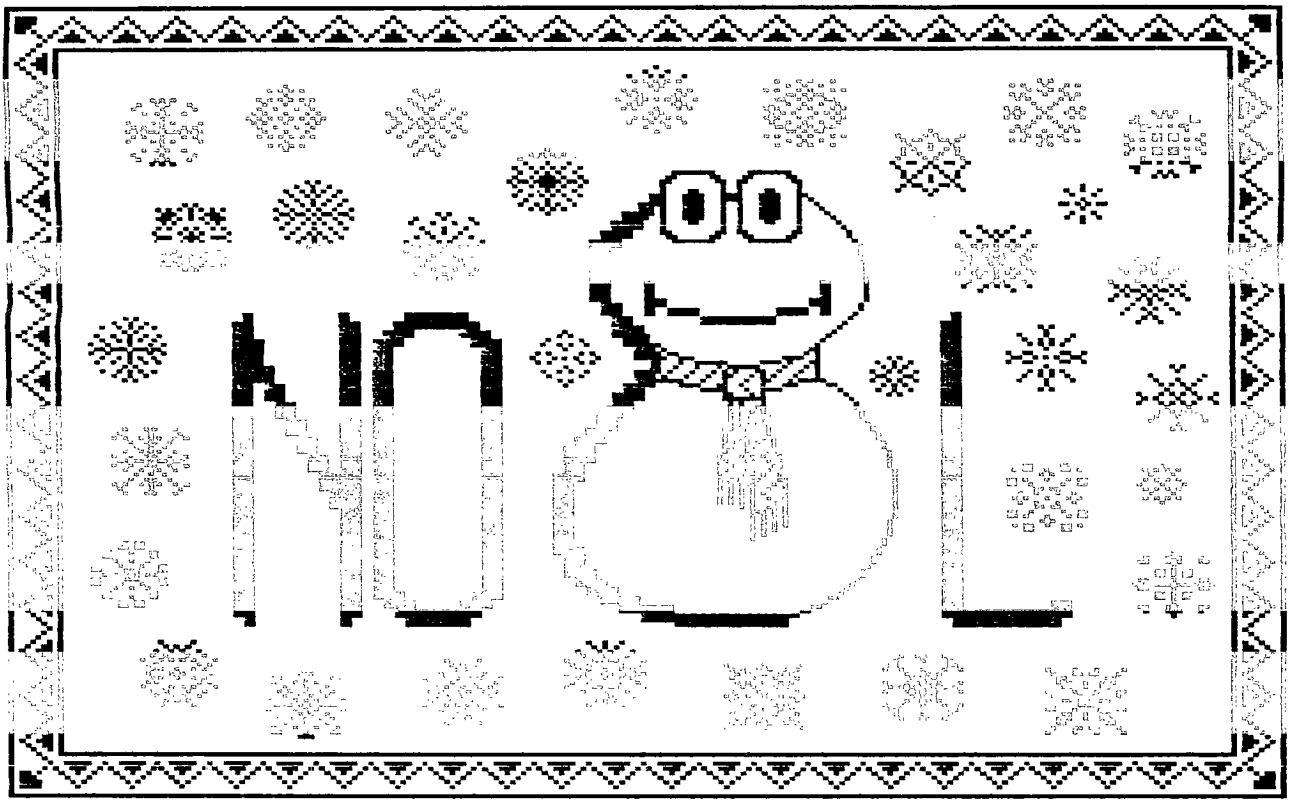
The Ottawa T.I.99/4A Users' Group

VOLUME 7 NUMBER 10.....DECEMBER 1988



DON'T FORGET THE MEETING -- December 6, 1988

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

COMING EVENTS

December Meeting:	December 6, 1988 7:30 p.m.	Merivale High School Guest: Floyd Donaldson of Buckingham.
January Meeting:	January 10, 1988	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, 1988 10:00 a.m.	Merivale High School Contact Bill Sponchia for further information.
4th Annual TI-FEST	April 29, 1989	Merivale High School Contact Jane Laflamme for details
Newsletter Deadline:	December 15, 1988	(or any time before that!)

EDITOR'S NOTES
from
Ruth O'Neill

Well, this is the last newsletter of 1988. We'll be mailing out the January newsletter to all of this year's members, but the December meeting is still the best time to renew your memberships. Talk to Lucie Dorais there, or send in your renewals to the address on the front of the newsletter. Make sure you don't miss out on any newsletters or club activities through procrastination!

In the "Coming Events" schedule, you may notice that there is no Saturday workshop planned for December. The next workshop is on utilities, and will be presented by Tom Bentley and Charles Earl on January 21 at 10:00 am at Merivale High School. If you have any ideas for workshops you would like to present or would like to see someone else present one on a certain topic, contact Bill Sponchia.

In the schedule above, please note that the January meeting is NOT on the first Tuesday of the month, but on the second. (The school is not available on the first Tuesday.) In case you want to start filling in your new calendars, here are the rest of the known dates for 1989 meetings:

January 10, 1989
February 7, 1989
March 7, 1989
April 4, 1989
May 2, 1989
June 6, 1989

All start at 7:30 pm at Merivale High School in Nepean.

Just a reminder... contact Bill Sponchia if you are interested in joining the beginners assembly language class. It isn't under way yet, but we expect it to be soon.

Thanks go to Lucie for the cover art, as usual - you'll find the program she used for the snowflakes in her "Fast Extended Basic" column this month.

The President's Two Cents' Worth
by Jane Laflamme

Back from Chicago, and what a time was had by the seven who represented Ottawa. Two of us drove down separately, (one from Ottawa and one from North Bay), and the balance of us flew, complete with two computers, one GENEVE and one TI. Not nary a scratch did they incur!

