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

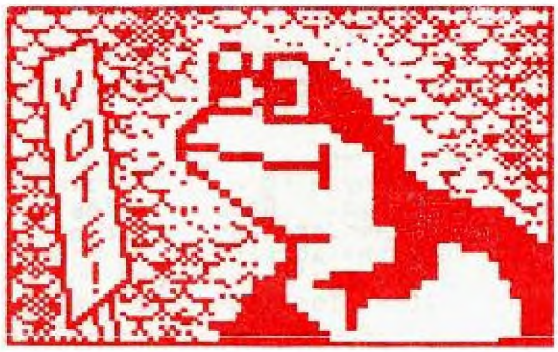
# MAGAZINE

VOLUME 7 NUMBER 08.....OCTOBER 1988

# JOURNAL

## ELECTION NIGHT IN OTTAWA!

OTTAWA (9-15) - As of today, Prime Minister Mulroney has not yet announced the date of the next election. But we do know that the OTTAWA TI-99/4A USERS'GROUP will vote for a new Executive Board at its monthly meeting on Tuesday, October 4th.




DON'T FORGET THE MEETING -- October 4, 1988

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

## COMING EVENTS

| | | |
|---|---|---|
| October Meeting: | October 4, 1988 | Merivale High School<br>Election time again!<br>Don't forget to contact<br>Bill Sponchia if you're<br>interested in running! |
| PRBASE Workshop: | October 15, 1988<br>10:00 a.m. | Merivale High School<br>Contact Bill Sponchia for<br>further information. |
| November Meeting: | November 1, 1988<br>7:30 p.m. | Merivale High School<br>Come on out for the<br>AUCTION! Bring your<br>items to sell and<br>cash to buy! |

Newsletter Deadline: October 15, 1988 (or any time before that!)

## EDITOR'S NOTES
from
Ruth O'Neill

Well, this issue marks number 10 for me as editor (a full year's worth!), and I've enjoyed all of them. I am indebted to the many people who have worked so hard to provide me with cover art, articles (sometimes at the last minute when I've had cancellations), ideas, help in assembling and mailing the newsletters, and support and encouragement along the way. Thank you all. And to anyone interested in helping out in any way, but who may have been too shy to speak up... Please, please, don't hesitate to contact me. One of the most exciting parts of this job is hearing from new people who are interested in writing or who just want to share ideas or information. We're always interested in new writers, and you don't have to write a column to join in. A paragraph's worth, a letter, or a short article on something that interests you could benefit some of our readers.

Enough on that -- on with the news. October, of course, is election month for the Ottawa TI99/4a Users' Group. Bill Sponchia has been contacting everyone by phone to see if they are interested in running. If you haven't heard from him, and you're interested in joining the executive this year, give him a call. If you'd like to reconsider a negative answer, give him a call for that, too. Nominations will also be accepted from the floor, so you have until the meeting night to decide. If you ARE interested, go for it! Putting a little extra into the club is a surefire way to get a lot more out of it.

In November, we are planning an auction. Any member can bring equipment, software, or anything else TI-related to be auctioned off. Be sure to label it properly, though. Reserve bids will be accepted, the terms of sale are cash, and the full amount goes to the owner of the item -- no percentage is being charged for the auction. So come on out! It's a perfect time to get rid of duplicates you don't need, and pick up a little extra cash for Christmas shopping. It's also a perfect time to pick up a something for the TIer's in the family, or even a present to yourself.

The next Saturday workshop will be on PRBASE, and will be held at Merivale High School on October 15. The Multiplan workshop was very well attended, so let's keep up the spirit for this one. I'm sure it will be worthwhile.

On a more personal note, congratulations go out to Paul Berlinguette (a former editor of this newsletter) on his recent promotion. Well done, Paul!

## The President's Two Cents' Worth
by Michael Taylor

As my term as president draws swiftly to a close, I reflect upon the events of the past year, both within our group, and within the larger TI community. As a group, we have lost some very able members to other, "newer", computers; but our membership is still strong.

Good programmers, such as Tom Bentley and Art Green, continue to write for the TI, and new programmers are emerging -- notably, David Caron and Charles Earl. Charles took over from Lloyd Galenzoski, most capably, to massage our BBS programme, Texlink, into its current advanced and many-featured format. Texlink is enjoying a moderate, but deserved, commercial success after receiving favorable reviews in the TI press. Charles' own shareware programme, TELCO, has enjoyed widespread success. No doubt Batch-It, the new programme he and Tom Bentley collaborated on (being released through Asgard Software) will enjoy similar success. David Caron's first major programme was a submission in our recent software contest. It was judged the outstanding entry. David calls it Music-Pro. During the demonstrations of the contest entries, it was such a hit with the members that the executive resolved to look into marketing it commercially, for the benefit of the club. (I need to add that contest submissions are club property.) With the help of Lucie Dorais, David has been whipping Music-Pro into shape for official release at the Chicago fair.

Now, having sung the praises to Charles and David, I would do a disservice not to at least mention Tom Bentley, for his C99 Windows and his PrEditor (via Asgard) packages, and Art Green, for his various utilities (including assemblers and linkers). I would say more about Tom's and Art's work, but since their programmes are primarily tools for programmers, and I am no programmer, I can only pass on that I have heard them praised by people whose opinion I respect.

