the "_I" suffix, the program will add it for you. Tex then reads the file and fills characters 104 to 143 with the sprite definitions found in the file.

The quickest way to create an Instance is to read the drawing from the screen, character by character; it will not work with sprites, since they are "in front of the screen". We must thus display our flakes in the form of characters (lines 200-220). We place them in two rows of five, with some space between them. K is the character number (we start with 104), and KN keeps track of the characters put on screen, so that the display ends with the last character in your flake file (which is kept in variable N).

All that is left to do is to call the INSTANCE sub, which we do in line 240. But where is the sub??? Right below, to be typed separately and saved as a MERGEd file. The parameters passed by CALL INSTANCE are the starting row and starting column of the area you want to save as an instance, then the last row and the last column; here, we save an area from 13,10 to 17,23. The next parameter is the Instance filename, saved in this case in the variable IN$. The last parameter is the character number of a sprite to be used in the sub; I include it in the CALL because you should use a character that is not used in your drawing, since the sub will redefine it to an ugly square. Here we use "@", not used by the flakes.

INSTANCES

In TI-Artist Version 2.0 and above (Publ. by Inscebot, Copyright 1985 Chris Faherty), the Instances are small pictures that you can load into your picture

screen; the format is DIS/VAR 80, same as a text file. If you read an Instance file with Ti-Writer, you will find that each line is made up of eight numbers, separated by commas, plus a first line with only two numbers, also separated by a comma. The interpretation is very simple: the two numbers on the first line are the total columns and the total rows covered by the Instance, then all the other lines correspond to one character definition, translated into decimal, byte by byte:

```
CALL CHAR hex definition:  "05182446FCABDE73"
 Byte by byte definition:   >05  >18  >24  >46  >FC  >AB  >DE  >73
      Decimal equivalent:     5   24   36   70  252  171  222  115
 TI-Artist Instance line:  "5,24,36,70,252,171,222,115"
```

Here is the MERGEable program that will do the trick: after having typed NEW (SAVE the Flake Utility first!), type it as you would a normal program, then SAVE it as a MERGE by typing SAVE "DSKn.FILENAME",MERGE; don't forget the MERGE attribute! To use it with a program, first OLD your program, then type MERGE "DSKn.FILENAME", and the lines 32500-32630 will be added at the end of your program. I used very high line numbers, to make sure that the new lines will not erase previous lines in your program.

```
32500 !@P+ User-Def Sub - XB>ART
32510 SUB INSTANCE(STROW,STCOL,ENDROW,ENDCOL,FILEN$,SPRCAR)
32520 P$="0,0,0,0,0,0,0,0" :: CALL CHAR(SPRCAR,"FFFFFFFFFFFFFFFF") ::
      CALL SPRITE(#28,SPRCAR,11,193,1)
32530 OPEN #20:FILEN$,OUTPUT,VARIABLE 80
32540 PRINT #20:STR$(ENDCOL-STCOL+1)&","&STR$(ENDROW-STROW+1)
32550 FOR R=STROW TO ENDROW :: FOR C=STCOL TO ENDCOL :: CALL
      LOCATE(#28,R-1)*8+1,(C-1)*8+1)
32560 CALL GCHAR(R,C,K):: IF K=32 THEN PRINT #20:P$ :: GOTO 32600
32570 CALL CHARPAT(K,C$):: P=1 :: A$=""
32580 CALL RBYTE(C$,P,HB):: CALL RBYTE(C$,P+1,LB):: A$=A$&STR$(HB*16+LB)::
      IF P<15 THEN A$=A$&","
32590 P=P+2 :: IF P<17 THEN 32580 ELSE PRINT #20:A$
32600 NEXT C :: NEXT R :: CLOSE #20 :: CALL DELSPRITE(#28)
32610 SUBEND
32620 SUB RBYTE(C$,P,Y):: Y=ASC(SEG$(C$,P,1)):: IF Y<58 THEN Y=Y-48
      ELSE Y=Y-55
32630 SUBEND
```

There are actually two subs: RBYTE (read byte) is extensively used by INSTANCE. After the sub title and parameters, we define P$ as the Instance line for char. "000000...", that is, a space: this makes the whole routine somewhat faster, since the program does not have to translate the character definition each time it encounters a space on the screen. We then create a small yellow square sprite and hide it below the screen.