There were several new pieces of software shown at the fair through Genial, Asgard, Myarc, and others, but the item that appeared to steal the show was the revolutionary new word processor written by Charles Earl, and marketed by Asgard. Charles, and Ruth O'Neill (who wrote the manual) didn't have a moment to themselves.

We attended a "mixer" on Friday night, the show on Saturday, and a wind-up dinner in the evening. This year they had a new venue, the Holiday Inn in Rolling Meadows. All functions were held there -- very convenient for us. At the dinner, we once again presented a trophy, (our sixth to people or organizations that have made a significant contribution to the TI and 9640). This fair's recipient was MICROpendium, a valued magazine dedicated solely to our computers. We were lucky to have Mr. John Koloen, the publisher, in attendance at the dinner to accept the trophy in person. Our thanks also to Laura Burns, the Editor. We Canadians appreciate your quality magazine.

Our meeting in December will feature software brought back from Chicago (e.g. FirstBase by Warren Agee) and we have Mr. Floyd Donaldson of Buckingham, who will be showing us his software.

As a matter of interest, for our January meeting, we have invited Mr. Edward Cheung, a lawyer who specializes in computer law. He will be speaking to us about copyrighting computer software. This is of special interest to our group with the anticipated release of our all new and exciting disk manager system, DM2000. I am hoping that the group will be well represented that night, so how about inviting programmers for other computers to our meeting, as well? I will also pass on invitations to our sister groups in the area, Almonte and Carleton. To our sister groups in North Bay, Sudbury, Peterborough, Toronto, and Montreal, please feel free to come and join us, in January or anytime, for that matter! (Please note that the January meeting will be held one week later, on the SECOND TUESDAY of the month, January 10th, 1989)

While I am on that subject, we would like to have a varied program for this year's meetings, but we are coming up short on ideas. Please let any executive member know what kind of presentations you would like to see, or if you know of a special guest that we could invite.

Until next month....

Jane

BROWSING THE LIBRARY
--with DAVE MORRISON

Let me first apologize for not realizing that C99REL4A was only part of Clint Pulley's release! Part B will be available at our next meeting as will a single-sided version of TI-TOOLS2.

As usual, members will be able to obtain any of the disks that have been offered since March of this year.

For our December selection, I have chosen (without much difficulty!) an album of Christmas music to which I have added Ray Kazmer's 1987 Christmas gift to all T.I.ers - WOODSTOCK. This disk will be available in your choice of single or double sided.

For those members who are interested in C99 programming, I will make available a hard copy of all Library files related to that subject. I hope that some of you will take the time to review the list of disks and files and indicate to me which should be retained and which should be discarded.

I have taken on the added responsibility of answering all mail relating to the Library and to Fairware as Stephen Bridgett has been devoting his time to an unusual pursuit - Programming! Strange, he always appeared to be such a nice guy! Anyway, for you out-of-town members, if you have requests or comments to which you require a response, please allow time for the mail to reach me - it sometimes passes through several pairs of hands; not all of those being employees of Canada Post!

In closing, let me offer you my very best wishes for a Merry Christmas and a Happy and Prosperous New Year.

Dave

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 has collected a few of the more interesting items together here. Some of them are extracted and/or paraphrased, but the original information and intent have been preserved.

FORTH - John H. Carver Jr. has announced his intention of acting as a "Clearing House" for FORTH programmers and asks that anyone "even remotely interested" write to him at RR#1, BOX 125-2, Bringhurst, Indiana, 46913 or leave messages on Genie; Compuserve; The Source, or Delphi. He points out the existence of the TI-FORTH International Information Center, 4122 Glenway St., Wauwatosa, Wisconsin 53222, which has public domain FORTH applications and tutorials. Also, the FORTH Interest Group, PO Box 8321, San Jose, California 95155, offers membership for US $30 per year.
(From: TIDBITS (Mid-South 99/4A Users Group, Memphis, Tennessee) Feb 88.)

E-MAIL - A national (U.S.) E-MAIL registry, The White Pages, is available to modem users and lists thousands of subscribers, (individual & companies). After a free registration, the user may make a search for someone, and would receive information on which commercial or public telecommunications system he can be found by E-MAIL. Each search is the cost of the call plus 20 cents for each name found. You might wish to register so that others can find what service(s) you use. To register, set your terminal for 7E1 (300, 1200, 2400 baud, and call 1-800-622-0505.
(Note: This service may not be available to Canadian subscribers)
(From: TIDBITS (Mid-South 99/4A Users Group, Memphis, Tennessee) Feb 88.)

QUICK COPYer II - QUICK COPYer II is published by Quality 99 Software (US $9.95+S&H). The programme will load quickly via XB, MM or EA. What QUICK COPYer does is impressive. It can copy all 360 sectors and a SSSD disk in three (sometimes two) passes when using XB or EA. With MM it never takes more than two passes. It can copy from SSSD to DSSD and vice-versa, if there is room. The entire disk does not have to be copied as you can specify which files are to be copied. You can also format a blank disk in any format that your system supports. For the reviewer, the programme worked perfectly and never crashed or miscopied. QUICK COPYer autoloads from XB, but be sure to read the instructions for loading via MM or EA. All options are selected from a menu which gives you the appropriate instructions when necessary. The reviewer points out one disagreeable oversight in that if he wanted to copy two SSSD disks onto a single DSSD disk, only the first could be copied with QUICK COPYer as copying the second disk would destroy (by overwriting) the files previously copied. He had to copy the files from the second disk by using DM-1000 or DMII. The reviewer finally states, "Despite this shortcoming, QUICK COPYer is a program that I use often. I works flawlessly and does everything that is claimed for it. I just wish the programmers had set their sights a tiny bit higher..."
(From: Twin TIers Users' Group Newsletter (Feb 88), Elmira, New York.)