On the hardware front, it is necessary to look beyond our immediate environs; but here, too, the past year has seen major developments. Most significant, I feel, has been the release, by Myarc, of the Geneve 9640 in production quantities, and, more recently, the Hard and Floppy Disk Controller card.

At this stage, with RAMdisks, hard disks, 80-column cards, and enhanced processor card (read 9640), I venture to say that hardware has outstripped software. True, we have a multitude of programming languages available (Basic, compiled Basic, enhanced Basics, Logo, Forth, Pascal, C, Pilot, Fortran, and Assembly), but applications software worthy of the hardware is scarce, really scarce! The Geneve cries out for an integrated package for home use, combining a word processor, a spelling checker, a data base, a spreadsheet, an outliner, a graphics programme, a telecommunications program, a memo pad, a calendar, a clock, an address book, a label maker, and a To Do list. The key here is "integrated". Such a package can be achieved with less than 400K memory and with a hard disk -- witness Spinnaker Software's Better Working Eight-in-One package (for IBM PC's) selling for $55. Such a package is not a superficial task. For a single programmer, it would be a Herculean endeavour; but could four or five good programmers handle it? Could it be done in six months? Are there enough Geneves out there to make such a project financially viable? Perhaps the ensuing twelve months will tell.

Wrapping up, I want to express my appreciation and thanks to the executive, Jane Laflamme, Bill Sponchia, John O'Connor, and Peter Arpin; to the committee chairs, Ruth O'Neill, Dave Morrison, Stephen Bridgett, Lucie Dorais, Dick Piche, and Henri Monat; to the committee members; to the software contest entrants; to those stalwart supporters (and haranguers) who contributed ideas and opinions at meetings; and to all who helped (with demonstrations, with equipment, with telephoning,...). The club exists through the participation of its members. With your ongoing help, the club will continue in strength and fellowship, to the benefit of TI'ers everywhere.

## BROWSING THE LIBRARY
--with DAVE MORRISON

During the past few weeks I had the opportunity of reading the Library columns of a number of newsletters from other groups in the U.S. and Canada. Needless to say, I voted that my column was, by far, the worst! After saying that (without Editorial disagreement), I can only apologize and hope, as you undoubtedly do, that my writings will improve.

I shall be offering three new disks in October. The first of these is PICASSO by Arto Heino. Whether PICASSO is in the public domain; is Fairware; is Copyright, or is even a plagiarized version of a commercial programme, has been the subject of a good deal of controversy. Without going into details, I can report that the matter appears to have been resolved and that PICASSO Version 1.1 is to be treated as Fairware. (Version 2.0 is a commercial release by Asgard.) This means, of course, that you are expected to send a contribution to Arto in Minto, N.S.W., Australia, if you decide that you like it and find it to be of some use.

Among other features, PICASSO has ten font styles and, in addition to the usual commands, will load any TI-WRITER or TI-ARTIST file for inclusion in your printing.

The second disk is also Fairware and it comes from none other than J. Peter Hoddie. The disk, entitled JPH games, contains six games (two in assembly language) and it will load automatically in Extended Basic. Peter originally wrote these programmes for the commercial market but when Texas Instruments abandoned the home computer market he shelved them. He apparently dusted them off in 1986 and after "a bit of cleaning up", he released this disk as Fairware.

I have not yet decided on the contents of the third disk, but there is bound to be a little bit of everything for everyone!

As usual, the disks that I have offered over the past few months will again be available. So, including the three new disks, you will have a choice of more than twenty disks at our next meeting.

Documentation (where available) will be displayed for your perusal. (n.b. Perusal means read it at the meeting and *NOT* take home!).

Remember, any suggestions for any improvements in the Library (or in the person of the Librarian!) would be appreciated.

## How to Print a Sheet from Multiplan
by Jane Laflamme

Unfortunately, we didn't get to the printing options in the Multiplan workshop held Saturday, September 17th, at Merivale High School. I hope this will help:

Select P from command line (for print, of course!), then O for Options.

o The first item you enter is the "Area:" you wish printed. The easy way of doing this is to press Ctl. Q (top of sheet), then ":" (full colon), then Ctl. Z (for bottom of sheet). Tab over to next field with Ctl. A or Ctl. 2.

o The second field is "Set up:". This is where you type "PIO" or your serial printer's string. Tab over to next field.

o The third option is whether you wish the formulas printed, yes or no. (I very rarely use this option of printing formulas. I believe if the formulas are printed then the data is not... but you'll have to check that for yourself. My answer is "N" for no.)

o Fourth is whether you wish to print row and column numbers, yes or no. Remember, if you do have them printed, it takes a lot more room on your paper and might mean you need more pages. I always say no to this one too. Press enter.

o Next, you have to set up margins. Select M for Margins. The defaults are displayed for you. Most often I like to save paper and will set the left margin to 0 and print width to 80. The right margin is set by the print width. You can set the print "LENGTH" to any amount, as long as you reset the "PAGE" to the same plus the amount that you have stated for the top of the page. eg: If you have selected 10 for the top of the page, and length of printing as 59, then the page length should be 69. You can also set it for smaller pages. The default is set for letter size paper. Needless to say, use the tab function to tab over to the each field.