After the Instance file is opened, the program prints to it the total number of columns and of rows, separated by a comma. Then the sub starts its job, and our sprite follows the progression. Each character is read into K. If a space (K=32), print the line defined as P$; if not, read its definition into C$. Now Tex has to translate this hexadecimal definition into its decimal equivalents: the values P (position in the hex string) and A$ (the line to be printed to the Instance file) are initialized.

Instead of doing complicated calculations to translate the byte values (two characters) into decimal, we take each character and read its high nibble into HB, its low nibble into LB. This is done in a very simple way by the second sub RBYTE. If the nibble is a digit 0-9, its ASCII value is 48-57 (the SEG$ is a character, not the digit itself!), we subtract 48 from it to get the equivalent decimal value; if the nibble is a letter A-F (hex 10-15), we subtract 55 to get the decimal value.

We then add the string values of the high and low nibbles to our string A$; if the position is less than 15 (a char. def. is 16 char. long), we add a separator comma and increment the position by two. When the whole char. definition has been translated, we print the line to the Instance file and start again for the next character, until all are done and we can close the file.

If you merged the above file to the Flake Utility and ran it, and if everything went well, you now have a pretty picture to add to your collection. To load it

into Ti-Artist, press "2" for ENHANCEMENTS from the main Menu, then press "S" for SLIDES, then "6" to LOAD an INSTANCE. When prompted, type the name of the file you have created, but without the "_I" suffix.

Remember, to CALL INSTANCE from your own programs, you must add a line with the following parameters:

CALL INSTANCE(STROW,STCOL,ENDROW,ENDCOL,FILEN $,SPRCAR)

where STROW and STCOL are the starting row and column of the rectangular area to be read by the sub, ENDROW and ENDCOL the ending row and column; those values must frame a rectangle (or a square), even if your picture does not fill it; to do the whole screen, enter "1,1,24,32" as parameters. FILEN$ is the Instance filename, which can be a variable or the full filename (see the Ship example below), and finally the ASCII value of a character to be used as a sprite by the sub, which must not be used by your drawing. In your program, place the new line right after you have drawn your picture on the screen. You can of course design a program that will create an Instance only if you press a certain key.

SHIP

If you are not into Snowflakes, you can try the routine with the following program, which draws the Mayflower. If you type the program exactly as it is, you will get the picture on the screen, floating on a sea of waves. To transform it into an Instance, MERGE the above sub to the program, and replace line 170 with the following:

170 CALL INSTANCE(7,13,16,20,"DSKn.MAYFLOW_I",35)

Since the area covered is from 7,13 to 16,20, the sea is not included... Here we put the Instance filename as a parameter (replace "n" with the desired disk number), and we use character 35 for the sprite. The program itself has no difficulty, although a lot of data! (Don't type the trailing remarks at the end of the DATA lines, they will not fit into a regular line; they are included here for debugging purposes only.). Please note that the pre-scan, line 190, does include the first DATA statement.

```
100 REM ** MAYFLOWER ** L.Dorais
110 REM
120 CALL CLEAR :: CALL CHAR(47,"00001028C6") ! wave
130 W$=RPT$("/ ",14)&RPT$(" /",14):: DISPLAY AT(16,1):RPT$(W$,4) ! sea
140 FOR X=48 TO 119 :: READ A$ :: CALL CHAR(X,A$):: NEXT X ::
    A$=RPT$(" ",10)
150 DISPLAY AT(7,13):"0123":A$&" 456789 ":A$&" :;<=>? ":A$&" @ABCDEF":
    A$&"GHIJKLMN":A$&"OPQRSTUV":A$&"WXYZ[\]":A$&"^_`abcde":A$&"fghijklm"
160 DISPLAY AT(16,11)SIZE(8):"nopqrstu" :: DISPLAY AT(17,16)SIZE(2):"vw"
170 GOTO 170
180 DATA 000000000000407C,04070404040704FC,00C04F784FC4FD01,00008060800000
    FC,0000000000010107,47447C2F23FEFC24,86FC44FFFF04040F   ! car. 48-54
190 !@P-
200 DATA 010101C1C181FFFF,040704FC0F00FCFC,F886F840C0000000,0C08080F1F3020
    60,247673FFFF277C78,0F15FFFF04040507,01010383E57FFF03   ! car. 55-61
210 DATA 0808888442FFFF80,0000000000F0F020,4040406020301C04,78707070B8A8AC
    AF,060E0C0C0C1616FF,0303030505050707,80888F484848C8C8   ! car. 62-68
220 DATA 407EC10000000000,0000EC3424242424,0000000001010302,0F3F60C0800000
    00,FFFE38306040C080,FF0C0F1E18306040,C1FFFF0303030D0D   ! car. 69-75
230 DATA 08F8F89898A86F68,000000000003EC1A0,2424242424E4AC,0202020203030
    02,0000000010181C3,8080808080C0E031,C080808080C0E0     ! car. 76-82
240 DATA 1D191939395B7377,58D8A8381C143EC1,606020203030 18F8,70000000000000
    00,050505060A0A0A0C,633F1F3567C58785,1FFFE141C143C243   ! car. 83-89
250 DATA E0F059CF4BCACBCF,D4D868B3FE93FC42,1CDDDDFF49FF0008,2C968EEA3EE51F
    21,1517171D2D79597E,8A0F0A0F151F959F                    ! car. 90-95
260 DATA C283828705070507,CCCCCDCD4DCF4AD3,C7C7477FE3003FE0,1C1C1CFFBEA2FF
    41,717171FDE380FE03,80804040C0E020E0,0907070302030203   ! car. 96-102
270 DATA 95EB7E2ADEBAEFBA,058FCA6F3A1F0AFF,5EA4B4B4C7CEFBEE,1BDBC03FEABFEA
    BF,495D5DFFAAFFAAFF,6C6D01FEABFEABFE,20A0A020E0A0E0A0   ! car. 103-109
280 DATA 060404047E02013E,EF3A0F00000815E2,AAFFAAFF0001FF00,BBEFB1E1018141
    3F,F000000000205F8,07060509080808FC                     ! car. 110-115
290 DATA 03808040C2AD7050,E04040407CC0700E,03001028C6000000,F ! c. 116-119
```

BITS FROM OTHER USER GROUPS

The executive is currently seeking a means of making the newsletters we receive from other users' groups more readily available to the membership. Meanwhile, Dave Morrison has spent a considerable amount of time going through many of these, and presents here an article reprinted in its entirety from the BITS, BYTES & PIXELS column of the NUTMEG TI-99ERS May 1988 Newsletter. Dave says "As BITS, BYTES & PIXELS is published by the Lima, Ohio User Group, and Charles Good is that Group's Librarian, it is assumed that is where the item originated!"

JOHN JOHNSON'S BOOT V4 - AN OVERVIEW BY CHARLES GOOD

A version of John Johnson's MENU program called BOOT has been written especially (for) the Genial Computerware Horizon Ramdisk eprom. BOOT is in the public domain, is now in version 4, and has been MUCH IMPROVED since I originally reviewed the Genial Horizon eprom in the November issue of BB&P. The comments about BOOT v4 which follow should be of general interest because you do not need either the Genial eprom or a Horizon Ramdisk to use BOOT. If necessary, BOOT will do its stuff using just a basic SSSD one drive disk system. As a means of displaying a powerup menu of selectable goodies at the push of a button, BOOT is in some ways easier to use and configure than either FUNNELWEB or a GRAM KRACKER.