CATCOM - A review of Marty Kroll's CATCOM (the companion disk to his well-known CATLIB).

Upon booting the programme you are faced with a menu that is as straight-forward as that in CATLIB. CATCOM loads the data and disk files from your CATLIB and allows you to add keywords and comments to each file in your library. It thus creates a database of commented files/disks and allows you to create nicely-formatted printouts of this information. As the reviewer comments, "Clean & simple". He goes on to say, "I found the program easy to use and very forgiving if you blunder from time to time. One real nice feature is if you delete some files from your CATLIB data, Companion will update its file if you choose. The documentation that comes with Catlib Companion is well written and complete and will have you using this fine program in no time at all. All in all another fine program from this prolific fareware writer!" He (the reviewer) then points out that it must be kept in mind that the USER must assign all key-words and comments and that in a library of 500-2000 files, "this setup could take quite some time". In conclusion the reviewer states, "I recommend that every one who has Catlib check this program and see if it will enhance your diskfile tracking".
(From: Southwest Ninety-Niners, Tucson, Arizona, Aug 88.)

(Note: Both CATLIB & CATCOM are available from our Library).

ANNOUNCEMENT!!!!

Ottawa now has a Compuserve node! Now we can call Compuserve directly, without paying the additional Tymnet charges. Call (613) 837-5427 at 300 or 1200 bps. If you do not yet subscribe to Compuserve, but are interested in joining, please call Ruth O'Neill at 234-8050 before you buy a subscription kit. You should also give Ruth a call if you are interested in seeing a Compuserve or Delphi conference. That's the best way to find out when the next group log-on will be.

The 1988 Chicago TI Fair:
A first-timer's eye view
by Ruth O'Neill

Chicago! What an experience! The excitement! The people! My only regret is that it was my first, so I've missed all the others. You can be sure I'll be there next year, though.

Jane Laflamme, Ralph Kuhn, Charles Earl, Bill Sponchia and I flew down to Chicago on November 11th with two full systems to use at the fair. We had made all our preparations ahead of time, so we had no trouble getting it all through customs, once we convinced the officials that we were carting it around for fun. Michael Taylor drove down and met us there, as did Bob Boone. A group also came down from Toronto, so there were quite a few Canadians there.

Everything was held at the hotel where we were staying, so we didn't have to waste any time travelling to and from the fair. There was a social Friday night, where it was fun putting faces to names I've read about or chatted with on the networks.

The fair itself on Saturday was a very busy place. There were thirty-odd vendors and well over 500 paid admittances, but it seemed like even more. I didn't get a chance for a leisurely tour of the fair, but I do know there was a lot to see. I managed to sneak out to catch the Genial presentation, and I'm *looking* forward to using some of the new products I saw there - especially Picture Transfer (by Paul Charlton) and Hypercopy (by Mike Dodd). Both of these are for the 9640 only, by the way. Unfortunately, once I'd seen the presentation, it was a little hard to buy the packages, since they had sold out of almost everything. I did manage to get an autographed copy of FirstBase (yes, Warren Agee was there too!) earlier in the day, though. FirstBase is Warren Agee's new database, produced by Genial Computerware, for any who haven't heard... and yes, it is available now!

Asgard had a busy day, too. They took orders for Press, which unfortunately wasn't quite ready for release, and also sold most of the other packages they had brought. People certainly brought a lot of money to the fair! Disk Only Software sold out of Hard & Floppy Disk Controller Cards pretty quickly, too.

The fair was exciting, and I wouldn't have missed it. More than the fair, though, I enjoyed meeting the people -- especially during a couple of all-night chats. (Or nearly all-night -- Going to sleep at 5 in the morning hardly seems worthwhile.) We stayed in Chicago an extra day to recover from all this, which made the return trip much more pleasant. As Jane mentioned, all our equipment made it safely home, although Bill's suitcase and Jane's monitor couldn't quite keep up with us.

Interesting products at the fair? I already mentioned Picture Transfer and Hypercopy, and there was another interesting Geneve-only program there-- Diskassembler for MDOS. For both the /4a and Geneve, there were FirstBase, MacFlix, Batch-It, Disk of Dinosaurs, and other products from both Genial and Asgard. Barry Traver was there with Genial TRAVelER, too. I'm sure there were many products there that I missed seeing because I was pretty busy answering questions about Press, but I wouldn't have had it any other way.

Thank you very much to the Chicago Users' group, especially those who arranged an excellent fair. Thank you also for the door prize I won -- an extensive set of back issues of the Chicago group's newsletter. (It is currently out on loan, but if anyone else in our group would like to read them, just let me know.)

I'll be back next year, with as many people as I can talk into it!

EXPANDING EXTENDED BASIC'S POWERS,
WRITING ASSEMBLY ROUTINES: PART 2
By David Caron

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

```
       DEF  START     *This places to 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
                      *pages 415-416 of the Editor/Assembler manual.
NEWREG BSS  32        *This is where our workspace will be. 32 bytes
                      *are reserved since 16 registers X 2 = 32
STRING TEXT 'START'   *The TEXT directive places the characters from
                      *the string in the operand field directly in
                      *memory in the form of their ascii values.
START  LWPI NEWREG    *This instructs the computer to use the memory
                      *locations indicated by NEWREG as the work-
                      *space registers.
       LI   R0,>0000  *Loads register zero with the 16-bit data 0
       LI   R2,STRING *Loads R2 with the value of STRING which is
                      *the first address containing the series of
                      *ascii bytes making up the word START.
LOOP   MOVB *R2+,R1   *The asterisk in this case does not mean a
                      *comment is starting but rather, now read
*carefully: The BYTE at the address indicated by R2 is COPIED
*into R1, so the value of R1 becomes 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
```

And that's all there is to it! Each time the loop is repeated, the next Byte at the address in R2 is put in the most significant byte of R1, added by 96 for compatibility with Extended Basic and sent out to the screen until R2 >STRING+4, which means that we have sent all five characters of START to the screen, and the loop exits. Incidentally, you may have noticed that although we have declared VMBW, VSBR and VMBR, we have not actually used them. Do not worry. We will be using them in the future and it is nice to already have them defined so that we can use them without looking up the calling address later on.