Once this has all been done, you then can select P for Print. Sounds worse than it actually is, and the good news is that you only have to do this set-up once, then save your file; these items will be saved with the sheet. So from there on, when you load this sheet, it will know how to print it.

**Caution**! If you add rows or columns to the sheet, *CHANGE YOUR PRINTING AREA*!!! (From one user who invariably is surprised when the full sheet is not printed!!! When will I learn?)

Bill Sponchia has set up a great file to set printer codes as MP can't do that for you. (You can, in a pinch, send control codes to your printer through an XB pgm. and then enter MP, but it means you can't turn off your printer!!) With Bill's file, you can bring in the printer codes and install them into a cell anywhere on the spreadsheet, through the Xtern option. It can set your printer to condensed, double width, script, among many other options. It then will be saved with the sheet. This actually, could be a follow-up workshop, or a demo at a meeting. It is a very useful utility that Bill doesn't seem to "toot his own horn" about too much!

If there are any other questions, please don't hesitate to contact me, either by the BBS, phone, or letter.

## Myarc's New Hard and Floppy Disk Controller Card
A Review by Charles Earl

The long-awaited Hard and Floppy Disk Controller Card from Myarc is at last shipping -- I received one just before our last general meeting, and I have been enjoying it thoroughly.

The card will support up to three hard disks, four floppies and a streamer tape for backup. Any MFM ST506/412 compatible hard drive up to 134 meg may be used with the controller. The floppies may be either 40 or 80 track 5 1/4 inch drives or the 720K 3 1/2 inch drives. Apparently the 1.44 meg 3 1/2 inch format is not currently available but it may be available in some future expansion. A real time clock is included in the card, but is not, unfortunately, battery-backed. The package contains the card (HFDCC), documentation and the cables needed to connect one hard disk.

The basic hardware is fully compatible with basic or extended basic, as well as many assembly programs, but some assembly programs such as disk managers and telecommunication programs don't currently support the hard drive. I'm sure you will be seeing a great deal software upgraded to include the hard drive support. The only software that is not 'copy-protected' and does not function well with the hard drive controller card itself is ARCHIVER 3.02 from Barry Boone. Unfortunately, this is an incompatibility with the floppy support as well. I'm sure that the problem will be corrected by either Barry himself or Myarc (since the problem is the result of a bug in the eprom).

The software you receive with the Myarc HFDC card includes MDOS v1.06, GPL v1.01, and MDM 5 v1.21. The MDOS and GPL is for the GENEVE users but MDM 5 will work with both the TI-99/4a and the GENEVE from GPL. A newer version of MDM 5 v1.23 is available on the major networks. MDM 5 supports both floppy and hard disk management. The program will allow TI-99/4a users to set the clock upon booting. The program is device dependant and will only work if a HFDCC is installed. All the basic functions for disk management have been provided as well as the extra support for the hard disk. A convenient feature is "find file", which will search the complete hard disk or floppy for a specific file. The program will also includes a backup utility for backing up some, all or just the new files on the hard disk.

The documentation provided with the package is in a three-ring binder with an addendum for MDOS users. The documentation consists of the basic initialization and operation of the hard disk for the average users as well as the technical information for the programmer. A definite step in the right direction, considering the overly-simplistic documentation TI provided with their Floppy controller. The documentation is fairly sound and will get the average user through the setup/initialization process with little difficulty. Users should try to get some detailed information about the hard drive they purchase. To initialize a hard disk, you must know the number of heads and cylinders as well as whether or not write precompensation and buffered stepping are needed. This information should be supplied with the hard disk when you buy it.

To make the hard disk compatible with older TI software, Myarc has provided three types of disk emulation. The easiest to use is the DSK1 directory. Whenever you type DSK1.filename the hard disk controller card will check to see if the file is in the DSK1 directory. If it is, the program is loaded from or written to the hard disk. If the file is not in the DSK1 directory, control is passed back to floppy. This means you can not save a new file to the DSK1 directory by simply typing SAVE DSK1.PROGRAM. If PROGRAM is not there already, the file will be saved to your floppy -- not your hard disk. If you do wish to save the file to the hard disk, you can use SAVE WDS1.DSK1.PROGRAM.

The second method of emulating a disk on the hard drive is through the DSK directory. This type of emulation is commonly used for programs such as PRBASE, which access the disk through DSK.PRBASE.filename. To simulate this on the hard disk, set up a DSK directory in the root directory of your hard disk. Next, create a sub-directory of the DSK directory called PRBASE and copy the files PRBASE requires into the PRBASE directory.

The third method of emulation will only work if your HFDCC is located at cru address >1100. This method is called DSK1 file emulation, and is an exact duplicate, sector by sector of a diskette. You may have any number of these

emulation files located on your hard disk but only ONE may be active at any time. The only way you can access the data or programs in one of these emulation files is when they are active. When you use an emulation file your physical floppies get bumped up a number; floppy one will become two, two will become three, etc.

The Myarc HFDC card is compatible with other cards such as a Horizon ramdisk but I have noticed that the menu program on the ramdisk version 7.3 does not load programs off the HFDCC properly.