BOOT will display a menu that allows you to do the following, usually with just one keypress:

1- Show on the screen or print to a printer or disk file a directory of any disk in any drive or any Horizon Ramdisk set at any CRU address. Some prior versions of MENU or BOOT only worked with Horizons at CRU 1000.
2- View or print to your printer any D/V80 text file.
3- Run any program that lists as "PROGRAM" in a disk directory from any disk or ramdisk at any CRU address using any drive number from 1-9. If it is an extended basic. or TI Basic PROGRAM, you need to have the XB module plugged in. If it is an EA#5 PROGRAM, BOOT does NOT need the EA module to run these. To RUN any PROGRAM you can either type in DSKxFILENAME, or mark the PROGRAM on the disk directory and then automatically run it, or select one keypress PROGRAM loading and execution from a menu of up to 15 user configurable choices. Previous versions of BOOT only allowed 6 user configurable menu choices and would only run PROGRAMS from Horizons set at CRU1000.
4- Run the installed cartridge.
5- Run the CorComp disk manager.
6- Cycle through 48 foreground-background color combinations.
7- Switch from GROM to a ROM cartridge if you have a SUPERCART or GRAM KRACKER.
8- Go to TI BASIC by pressing "B" without the need to reset the computer and go through the title screen and powerup menu.
9- Display the time if you have any of the currently known real time clocks (such as the CorComp clocks). This display occurs automatically as soon as BOOT is booted.

Configuring the user selectable options of BOOT used to be cumbersome, requiring either a sector editor, or reassembly of the source code. Configuration of BOOT v4 is VERY EASY. All you do is press BEGIN (FCTN 5), and you get a display of each menu item, as it would be displayed on the screen, and the PROGRAM file name that is booted by this menu item. The cursor is on the screen display of the first configurable menu item. Just type over what is already there (up to 15 characters for the menu display) and press enter. The cursor is now on the disk file name that is booted by the first configurable menu item, so just type over that (DSKx.FILENAME) and press enter.

The cursor is now on the screen display of the second configurable menu item which you can configure in the same way, etc. When finished typing in screen displays and file names press BACK and then any other key, and your configuration is automatically saved back to BOOT in whichever drive or ramdisk BOOT came from. What could be simpler! This procedure is much easier that configuring the user list of FUNNELWEB. You can, of course RUN either the LOAD or UTIL portion of FUNNELWEB from BOOT.

BOOT allows for boot disk tracking. It knows the drive name and number from

which it was loaded. If you specify a wildcard (*) in a configuration file name as DSK*.FILENAME, BOOT will go back to the drive that contains BOOT to load the file. Configured file names can be up to 30 characters, and BOOT will support hard disks with file names such as WDS1.DIR1.DIR2.DIR3.FILENAME.

BOOT is an assembly program file that can be loaded from the E/A module option 5 or from extended basic using the accompanying XB loader program. If you have a Horizon Ramdisk with the Genial Computerware eprom, you can type CALL BOOT from either extended basic, or configure the ramdisk to automatically load BOOT as soon as you turn on the computer. The major advantages of BOOT v4 over previous versions are super easy configuration and the ability to boot software from literally any drive or any Horizon Ramdisk irrespective of ramdisk CRU setting. BOOT is also said to be compatible with the Grand Ram ramdisk.

The major limitation of BOOT is that it will only run PROGRAM files. You can't load long extended basic programs that list in a directory as I/V 254 and you can't load D/F80 assembly code. If you want instant menu access to a list of programs, combined with some disk manager functions, than BOOT deserves serious consideration. It is especially useful with Horizon Ramdisks. If you have floppy drives, you might consider putting a copy of BOOT on each of your commonly used floppies. User groups (not individuals) can obtain boot v4 and its extended basic loader from the Lima User Group. Send a disk and postage paid return mailer to P.O. Box 647, Venedocia OH 45894.