WHY 96 MUST BE ADDED TO ALL CHARACTERS BEING SENT TO THE SCREEN

Take a look at page 403 in the Editor-Assembler manual. Disregard everything except the Screen Image table and the Pattern Generator table. Notice that the Screen Image table starts at >0000, which is exactly what the above assembly routine takes for granted. Everything one sees on an Extended Basic screen is

in this table. The table contains 768 bytes as does the Extended Basic screen (24 X 32 =768). Each byte can represent any of the 256 characters which exist in the Pattern Generator Table.

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

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

HOWEVER, unlike the VDP RAM table on page 403 which is set up for Editor-Assembler, the Pattern Generator Table does not start at >0800 or 2048 but at >0000 or 0, the same address as the screen image table. Furthermore, the ACTUAL data for the Pattern Generator Table has not been moved back 2048 bytes but 2048-300 = 7496. This means that the data for the Pattern Generator Table starts at VDP address 768 and the Pattern Table itself begins at 0. The programmers of Extended Basic moved the Pattern Table Back so that they could use VDP memory starting at 2048 for Program Space. The reason the actual Data did not get moved back as far is due to the fact that the first 768 bytes would have been erased by the Screen Image Table. The net effect is that the Data for the Pattern Generator Table starts 768 bytes AFTER the Table itself, or 768/8 = 96 character blocks later.

This is essentially why 96 must be added to the character ascii code, because the Data for the character you want does not appear until 96 character blocks later. If you still desire a more detailed explanation you will have to know all about VDP registers and how they relate to each other.

You are now probably wondering why Extended Basic lets you access only 112 characters when there appear to be another 48 available (256-96-112=48). These 48 characters translate into 384 bytes of VDP memory (48 X 8) which is where sprite data is stored as well as Extended Basic's variables. Therefore, this memory cannot be used for character definitions.

In my next article I will introduce four more Extended Basic Assembly utilities and explain how they can make the above assembly routine much more versatile. If you have any questions do not hesitate to call me at 745-4618.

FAST EXTENDED BASIC

LUCIE DORAIS

As we say in French: "Jamais deux sans trois..." (or, more apologies...). Line 270 has mysteriously disappeared from last month's Listing; there was enough information in the text to rebuild it, but here it is:

```
270 ACCEPT AT(20,8)VALIDATE(DIGIT)SIZE(-5)BEEP:JP :: ACCEPT AT(20,23)
    VALIDATE(DIGIT)SIZE(-5)BEEP:BK :: BST=INT(BK/2)
```