The card does have a few quirks. I know of only two hardware quirks. The first is the problem with ARCHIVER 3.02, which appears to be hardware-oriented and most likely will not be corrected until a new eprom is released. The other is not exactly a quirk, but more an annoyance -- the clock is not battery backed. On the Geneve, this is not a problem since at boot-up the card will set itself using the clock built into the Geneve. It is the TI-99/4a users who must set the thing every time they turn on the computer.

Overall, I am very impressed with the card, on the one condition that Myarc corrects the bug in the eprom which throws ARCHIVER 3.02 for a loop. The documentation is good, but I did find a few sections a bit vague, and there is some information I would have liked to see included, such as some explanation of interlaces. Generally, though, it is clear and much more complete than such documentation usually is.

## FAREWARE - THE FARE MARKET SYSTEM
by Stephen Bridgett

We have all heard the term FAREWARE, and safe to say, we all know what it implies. I'm not here to preach and we know that we (most of us) are all just a little uncomfortable from time to time, knowing that we really should make a contribution to some particular software author. We know their names -- they keep blasting up on our monitors every time we run the software. We know we use and enjoy the program, but, well, you know, we just never quite get around to making out the cheque. The theme keeps playing in the back of our brains, 'I should mail that guy 10 bucks...'.

For those of you actively involved in your club, you know that Ottawa is a leader in the TI community. Members and friends are found throughout the world. We as a club feel a responsibility to ensure that FAREWARE authors are adequately rewarded in accordance with the use that members make of their products.

In 1987 the club compensated Marty Kroll for his library utility of which the club has made continued use. This summer, through a chance conversation on the BBS, a number of Ottawa members decided to make a collection for Barry Boone. Barry is a student who has worked tirelessly on his archiver program and yet has not been properly rewarded. His current version 3.02 represents 2 years of upgrades, and an enormous amount of work. From anyone who has ever downloaded a package from a TI board anywhere, or has ever archived software programs, Barry is owed a lot of thanks.

For anyone wanting to make a donation to Barry, funds are being collected. A letter will be sent to Barry, with the names of the users who have paid for his product.

Contact: Stephen Bridgett through any means you desire, BBS, meetings or telephone (521-3631).

I encourage all members to make some representation to the software authors of their choice. And when you do... tell them you're from Ottawa. You can be proud to be associated with one of the longest running, largest, most successful, respected and talented TI groups that ever existed.

# EXPANDING EXTENDED BASIC'S POWERS -- WRITING ASSEMBLY ROUTINES: PART 1

by David Caron

Many of you have probably bought the extended basic module thinking it was a programmer's paradise, only to realise a few months later that it really was not that much faster than TI-BASIC, certainly slower than other comparable BASIC languages such as Apple BASIC and Commodore 64 BASIC. (I will try hard not to mention them again). Well! With assembly routines you can expect parts of your TI Extended Basic program to run faster than Apple's or Commodore's assembly routines could ever hope to achieve. (This is because the TI-99/4A contains a superior CPU chip which does 16-bit operations, instead of 8-bit operations AND runs THREE times as fast as the Apple II(E) or Commodore 64).
If you are also thinking of learning assembly language, or have already had a go at the Mini-Memory, this series of articles is something that should interest you.

To follow these articles you will need an Editor / Assembler program (Such as Funnelweb) as well as the hardware to run the program and, of course, an Extended Basic module. This article also assumes you know something about assembly programing such as how the CPU memory is mapped and a bit about the actual instructions involved (like MOV for example). If you are not familiar with the material just mentioned, obtain a copy of the famous TI Editor/Assembler manual. For now you must know the difference between decimal and hexadecimal. Take a look at page 412 in the Editor/Assembler manual while you are reading this article or obtain a CPU memory map for the TI.

When you use Extended Basic along with the 32K memory expansion, Extended Basic uses the high memory for Basic programs. The high memory is a 24K block starting at address -24576 decimal or >A000 hexadecimal, and ending at -1 or >FFFF hexadecimal. The memory expansion has 32K memory, yet I have only accounted for 24K of it; what about the remaining 8K?

The remaining 8K is called low memory and starts at address 8192 decimal or >2000 hex, and ends at address 16383 decimal or >3FFF hexadecimal. This is where your very own "Assembly routines" would be placed and executed. Most of that 8K block is for your actual assembly routines, twice as much as the Mini-Memory module, and you get to take advantage of Extended Basic's power.

The Extended Basic commands reserved exclusively for assembly routines are:

```
CALL INIT
CALL LOAD
CALL LINK
CALL PEEK
```

(sorry -- unlike the Mini-Memory and the Editor Assembler module, the makers of Extended Basic forgot about CALL POKEV and CALL PEEKV, but you can always make an assembly routine to make up for this.)

**CALL INIT**

This command causes Extended Basic to load a bunch of utility routines into low memory, if it is there. If low memory is not present, or it is not working, Extended Basic issues a SYNTAX ERROR message. CALL INIT must be successfully executed in order for CALL LOAD and CALL LINK to work.

**CALL LOAD("file name")**

This command will load your uncondensed assembled routine in low memory starting at 9456 or >24FA. If you load another assembled routine afterwards, it will be placed after the first one and so on until the low memory is full. At this point, you will get a MEMORY FULL ERROR message if you try to load yet another assembled routine (also known as object code). Doing a SIZE command will not help you since that will only tell you the free memory in VDP RAM and the high CPU memory. I will tell you later how to find out the remaining free memory in low CPU memory.