(Note: BOOT By John Johnson is available from our Library. - Dave Morrison)

TI BASIC continued from December
by Steven Shaw

The following is a PART of the source code of a program to DISPLAY AT:

```
DS  MOV  R11,R10
    CLR  R0
    LI   R1,1
    BL   @GN
    BL   @LC
    DATA 1
    DATA 23
    MOV  @FC,R$
    DEC  R4
    SLA  R4,5
```

...which is adequate to demonstrate the difference to BASIC.

TERMINAL EMULATOR 2

The Terminal Emulator 2 module is designed for telecommunications, but as no suitable modem is available to connect your 99/4a to the Post Office network, you will not use that facility.

Some major users of the computer use the TE2 module to link their 99/4A to 'main frame' computers, using the TI computer as an 'intelligent terminal', which has its own programs and passes data to and from the larger computer.

Of interest to domestic purchasers however is the much improved speech facilities of the TE2 module.

With a speech synthesiser connected, your program in TI Basic can say anything that you wish it to. You also have control over pitch and emphasis.

The method used is to open a file:

```
OPEN #1:"SPEECH",OUTPUT
```

and then when you wish your program to say something, you PRINT to this file:

```
PRINT #1:"I CAN SAY ANYTHING YOU WANT ME TO"
```

Speech is much faster with TE2, and because there are no limitations on the string printed to the file, you may adjust pronunciation by changing your spelling.

Would you like your computer to read your program to you? This can be of help when checking a listing to your program, looking for a missing line for

instance.
Use:
LIST "SPEECH"

LOGO

LOGO is a language module, and requires the 32k memory expansion.Disk drive and controller are advisable.

LOGO is a 'build it yourself' language, in which you build up your own commands from a small set of 'primitives'.

It is not therefore a program to exchange programs in, but a language to learn with, and is extensively used in a few primary and junior schools.

A redrafted version of LOGO to be known as LOGO 2 has been announced from TI, with greater user memory and added features, but at the time of writing was not available.

LOGO is of great interest to schools, and you may find it useful if you have young children, or an interest in creating your own language, or learning to express yourself in a clear and logical manner.

The following is an extract from a TI LOGO procedure, and informs the computer how to carry out the command:BLINK:

```
TO BLINK
TELL :ALL
SC :RED
TELL TILE 32
SC [4 15 ]
WAIT 40
TELL :ALL
SC :WHITE
TELL TILE 32
SC [15 4 ]
WAIT 40
BLINK
END
```

MULTI PLAN

Multi Plan is a 'spread sheet' program module, and requires the 32k memory expansion and a disk system. A printer (with rs232 card) are advised but optional.

The program was written by Microsoft (whose versions of BASIC are widely used in American computers), and is similar to a popular program called VISICALC (not available for the 99/4a).

The idea of spread sheet is to set up data in rows and columns and tell the computer of the relationships between certain figures. You may then investigate 'what if...' situations by changing certain data, and allowing the computer to change the remainder in accordance with the relationships between the data.

The use of spread sheets is complex, but in making business decisions helpful information can be quickly calculated and obtained.

OTHER LANGUAGES

The following are announced by TI. In addition independent sources may be able to supply other implementations of these languages:

FORTH is announced on DISK requiring Editor Assembler module and 32k memory, plus disk system. Not yet released.

PILOT is announced on disk for P Code card and 32k memory expansion. Not yet released.

PASCAL is available on three disks, requiring the P Code card and 32k memory expansion.

CHAPTER NINE

PERIPHERALS

SYSTEMS:

The original expansion system comprised separate peripherals, each with their own power supply, which plugged into the right hand side of the computer and each other.
As the number of peripherals increased, this resulted in a number of electric supply cables, and the need for a very long desk.