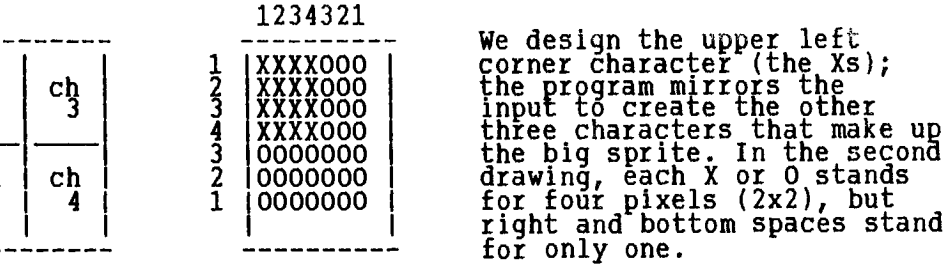
Now to this month's topic: SNOWFLAKES! Hopefully, as you read this, they are still confined to the cover page of this Newsletter. But how did they get there in the first place? No, I did not painfully draw them with Graphx or Ti-Artist, but used the program below to design them. I wrote another utility to transform them into Ti-Artist instances (next month's program).

```
100 REM ** SNOWFLAKE ** L. Dorais / Ottawa U.G. / Oct. 1988
110 REM
120 CALL CLEAR :: CALL MAGNIFY(3) :: OPTION BASE 1 :: DIM SC(10)
    SD$(11),C$(4)
130 GOTO 160 :: A$,C,C1,C2,CH,CN,FN,FN$,H,HEX$,HI,HI$,K,LO,LO$,R,S,
    SP$,TM,TR,X :: CALL KEY :: CALL SCREEN
140 CALL CHAR :: CALL HCHAR :: CALL VCHAR :: CALL GCHAR :: CALL COLOR
    :: CALL SPRITE :: CALL DELSPRITE :: CALL LOCATE
150 CALL CUR :: CALL VERT :: CALL AR :: !@P
```

```
160 HEX$="0123456789ABCDEF" :: FOR X=1 TO 9 :: SC(X)=4*(X+23):: NEXT X
    :: SC(10)=140
170 A$="FFFFFFFFFFFFFFFF" :: CALL CHAR(35,A$,135,A$&"FFFFC3C3C3C3FFFF",
    137,"",138,"",139,"") :: CALL COLOR(13,16,1,14,9,2)
180 ! ** display screen and define flake **
190 DISPLAY AT(20,1):"CURSOR:  [1] ON/OFF [V]IEW": :" W E R   [3] REDO
      [G]O DEV":" S D":" Z X C   [5] NEXT   [Q]UIT" 1 menu
200 CALL VERT("13579",20):: CALL VERT("24680",28):: FOR X=23 TO 29 ::
    CALL VCHAR(2,X,35,16):: NEXT X  ! status board
210 FN=FN+1 :: IF FN=11 THEN 490 ELSE TR=(INT(FN/2)=FN/2) :: CALL
    AR(FN-1,-(TR+1),32):: CALL AR(FN,TR,62+TR*2) ! FN=current flake no.
220 FOR X=2 TO 18 :: CALL HCHAR(X,2,35,17) :: NEXT X ! design grid
230 R,C=10 :: CALL SPRITE(#11,136,9,73,73)
240 CALL KEY(1,K,S) :: IF S=0 OR K>19 THEN 240
250 ON K+1 GOTO 260,240,270,280,300,290,310,240,220,240,500,240,240,
    380,320,330,240,530,580,350
260 CALL CUR(R,1) :: GOTO 340   ! K=0 "X" down
270 CALL CUR(C,-1) :: GOTO 340  ! K=2 "S" left
280 CALL CUR(C,+1) :: GOTO 340  ! K=3 "D" right
290 CALL CUR(R,-1) :: GOTO 340  ! K=5 "E" up
300 CALL CUR(R,-1) :: CALL CUR(C,-1) :: GOTO 340 ! K= 4 "W" up/left
310 CALL CUR(R,-1) :: CALL CUR(C,1) :: GOTO 340  ! K= 6 "R" up/right
320 CALL CUR(R,1) :: CALL CUR(C,1) :: GOTO 340   ! K=14 "C" down/right
330 CALL CUR(R,1) :: CALL CUR(C,-1)              ! K=15 "Z" down/left
340 CALL LOCATE(#11,R*8-7,C*8-7) :: GOTO 240
350 CALL GCHAR(R,C,CH) :: CH=35-100*(CH=35) :: CALL HCHAR(R,C,CH)
360 CALL HCHAR(20-R,C,CH) :: CALL HCHAR(R,20-C,CH) ::
    CALL HCHAR(20-R,20-C,CH) :: GOTO 240
370 ! ** flake definition **
380 C$(1),C$(3)="" :: C$(2),C$(4)="00"
390 FOR R=3 TO 10 :: CALL LOCATE(#11,R*8-7,9)
400 CN=1 :: C1=3 :: C2=10 :: GOSUB 440   ! define char. 1-2
410 CN=3 :: C1=11 :: C2=18 :: GOSUB 440  ! define char. 3-4
420 NEXT R :: SP$=C$(1)&C$(2)&C$(3)&C$(4) :: CALL CHAR(SC(FN),SP$) ::
    SD$(FN)=SP$
430 R=8*(INT((FN+1)/2)*3)-7 :: C=185-24*(INT(FN/2)=FN/2):: CALL
    SPRITE(#FN,SC(FN),16,R,C) :: GOTO 230
440 H=0 :: FOR C=C1 TO C2 :: CALL GCHAR(R,C,CH):: IF CH=135 THEN
    H=H+2^(C2-C)
450 NEXT C :: HI=INT(H/16):: LO=H-16*HI
460 HI$=SEG$(HEX$,HI+1,1) :: LO$=SEG$(HEX$,LO+1,1) ::
    C$(CN)=C$(CN)&HI$&LO$
470 IF R<10 THEN C$(CN+1)=HI$&LO$&C$(CN+1) :: RETURN ELSE RETURN
480 ! ** other routines **
490 A$="ALL FLAKES DONE!" :: FN=10 :: GOTO 510
500 IF SD$(FN)="" THEN A$="FLAKE NOT DEFINED!" ELSE 210
510 FOR X=1 TO 20 :: DISPLAY AT(1,1):A$ :: DISPLAY AT(1,1):"" :: NEXT X ::
    GOTO 240 ! flash a message
520 DISPLAY AT(1,1)BEEP:"DEV:";:: ACCEPT AT(1,6)SIZE(-15):FN$ :: IF
    FN$="" THEN 560 ! go printer, disk or cassette
530 A$=SEG$(FN$,1,3) :: IF POS("DSKPIORS2CS1",A$,1)=0 THEN DISPLAY
    AT(1,23)BEEP:"ERROR!" :: GOTO 520
540 FN=FN+(SD$(FN)="") :: OPEN #1:FN$,OUTPUT,VARIABLE 65 !see text for CS1
550 PRINT #1:FN :: FOR X=1 TO FN :: PRINT #1:SD$(X) :: NEXT X :: CLOSE #1
560 DISPLAY AT(1,1):"" :: IF FN$<>"CS1" THEN 240
570 CALL DELSPRITE(ALL) :: CALL CLEAR :: CALL SCREEN(2) ::
    FN=FN+(SD$(FN)="") :: RANDOMIZE ! quit
580 FOR X=1 TO FN :: CALL SPRITE(#X,SC(X),16,193,RND*255+1,RND*80,
    RND*160-80) :: NEXT X
590 CALL KEY(0,K,S) :: IF S=0 THEN 590 ELSE END ! press any key to end
600 !@P+ ** user-def subs **
610 SUB CUR(X,Y) :: X=X+Y :: IF X=11 THEN X=10 ELSE IF X=2 THEN X=3
620 SUBEND
630 SUB VERT(A$,Y) :: FOR X=1 TO 5 :: DISPLAY AT(3*X+1,Y):SEG$(A$,X,1) ::
    NEXT X :: SUBEND
640 SUB AR(X,Y,Z) :: CALL HCHAR(INT((X+1)/2)*3+1,21-Y*10,Z) :: SUBEND
```