**CALL LINK("program name",argument 1, argument 2,...,argument 16)**

This command will execute your assembly routine. Notice that you need to type in the name of your program. This is the name you would type after a DEF instruction in your assembler source (the file containing your program created from the editor). A maximum of six characters are allowed for the program name. When you load your program with CALL LOAD, the program name is placed in the top of low memory 16383 (>3FFF) with the location of the loaded program placed immediately after the name. For example, if you loaded the program called START and it was the first assembler program you loaded after executing a CALL INIT, starting at 16384-8 or 16376, an S would be placed, then a T,A,R,T (ascii numbers) and this would bring us up to address 16381. Since the name was only five characters instead of six, a space character (32) would be poked. You do not have to include the space when you insert the program name in a Call Link command. The 16-bit number 9460 would be poked into the word address 16382. (In reality this would probably not be the number poked at the word address 16382, but for simplicity, assume so, for now). Thus, starting at 16384 the following bytes would be poked: 83, 84, 65, 82, 84, 32, 36, and 244

The first six bytes contain the program name in ASCII format and the last two bytes indicate where the program starts. (36*256 + 244 = 9460).

Arguments are numbers, arrays or strings which you can transfer between Extended Basic and assembly. You can have up to sixteen Arguments. I'll explain more about that in the next article.

**CALL PEEK(address,byte)**

This is the only command which will work without doing a CALL INIT, and therefore will also work without memory expansion. Call Peek does exactly what it appears to do -- you indicate an address in the first parameter, and then place a variable such as A in the second parameter (or argument, same thing). When it is executed, A will be set to the value at the specified address. If you place a variable in the third or fourth parameter, say B and C, B is assigned the value in the next address (AFTER A), and so on. For example if you wanted to know how much free memory existed in low memory, the following CALL PEEK would let you find out:

**CALL PEEK(8194,A,B,C,D)**

The variables A and B indicate the first free address while the addresses C and D indicate the last free address. You can reconstruct the actual 16-bit address using:

first free address = (A * 256)+B
last free address = (C * 256)+C

The number of free bytes can be determined by taking the difference.

**CALL LOAD(address,byte)**

Call load will function as a poke command as well as an object loader. The only difference is that you are poking in data from Extended Basic instead of off the disk. The parameters are identical to those used for Call Peek except that you are now transferring the value of the variable, say A, to low memory instead. If you wanted to clear low memory without doing a Call Init, you could use

**CALL LOAD(8194,36,244,64,0)**

(Notice that the last free address is 16384, which is the first byte PAST the top of low memory. This is just the way Extended Basic keeps track.)

If you loaded an object file from disk after executing the above, that object file would be placed in low memory starting at address 9460 or >24F4 .

And that's how Extended Basic treats assembly files. In the next article I will take a look at writing simple assembly routines which would be executed from Extended Basic. If you are already writing assembly routines for extended basic, and already have questions you would like answers to, do not hesitate to call me at (613) 745-4618.

# FAST EXTENDED BASIC

LUCIE DORAIS

My apologies to all of you who were interested in last month's program, but could not use it because they don't have a printer... This was brought to my attention by Philip Hawtrey at the last meeting, and I remembered having written the CALENDAR program after another one, called PERPETCAL, that gives the calendar on screen, one month at a time.