Hence TI produced the PERIPHERAL EXPANSION BOX, which plugs into the right hand side of the console by means of a cable. It has its own power supply, but this is used for all the peripherals placed in it.
The TI BOX (as we shall call it) is supplied with a single interface 'card' which merely allows it to be connected to the 99/4a.
The 'cards' used in the Box are far more than the usual printed circuit board usually associated with expansion systems: each card is inside a strong metal sub chassis. The box itself is also very strong (and heavy) metal.

The box has space for a single disk drive, although it may be possible to use two low power 'half size' drives mounted side by side. TI do not provide half size drives and you will need to have a well informed and helpful dealer if you wish to fit them.

The standard TI Disk Drive is a single sided single density drive which uses soft sectored disks with 40 tracks. Each disk can store about 90k of information, and each disk can contain up to 127 named files.
To operate the disk drive you need the DISK CONTROLLER card, which is supplied with a DISK MANAGER module. The controller card can operate up to three disk drives: the second and third must have their own power supply and case, and are used outside the Box.
The Disk Manager allows you to test your disks, initialise them, change file and disk names, and provide a catalogue of disk contents.
You can use double density disks, but the computer will only use them as single density.
It is possible to modify the controller so that a double sided disk drive can be used, with the second side treated as DISK 2.
A Disk Manager 2 module and double sided drive have also been announced from TI, but are not available at the time of writing.

A disk system allows you to load and save programs or data much faster than from cassette. Also, because a disk does not have to be read in the same order it was saved, random access files are possible, for faster and more powerful data handling.

A program may be LISTed to disk as well as SAVEd to disk. A LISTed program is placed on disk in DISPLAY VARIABLE 80 format, and may be used with the TI WRITER module.

Extended Basic permits a program to be saved in MERGE format, to allow programs to be merged into each other, and also allows you to manipulate the program : you may create utilities which remove REM lines, or shorten variable names for instance.

Also with Extended Basic, a disk system will allow you to load a program which exceeds the 12k allowed under the cassette loading system.

MEMORY EXPANSION

The 32k MEMORY EXPANSION CARD adds 32k of CPU RAM to your system. It is NOT usable unless a suitable module is used.
Extended Basic programs may be up to 24k when the 32k expansion is fitted, but the tape loader can only load programs up to 12k.
However, with Extended Basic, when the 32k is fitted, in addition to the 24k for your program, you have about 13k for storing variables. Thus you may load a 12k tape program and not have to worry about the memory used for variables (for example, large arrays of data).

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 3am, or use the BBS to leave a message at 738-0617, 24 hours a day, 7 days a week.

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

AL PALMER.....VICE PRESIDENT.....594-9216

MARCELLE GIBSON.....SECRETARY.....233-2384

BILL SPONCHIA.....TREASURER.....523-0878

MICHAEL TAYLOR.....PAST PRESIDENT.....831-0143

PETER ARPIN.....SYSOP.....523-0017

RUTH O'NEILL.....NEWSLETTER EDITOR.....234-8050

TONY HOPKINS.....ADVERTISING.....746-4463

DAVE MORRISON.....LIBRARY CHAIRMAN.....737-4889

JACK McALLISTER.....CASSETTE LIBRARY.....225-6989

HENRI MONAT.....ARCHIVES.....824-0941

LUCIE DORAIS.....MEMBERSHIPS.....232-0393

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

ART GREEN.....ASSEMBLY HELP.....837-1955

DICK PICHE.....TECH.....521-8667

CLUB BBS.....SET MODEM TO 8N1.....738-0617

1989 RENEWAL
$25.00

NEW MEMBERS
$25.00

NAME _____

ADDRESS _____

CITY _____ PROVINCE/STATE _____

POSTAL 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.

FROM

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

EDMONTON 99er USER'S GROUP
P.O. Box [illegible]
EDMONTON, ALBERTA
T5J [illegible]