Most "flake" or "quilt" design programs use a mirror routine, but it has one major flaw: if you mirror all characters, the middle ones (up or across) are doubled... not very aesthetic. SNOWFLAKE also uses a mirror routine, but it does not double the middle lines: the finished flake is thus 15x15 pixels in

size, "padded" to fit the standard 16x16 pixel size of a big sprite (four characters):



After the usual pre-scan and initializing, you are presented with the working screen: a big square at the left, to design the flake, and a status board at the right, to tell you where you are, and display the flakes already done. The Menu at the bottom of the screen instructs you on how to move the cursor, how to put a pixel on or off (toggle function), and other options: REDO the flake, NEXT flake, VIEW the flake as a sprite at right (you cannot go to Next if you have not done this), GO DEVICE (to get the sprite definitions on printer, disk or cassette), and finally QUIT: when you do, all your flakes will fall in a storm until you press a key.

Initialization is easy to understand, except for part of line 160: the array SC() keeps in memory the ASCII values of the characters that will be used for sprites; each uses four characters, so the SC(1-9) array will be filled with chars. 96 to 128, step 4, and SC(10) with char. 140, since we reserve char. 135 to 139 for other uses. In line 170, char. 135 is the white pixel, and 136 will be our red cursor, a sprite; but since we do a CALL MAGNIFY(3) to display big sprites, the cursor is in fact also four char., so we make the following three transparent.

The CALL VERT routine in line 200 prints vertically. The CALL AR routines in line 210 are a bit hard to understand, but keep on -- I will explain that later, when I introduce relational expressions. I use them a lot in this program; remember, this series started as tutorials...

Once again, as I did last month, I chose the Keyboard "1" for the CALL KEY because it enables the arrow keys to be read with or without the FCTN key; also, it makes the key check a lot easier, since in line 250 we check all 20 keys (keys 0 to 19: we add one to the value of K, because ON-GOTO does not work with a zero value); if a key is not needed, we go back to the CALL KEY statement in line 240. This setup also allows for easy modification of the menu: just change the line number for any key you want to add or delete from the menu list.

Our designing board is 17x17 (our 15x15 grid plus a nice border), but the cursor moves only in the upper left portion, between columns 3-10 and rows 3-10 (an 8x8 grid for character 1). This makes out of range movements easy to check, and prompted me to put the cursor movement in a user-defined sub CUR, with passed parameters for each movement: move Row or Column, positive (right/down) or negative (left/up). Of course, if you move diagonally, you need to CALL CUR twice. Line 340 puts the cursor on the screen; it is a sprite, therefore Tex derives its location from the current R and C values. By using a sprite, your design is not modified by the cursor movements.

Moving the cursor does not draw the flake: when you are where you want to be, press "1"; if the character already there (CALL GCHAR) is character 35, black, it will become character 135, white, and vice-versa. This is what the relational expression "(CH=35)" means (see further). Tex then puts all four characters on the screen: first at (R,C), then at values derived from them relating to the middle row and column, always 10. No problem with the characters that fall in the middle lines or dead center: Tex will copy them to themselves so quickly that you will not see it happening.

Now for the heavy stuff: RELATIONAL EXPRESSIONS (XB Manual, page 41). We know that IF-THEN-ELSE statements use a lot of code, and some time to perform. Basic has a nice way to cut the work for Tex: instead of saying IF CH=35 THEN CALL HCHAR(R,C,135) and its opposite, you can put the expression to be tested into brackets (essential!) and include it in a longer statement. If the expression

is true, Tex will return "-1", and you can use this value in the statement. Here, if CH=35, the ()s get a value of "-1", and the total new value of CH will be 35-100*(-1), i.e. 35-(-100), i.e. 35+100, i.e. 135, white. If the character read by GCHAR is 135, the expression is false and returns a "0" value, for a new value of 35-100*(0), i.e. 35-0 i.e. 35, black. Faster coded than explained!

Back to line 210, where I used the variable TR to store the truth/falseness of the (if) expression "(INT(FN/2)=FN/2)"; in other words, is FN, the current flake number, even or odd? If the expression is true, i.e. if FN is even, TR=-1; if it is odd, TR=0. That TR value is then used to display the pointing arrow to the current flake on the right blackboard, and erase the previous one.

The first parameter passed to the AR sub, based on the current flake number (incremented just before the CALL AR statements), has to do with the row, the second with the column: this is the one that needs to know if FN is odd or even. I arrived at those equations by doing a list of what I have, what I want, and finding the common values: both flake numbers and blackboard row numbers are linear progressions; column is either 21 or 31, depending if FN is odd or even. The third parameter is the character to HCHAR: space to erase a previous arrow, 60 or 62 for a "<" or a ">", this value again using TR. Remember, true is "-1", thus we will get character 62+(-1*2), 62+(-2), 62-2=60, the ASCII value of "<", for the even number arrow in column 31. Vice-versa: TR=0 will give us ASCII 62, ">", for the odd arrow.

FLAKE DEFINITION: To transform our flake design into a sprite, we need to define the four characters that will make it up. Instead of keeping the on/off status of each pixel in an array, like most CHARDEF-type programs, we read each pixel/character directly from the screen, adding the equivalent value of the powers of 2 for ON pixels to calculate the value of each row of two 8x8 grids (left and right top characters: the right border char. will become the last pixel of our char. definitions).