So, for you, Philip, and all others intrigued by the notion of a "perpetual calendar", here is the screen version, with still a possibility to print, but only one month at a time. It is also based on Mr. Legault's program in "Computing Now!", and some lines are identical to lines in last month's program. Of course, following my recent experience, I went back to it and optimized it to run and print faster, resorting once again to a CAL$ string that contains one whole month.

```
100 REM ********************
110 REM PERPETUAL CALENDAR
120 REM L. Dorais/10-83/9-88
130 REM ********************
140 REM
150 PR$="PIO" :: L$=RPT$("_",28) :: W$="S   M   T   W   T   F   S"
    :: ER$=RPT$(" ",168)
160 MO$="  1   2   3   4   5   6   7   8   9  10  11  12  13  14
      15  16  17  18  19  20  21  22  23  24  25  26  27  28  29
      30  31"
170 DIM MON$(12),LD(12)
180 DATA JANUARY,123,FEBRUARY,112,MARCH,123,APRIL,120,MAY,123,
    JUNE,120,JULY,123
190 DATA AUGUST,123,SEPTEMBER,120,OCTOBER,123,NOVEMBER,120,
    DECEMBER,123
200 CALL CHAR(95,"000000FF",121,"0010385410101010001010101054381000
    000204080FE4020000000004027F020400")
210 CALL CLEAR :: GOTO 220 :: CALL KEY :: CALL GCHAR :: CAL$,DAY$,
    DAY1,K,L,LDAY,L1,L2,L3,L4,L5,L6,M$,MON,PR,S,X,Y,YR :: !@P-
220 DISPLAY AT(4,1):L$:" "&W$
230 DISPLAY AT(10,6):"P E R P E T U A L": :"      C A L E N D A R"
    : : : : :"From 1583 to the end of time": : :L$
240 FOR X=1 TO 12 :: READ MON$(X),LD(X) :: NEXT X
250 DISPLAY AT(22,5):" YEAR (xxxx):": :"    MONTH (1-12):":""
260 ACCEPT AT(22,19)VALIDATE(DIGIT)BEEP:YR :: IF YR<1583 THEN 260
270 ACCEPT AT(24,19)VALIDATE(DIGIT)BEEP:MON :: IF MON<0 OR MON>12
    THEN 270 ELSE 370
280 REM ** calendar **
290 DISPLAY AT(22,5)BEEP:"{ MONTH |   y YEAR z": :
    "(A)nother (T)Print (Q)uit"
300 CALL KEY(1,K,S) :: IF S=0 OR K<0 OR K>18 THEN 300 ELSE K=K+1
310 IF K=2 THEN 250 ELSE IF K=12 THEN 470 ELSE IF K=19 THEN
    CALL CLEAR :: END
320 IF K=6 THEN YR=YR+1 :: GOTO 370
330 IF K=1 THEN YR=YR-1 :: GOTO 370
340 IF K=3 THEN MON=MON-1 :: IF MON=0 THEN MON=12 :: YR=YR-1 ::
    GOTO 370 ELSE 370
350 IF K=4 THEN MON=MON+1 :: IF MON=13 THEN MON=1 :: YR=YR+1 ::
    GOTO 370 ELSE 370
360 GOTO 290
370 DISPLAY AT(2,1):"" :: DISPLAY AT(8,1):ER$:ER$
380 IF YR<1583 THEN YR=1583 :: IF MON=12 THEN MON=1
390 L1=YR-INT(0.6+(1/MON)) :: L2=MON+12*(INT(0.6+(1/MON))) ::
    L3=INT(13*(L2+1)/5) :: L4=INT(5*L1/4) :: L5=INT(L1/100) ::
    L6=INT(L1/400)
400 L=L3+L4+L6-L5 :: DAY1=L-7*INT(L/7)
410 M$=MON$(MON):: LDAY=LD(MON):: IF MON<>2 THEN 430
420 IF INT(YR/4)=YR/4 AND(INT(YR/100)<>YR/100 OR INT(YR/2000)=
    YR/2000)THEN LDAY=116
430 CAL$=SEG$(MO$,1,LDAY) :: FOR X=1 TO DAY1 :: CAL$="    "&CAL$
    :: NEXT X
440 DISPLAY AT(2,INT((24-LEN(M$))/2)):M$;YR
```

```
450 FOR X=1 TO 6 :: DISPLAY AT((2*X)+6,1):SEG$(CAL$,28*X-27,28)
    :: NEXT X
460 GOTO 290
470 REM ** print calendar **
480 DISPLAY AT(22,5):"PRINT THE CALENDAR...": :"" :: IF PR THEN 500
490 DISPLAY AT(24,1):"    "&PR$ :: ACCEPT AT(24,5)SIZE(-28):PR$ ::
    IF PR$="" THEN 290 ELSE PR=99
500 OPEN #2:PR$ :: PRINT #2:TAB((26-LEN(M$))/2);M$;YR ::
    PRINT #2:L$ :: PRINT #2:"  "&W$ :: X=1
510 PRINT #2:SEG$(CAL$,X,28) :: X=X+28 :: IF X<169 THEN 510
520 PRINT #2:L$ :: PRINT #2:"":"" :: CLOSE #2 :: GOTO 290
```

When you run the program, it will first ask you for a year and a month, then it will display that month. You can then press the arrow keys to see the previous or next month, or the same month in the previous or next year. A nice way to teach young children about calendars and leap years! You can also ask for another year and month, or print the screen, or end the program.

We first initialize the variables, define characters (nice little arrows) and data, etc. In this second program, the month name and "last day" position in MO$ are kept in the arrays MON$() and LD(), filled by reading the DATA in line 240. ER$ erases only six lines, since we need a total of twelve to erase part of the screen before each month is displayed. The pre-scan comes rather late in the program (line 210), to add a whole bunch of variables and the data statements to it.

Lines 250-270 ask for your input of year and month, and check for out of range values, the "VALIDATE(DIGIT)" already checking for non-digit characters. A note about lines 290-300, which display the available keys and check for them. You will notice that I use a "T" for "Print", instead of a "P"... This is because the CALL KEY routine is defined for "keyboard # 1", which is the left half of the keyboard, so the "P" is not included in the keyscan. I did it for two reasons: a smaller keyboard is easier to check and, more importantly, the check on the arrow keys works whether you press the FCTN key or not.

To see the new values given to the keys in the split keyboard, look at Appendix I in the XB manual: the keys are renumbered 0 to 19. Now, you can check a "0" in an IF statement, but cannot use it in an ON GOTO statement. Out of habit, I added a "1" to "K" in line 300, but this was not really necessary in this program.

Line 310 acts upon the "ATQ" keys, while lines 320-350 act upon the arrow keys; the year is automatically adjusted if you go from December to January, or vice versa. Line 360 is for those keys that are not used in the program: return to the display, to include the BEEP. All arrow routines, as well as the input routine above, branch out to line 370, which erases part of the screen. Line 380 simply prevents the program from going before January 1583, the cutoff date.