Line 380 initializes the definition strings C$() for our four characters: 1 and 3, the top ones, will be all filled in (8 rows), while the bottom ones, 2 and 4, getting only 7 rows filled by the design, need an ending "00" for the last row. For each row, we reLOCATE the cursor to monitor the progression of the routine. We then GOSUB 440 to get our character definition, from line 400 for the top left character 1 (Char. Num. CN=1, starting col. C1=3, ending col. C2=10), line 410 for the top right character 3. Mirroring to the bottom characters 2 and 4 is done in the sub-routine.

For each row (sub in lines 440-470), we CALL GCHAR every column; the variable H, which will contain the decimal byte value for the row, is set to zero at the beginning of the sub. We add up the bit values according to the position of the ON bits in the byte, with one byte (eight bits) for each row; the normal way is to divide the row in two, get the hex value 0-F (0-15) for left then right, but it is faster to do the whole row all at once; the resulting value of H will be bigger (0-255), but nobody will know. Line 440 adds a value to H only if the pixel in on (CH=135). Working from left to right, this value is 2^(7 to 0); the power value is derived from the current column. Another little graphic:

```
 Eight bits:    1    1    1    1    1    1    1    1
Powers of 2:  2^7  2^6  2^5  2^4  2^3  2^2  2^1  2^0
      Value:  128   64   32   16    8    4    2    1
              ------------------- -------------------
Powers of 16:   HI nibble 16^1       LO nibble  16^0
```

We must now get the hex equivalent of the H value; to do this in the fastest way, we divide H into a left nibble HI and low nibble LO (a nibble, four bits, is a half-byte); each value will be between 0 and 15, and it is easy to get the hex equivalents 0-F into HI$ and LO$. "HI$&LO$" is thus the hex value for our complete row, and we add it at the end of the C$(CN) string for the top character; since the bottom character is its mirror, we add it at the beginning of the C$(CN+1) string, but only if the row is not 10, our middle line, that we don't want to mirror.

Finally, back to line 420, we do the next row... When all four characters are filled, we build the sprite definition into SP$, then we use the character with the ASCII number as stored in SC(FN) to display the sprite on the screen (line 430, which again uses a relational expression for the column depending if the current flake is odd or even); we also save the definition into an array SD$(),

to use when we print or save all our flakes.

Don't worry if you don't understand everything: just type the program. It works much faster than it is explained! The "other routines" are much easier. You get to line 490 if you press NEXT when you already have defined 10 flakes, and to line 500 if you press it when your current flake has not been VIEWed (i.e. defined as a sprite); in both cases, a flashing message appears, and you are returned to the menu CALL KEY.

Lines 520-560 is an all-purpose saving routine, accessed when you press [G]O DEVice. You can enter your printer filename to get a Printout of the sprite definitions, or you can Save them to disk or cassette; line 530 checks for filename errors. By using the DIS/VAR 65 format, you can print (VARiable), but you will not confuse your disk file with a TI-WRITER file (if you want to, just replace 65 with 80!). Cassette users need to change the opening attributes to "FIXED 128", because CS1 can only use FIXED formats of 64, 128 and 192 bytes; and we cannot use 64, since each sprite definition, already 64 char.-long, needs a 65th byte for its length. Line 540 deducts one from FN (last current flake) if that current flake has not been defined yet (another relational expression!).

This is my longest text to date, and it is time to [Q]uit, which we do in lines 570-590. Cassette users go there directly after a SAVE, since the screen will be disrupted, so save only when you want to quit! The screen changes to black, and all the flakes you have created (minus last one if not defined) get random column location, down direction, and right or left direction. The starting sprite-row, 193, is below the visible portion of the screen, so you don't witness the positioning of the sprites. To END the SNOWSTORM, press any key.

TI BASIC continued from November
by **Steven Shaw**

The module is compatible with programs which have been LISTed to disk (eg LIST "DSK1.PROGNAME") which allows programs to be inserted into text, and also manipulated with the various editing options allowed.

TI Writer can also be used to create or edit any disk file using 'DISPLAY VARIABLE 80' format, such as the Editor/Assembler uses for machine code. You may use the TI Writer to edit data created and used in your own programs.

The screen is changed to 40 columns, but the TI Writer page is 80 columns long. These 80 columns are displayed in three windows, covering columns 1 to 40, 20 to 60 and 40 to 80.

Although the page is 80 columns wide, by using the commands provided with the TEXT FORMATTER (one of the TI Writer programs), it is possible to print up to the maximum length your printer will allow.

The module allows the use of any printer connected to the RS232 Card, using either the serial or parallel interfaces (see next chapter).
The command codes used by your printer can be inserted into your text, to allow you to switch say from normal print to italic print.
Centering of text is possible using TI Writer, as is 'right justification', where all lines finish in the same column at the right (the normal style of a book). TI-Writer does this by inserting extra spaces between words. The result is quite effective.

If you have the peripherals and you write fairly often,a word processor may be of use to you. This module is effective, and considering the large number of different commands a word processor must be able to handle, it is fairly easy to use.

The module is compatible with programs which have been LISTed to disk (eg LIST "DSK1.PROGNAME") which allows programs to be inserted into text, and also manipulated with the various editing options allowed.

TI Writer can also be used to create or edit any disk file using 'DISPLAY VARIABLE 80' format, such as the Editor/Assembler uses for machine code. You may use the TI Writer to edit data created and used in your own programs.

The screen is changed to 40 columns, but the TI Writer page is 80 columns long. These 80 columns are displayed in three windows, covering columns 1 to 40, 20 to 60 and 40 to 80.

Although the page is 80 columns wide, by using the commands provided with the TEXT FORMATTER (one of the TI Writer programs), it is possible to print up to the maximum length your printer will allow.

The module allows the use of any printer connected to the RS232 Card, using either the serial or parallel interfaces (see next chapter).
The command codes used by your printer can be inserted into your text, to allow you to switch say from normal print to italic print.
Centering of text is possible using TI Writer, as is 'right justification', where all lines finish in the same column at the right (the normal style of a book). TI-Writer does this by inserting extra spaces between words. The result is quite effective.

If you have the peripherals and you write fairly often,a word processor may be of use to you. This module is effective, and considering the large number of different commands a word processor must be able to handle, it is fairly easy to use.

MINI MEMORY MODULE

The mini memory module carries out a number of functions, but only one at a time:
You may use it for ONE of:
FILE HANDLING:
The module itself can be used in a TI Basic program as though it was a single disk file called "MINIMEM", and all the file handling commands available with disk drives will work with the module. It has a battery backup, and the information you store in the module will therefore remain after you switch your console off.
The module permits you to use the 32k Expansion Memory as a second 'solid state disk drive' called "EXPMEM2", which may store up to 24k of data. This data is lost when the 32k expansion is switched off.
Using either the module or the 32k expansion as data files, the information is retrieved even more quickly than with a disk drive. The computer does not have to waste time in moving a disk drive head over the disk.
It is possible to store data in the module or expansion memory with one program, and then to access the data with a second program, provided you do not reset the system by using QUIT or removing the module or power supply. This may help you to run a long adventure program for instance, by first placing the text into the memory and then loading your control program.

PROGRAM STORAGE:
A small program (up to 4k) may be stored in the module using SAVE MINIMEM and recovered using OLD MINIMEM. The program is loaded almost instantly.

ASSEMBLY LANGUAGE ACCESS:
With the module a cassette is supplied with a 'line by line assembler' which provides a primitive and difficult to use method of writing your own machine code programs.
You will need to purchase the Editor/Assembler manual for information on the 99/4A Assembly language, and should be aware that the manual is not written for the novice.
The LBLA itself occupies the module, and the maximum machine code program you may write with it is therefore about 750 bytes.
A few machine code games are now appearing on cassette which can be loaded into the module.
The minimemory provides a low cost entry into the field of machine code programming, but at present no book suitable for the novice is available.
Machine code is a 'low level' language, which is not as easy to use as BASIC.Because the computer does not have to translate the commands, a machine code program may be as much as 1600 times faster than a TI Basic program.

EXTENSIONS TO BASIC:

The mini memory adds some commands for use in your TI Basic programs, allowing you to PEEK and POKE both CPU and VDP memory, and to obtain the hexadecimal string defining any character:

PEEK and POKE are used in many computers to look at and change the contents of one single memory location in the computer. The 99/4A console has 16k of user memory (RAM) known as VDP RAM, which is not directly addressable by the CPU (Central Processing Unit). The Mini Memory is the ONLY module available which allows you access to the VDP ram.

CALL PEEKV and CALL POKEV are used, and samples may be seen in preceeding chapter on advanced programming. They may be used to look at your PROGRAM, or to manipulate the SCREEN DISPLAY.

CALL LOAD and CALL PEEK are used to access the CPU RAM, which comprises of the 4k mini memory, the 32k expansion memory, and the 255 bytes of CPU ram in the console. CALL PEEK can also be used to examine the contents of CPU ROM (READ ONLY MEMORY).

CALL CHARPAT is used to obtain the defining string for a character, which you may then manipulate with SEG$ and redefine with CALL CHAR.

```
e.g.CALL HCHAR(1,1,94,760)
    CALL CHARPAT(94,A$)
    A$=SEG$(A$,1,14)&"FF"
```

CALL LINK permits a TI BASIC program to use a machine code utility or program stored in the Mini Memory with CALL LOAD.

CARE: The mini memory containes a bettery with a stated life of two years, and will retain any data you load into it, even after the console is switched off and the module removed.

However, data is destroyed if you:
Insert or remove the module when the console is switched on.
Use CALL INIT or the INITIALISE option.
Use the module for something else.

Data in the module is also subject to corruption by static electricity, and you should not rely on it as a sole copy of your program or data. Always keep a tape backup.

If you use the module as a data file, the contents can be saved to tape: thus you may store adventure text into the module with a BASIC program, and then copy the data onto tape easily using the 'S' option from the 'Easybug' selection from the main menu. Data is reloaded with the 'L' option.

EDITOR ASSEMBLER:

The editor assembler package comprises a module, two disks and a large manual. One disk contains the 'source code' for one of TI's module games, to help you to understand the language.

Although large, the manual is not suitable for novices, and some information is difficult to find.

The 32k expansion memory, disk drive, disk controller and peripheral expansion box are REQUIRED for this package.

The EDITOR allows you to enter source code, and uses a good screen editor. When you are satisfied, the ASSEMBLER will turn your source code into MACHINE CODE in one of three formats chosen by you: Standard, required if you wish to run the program with the Extended Basic module.
Condensed, which uses less disk space.
Program format: Which uses even less space but but cannot be loaded by a BASIC program. The Editor Assembler and Mini Memory have special commands

Programs you write in assembly language may run with the extended basic, mini memory, or editor assembler modules, but you may need to use different coding for each:

As example, Extended Basic uses different internal memory mapping, and therefore you have to use different memory locations for example to print to the screen.

With Editor Assembler, you may run the disk versions of TI's games modules.

Although providing much greater speed, assembly language is not for everyone.

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) 476-0265*

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

DICK PICHÉ.....TECH.....521-8667

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

Bob can be reached through his sister in North Bay at the number above. If you like, you can also reach him on Delphi (CDU) or on Compuserve (73657,3617). Watch for a more permanent number here.

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 5W2 ***

EDMONTON 99er USER'S GROUP
P.O. BOX 11983
EDMONTON, ALBERTA
T5J 3L1