Lines 390-420 then calculate the number of "empty spaces" before the first day of the month, using the same routine as last month's program. Leap years are dealt with in line 420, which is read by Tex only if the month is February; this time, we change the value of "last day" to the position of " 29" in the MO$ string, since we have not built CAL$ yet, which is done in the next line.

The month and year are then displayed at the top of the screen, centered according to the length of the month name; since the year is always four char. long, the right column calculation is based on 24 char., not 28. The calendar itself is displayed by line 450. It takes 6 rows, separated by empty rows. Now, the string CAL$ has a length of 123, plus the starting spaces; we don't need trailing spaces as we did last month, since we don't display or print more than one month at a time. The "len" in SEG$(strg,start,len) will work even if the string is shorter.

To find out the formulas for both rows (8 to 18) and position of each segment of the CAL$ string (1 to 141), I did a list of my starting values, 1 to 6, and a list of the ending ones, then tried to figure out what was common to all:

| X | RW | POS | X | ROW = (2*X)+6 | X | POS in CAL$ = 28*X-27 |
|---|---|---|---|---|---|---|
| 1 | 8 | 1 | 1 | (2*1)+6 = 2+6 = 8 | 1 | 28*1-27 = 28-27 = 1 |
| 2 | 10 | 29 | 2 | (2*2)+6 = 4+6 = 10 | 2 | 28*2-27 = 56-27 = 29 |
| 3 | 12 | 57 | 3 | (2*3)+6 = 6+6 = 12 | 3 | 28*3-27 = 84-27 = 57 |
| 4 | 14 | 85 | 4 | (2*4)+6 = 8+6 = 14 | 4 | 28*4-27 = 112-27 = 85 |
| 5 | 16 | 113 | 5 | (2*5)+6 = 10+6 = 16 | 5 | 28*5-27 = 140-27 = 113 |
| 6 | 18 | 141 | 6 | (2*6)+6 = 12+6 = 18 | 6 | 28*6-27 = 168-27 = 141 |

This is not the only way to do it -- my first version was more complicated, finding the correct row from the six wanted positions in CAL$; even if there was only one calculation, it took slightly longer and the line was much more difficult to understand:

```
450 FOR X=1 TO 141 STEP 28 :: R=((X/1)/28+4)*2 :: DISPLAY AT(R,1):
    SEG$(CAL$,X,28) :: NEXT X
```

I did keep the idea of segmenting CAL$ into 28 char. sections to print the calendar on paper, but decided against a FOR-NEXT loop, to print faster (see line 510 -- the value "169" is 28x6+1, i.e. a position for a seventh week, which we will never need). The variable PR is a flag set to tell Tex not to ask for the printer name more than once. I could have skipped the printer name routine altogether, just OPENing PR$ as defined in line 150, but I sometimes forget to turn the printer ON...

The centering of the month and year is based this time on 26 char., because of the "margin" we add to the W$ string in line 500. When the month has been committed to paper, we return to the "menu" at the bottom of the screen, ready to press a new key.

## **TI BASIC** continued from September
by **Steven Shaw**

SIZE returns the amount of free memory.

EXTENDED BASIC uses some of the system RAM, and you do not have quite as much memory available for your programs. In addition, the cassette loader cannot handle programs over 12k.

The good news is that with Extended Basic you may access the memory expansion unit, which permits you to load (from DISK) a program up to 24k, and still have some 14k available for variables and so on.

The new function key REDO will repeat your last entry, and if the last entry was a program line (either just entered, or recalled using FCTN X) the line reappears on the screen with the cursor at the beginning of the line NUMBER, allowing you to change the line number if you wish. This function is useful if your program contains a lot of lines either the same or with only small differences. That was a brief summary only of the attractions of EXTENDED BASIC. It does require a little more care in use, but gives you considerably more programming power.

An added attraction of the module is that it permits you to load and run Assembly language programs, provided you have the extra peripherals required.

Example: In the USA, TI release TI INVADERS on disk for half the price of the module. You require a disk system and the 32k memory expansion.

A utility package is sold in the USA on tape which requires the 32k RAM and permits fast pseudo-high resolution graphics, or the rapid dumping of the screen to a printer (RS232 card required for the latter) or to disk(disk system required).

Lockouts have been found to occur in the present Extended Basic by using CALL PEEK at one particular section of memory (the addresses vary from console to console), and by using a number of print separators without spaces:

```
PRINT :::::::
```

(Correct in TI BASIC, but Extended Basic requires a space between each colon).

The following program has been included to show how SPRITES are used in EXTENDED BASIC.

The program was developed in a highly experimental manner, as various routines and values were tried.

To obtain the best from SPRITES it is usually necessary to work in this manner.

```
100 REM SPEEDRACE
110 REM A SAMPLE PROGRAM IN
120 REM TI EXTENDED BASIC
130 REM USING SPRITES
140 REM
150 REM ====================
160 REM
170 CALL CLEAR
180 PRINT "SPEEDRACE":"COPYRIGHT 1981":"BY STEPHEN SHAW"
190 PRINT "USE S & D TO MOVE ":"LEFT & RIGHT":" ":"USE KEYS 1,
    2,3,&4 TO":"SELECT GEAR"
200 PRINT "DISTANCE & TIME ARE ":"DISPLAYED.":"DISTANCE
    SUFFERS IF YOU":"CRASH"
210 PRINT "PRESS ANY KEY TO CONTINUE"
220 CALL KEY(3,V,M)
230 IF M<1 THEN 220
240 CALL SCREEN(2)
250 FOR X=1 TO 100 :: NEXT X
260 CALL CLEAR
270 CALL MAGNIFY(3)
280 M=1
290 X$=RPT$("0",40)
300 CALL CHAR(100,"96FEBA3838BAFEBA"&X$)
310 CALL CHAR(108,"5A5A5A5A5A5A5A5A5A5A"&X$)
320 CALL CHAR(104,"FF1111FF0000FF11FF"&X$)
330 CALL SCREEN(4)
340 CALL SPRITE(#6,108,13,80,9,90,0)
350 CALL SPRITE(#7,104,13,75,25,90,0)
360 CALL SPRITE(#8,104,13,70,38,90,0)
370 CALL SPRITE(#9,108,13,65,9,90,0)
380 CALL SPRITE(#10,104,13,60,25,90,0)
390 CALL SPRITE(#11,104,13,55,38,90,0)
400 CALL SPRITE(#12,104,13,50,9,90,0)
410 CALL SPRITE(#13,104,13,45,25,90,0)
420 CALL SPRITE(#14,104,13,40,38,90,0)
430 CALL SPRITE(#15,104,13,85,139,90,0)
440 CALL SPRITE(#16,104,13,80,150,90,0)
450 CALL SPRITE(#17,108,13,75,170,90,0)
460 CALL SPRITE(#18,104,13,70,139,90,0)
470 CALL SPRITE(#19,104,13,65,150,90,0)
480 CALL SPRITE(#20,104,13,60,170,90,0)
490 CALL SPRITE(#21,104,13,55,139,90,0)
500 CALL SPRITE(#22,104,13,50,150,90,0)
510 CALL COLOR(8,3,4)
520 CALL SPRITE(#23,108,13,45,170,90,0)
530 CALL VCHAR(1,8,140,216)
540 CALL COLOR(14,12,12)
550     CALL     VCHAR(1,7 95,24)::      CALL      VCHAR(1,17,95,24)::     CALL
CHAR(95,"5555555555555555")
560 FOR CT=1 TO 4
```

```
570 CALL SPRITE(#CT,100,CT+6,CT*47-45,93-CT*8,0,0)
580 NEXT CT
590 CALL SPRITE(#5,100,16,160,74,0,0)
600 REM **
610 REM   ***
620 CALL SOUND(-1000,-2,30-7*SPEED)
630 CALL COINC(ALL,D):: IF D<0 THEN GOSUB 780
640 CALL KEY(0,A,B):: IF A=ASC("S")THEN CALL MOTION(#5,0,-10)
650 IF A=ASC("D")THEN CALL MOTION(#5,0,10)
660 IF A<30 THEN CALL MOTION(#5,0,0)
670 CALL COINC(ALL,D):: IF D<0 THEN GOSUB 780
680 IF A>48 AND A<53 THEN SPEED=(A-48)/3
690 CALL COINC(ALL,D):: IF D<0 THEN GOTO 760
700 T=T+1 :: S=S+6*SPEED :: DISPLAY AT(10,18)SIZE(10):STR$
    (S)&" "&STR$(T)
710 CALL COINC(ALL,D):: IF D<0 THEN GOTO 760
720 IF T/5=INT(T/5)THEN M=-M
730 CALL MOTION(#1,SPEED*40,M*5,#2,SPEED*40,M*5,#3,
    SPEED*40,M*5,#4, SPEED*40,M*5)
740 CALL COINC(ALL,D):: IF D<0 THEN GOSUB 780
750 GOTO 620
760 GOSUB 780
770 GOTO 620
780 CALL SOUND(-900,-6,0)
790 CALL MOTION(#1,0,0,#2,0,0,#3,0,0,#4,0,0,#5,0,0)
800 SPEED=1/3
810 S=S-50
820 IF S<0 THEN S=0
830 CRASH=CRASH+1
840 IF CRASH=15 OR T>200 THEN GOTO 920
850 M=+1
860 T=T-(5*(T/5-INT(T/5)))
870 FOR CT=1 TO 4
880 CALL SPRITE(#CT,100,CT+6,CT*47-45,93-CT*8,0,0)
890 NEXT CT
900 SPEED=0
910 RETURN
920 CALL CLEAR
930 PRINT "YOU HAVE TRAVELLED   ":"A DISTANCE OF ";S
940 PRINT "AND HAD ";CRASH;" CRASHES!"
950 IF S>500 THEN PRINT "YOU ARE NOT A BAD DRIVER"
960 IF S<100 THEN PRINT "YOU SHOULD NOT BE ON THE":"ROAD"
970 PRINT "TO TRY AGAIN,ENTER 'RUN'"
980 END
```

Note especially the number of CALL COINCs which have been used. Even with this number, some crashes will still go undetected.

# 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.

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

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

PETER ARPIN.....TREASURER AND SYSOP.....523-0017

JOHN O'CONNOR.....SECRETARY.....833-2626

BILL SPONCHIA.....PAST PRESIDENT, SOFTWARE CONTEST.....523-0878

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.

FROM

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

EDMONTON 99er USER'S GROUP