

99'er Online

March 1985

P.O. Box 11983
Edmonton, Alberta
Canada T5J 3L1

TO: (

99'er ON LINE is the news letter of the Edmonton 99'er Computer User's Society published ten times a year. All material contained in this news letter may be published in other news letters provided that source and author are identified unless otherwise stated. We welcome correspondence from all TI User Groups and will extend source credit courtesy in 99'er ON LINE.

CORRESPONDENCE: News letter editor: BOB PASS, 59 LABELLE CR, ST. ALBERT, ALBERTA, CANADA T8N-2G6. All other correspondence: EDMONTON 99'er COMPUTER USER'S SOCIETY, P/O BOX 11983, EDMONTON, ALBERTA, CANADA T5J-3L1

OFFICERS: PRESIDENT--GILL CANNON. VICE PRES--PAUL HELWIG, TREASURER--EVAN SMITH, SECRETARY--SUSAN LIVINGSTON

DISCLAIMER: All information published in this news letter is, for the most part, the fruits of the labors of amateurs; therefore, we cannot guarantee that the information presented is always correct.

REGULAR MEETINGS: Regular meetings of the Edmonton User's Group are held on the second Tuesday of each month on the 3rd floor of the General Services building of the University of Alberta from 7:00 till 10:00 PM and are open to all members in good standing. Non-members may attend their first meeting free of charge. The Executive Committee meets monthly. Members may attend these meetings as observers or to address a particular issue. Arrange with one of the officers listed above if you wish to attend.

ADVERTISING: Commercial advertising space is available in this news letter at the following rates: FULL PAGE--\$20.00, HALF PAGE--\$15.00, 1/4 PAGE--\$10.00. Discuss your commercial needs with Paul Helwig at the next meeting or write to the P/O Box above. Members may advertise their personal computer related items for free but are asked to limit their ads to about 20 words. Mail your ads to the EDITOR'S ADDRESS or hand it to him at the general meeting; newsletter deadline 15th of the month.

MEMBERSHIP FEES: FAMILY--12 MONTHS, \$20.00, 6 MONTHS, \$15.00. STUDENTS--12MONTHS, \$15.00, 6 MONTHS, \$10.00.

BOB'S LATE-WINTER NOTES

by: Bob Chapman

It's getting close to that time of year when the evenings are getting longer and time to review the programming accomplishments of the winter...

No-one owned up earlier on to having a program to calculate the payment schedule on the "Canadian Mortgage", where the interest is calculated semi-annually, not in advance. The U of A put on an evening course titled "Solving the Mysteries of Home Mortgage Financing". I went on this course so I've been able to make up the program myself. It is in console basic and it prints to the screen. It is self explanatory so I won't bother telling you how to run it.

I have developed an X-Basic version that will print to the "PIO". The printout formats nicely with the "PRINT USING" "IMAGE" statements. (Modem owners call me up for a copy).

Compare the results from this program with a printout from your mortgage company! Some mortgage companies calculate the interest on a daily basis: I do have a program to do this, but it needs a bit more work as leap years have to be considered along with 29, 30, 31 day months!

I mentioned recently that the indication lamp on my CORCOMP 32k Memory Card had packed up, and that I was writing to CORCOMP regarding repairing it. I think CORCOMP may have sent out the wrong reply to me, as I've just received a pile of CORCOMP drawings: Disk Controller Card, RS 232 Card (2 Versions), and 32k Memory Expansion. For their Memory Expansion System, there is also a drawing called: MES -1 & -2. Anyone wanting to see these drawings, please see me. I think I'll have to send another note to CORCOMP about my lamp.

THIS LISTING IS IN THE SAME
FORMAT AS IT WILL APPEAR ON
YOUR SCREEN UPON ENTRY.
THIS IS AN ASTERISK (*),
THIS IS A ZERO (0), AND THIS
IS A LETTER O.
THE FOLLOWING LINE IS A ROW
OF ALTERNATING DASHES AND
SPACES.

10 REM C1985 EDMONTON TI
USERS GROUP

20 REM CANNMORT BASIC

30 CALL CLEAR

40 PRINT " MORTGAGE CALCUL
ATIONS":

50 PRINT " AMORTIZATION PE
RIOD IS USUALLY 25 YRS,
THUS AN. PERIOD INITIAL
LY MAY BE 25*12, (300
MONTHS)":

60 PRINT "TERM IS USUALLY 12
-60 MONTHS":

70 PRINT " FOR COMPLETE SCH
EDULE, MAKE TERM EQUAL
TO AM. PER
IOD":

80 INPUT " INTEREST RATE ?
":IN

90 INPUT " PRINCIPAL?
":PR

100 INPUT " AN. PERIOD(MNTHS)
? ":M

110 INPUT " TERM (MNTHS)?
":TERM

120 J=((1+IN/200) (1/6))-1

130 AM=J/(1-(1/(1+J) M))

140 ORIG=PR

150 PMT=AM*PR

160 PMT=(INT(PMT+.5))/10
0

170 PRINT "
PUSH ANY KEY TO
PAUSE"

180 REM ***START RUN***

190 REM -----

200 FOR I=1 TO TERM

210 INP=J*PR

220 INP=(INT(INP+.5))/10
0

230 IF I<M THEN 250

240 PMT=INP+PR

250 PRIN=PMT-INP

260 PRI=PR PRIN

270 TINP=TINP+INP

280 TPRIN=TPRIN+PRIN

290 TPMT=TPMT+PMT

300 PRINT

310 PRINT "PAYMENT NO. ";I

320 PRINT "PAYMENT \$";PMT

330 PRINT "INTEREST \$";INP

340 PRINT "PRINCIPAL \$";PRIN

350 PRINT "NEW BAL. \$";PRI

360 PR=PRI

370 CALL KEY(0,K,S)

380 IF S<>0 THEN 370

390 NEXT I

400 PRINT
 410 PRINT "SUMMARY"
 420 PRINT "-----"
 430 PRINT "ORIGINAL PRIN. \$"
 ;ORIG
 440 PRINT "TOTAL PAYMENTS \$"
 ;TPMT
 450 PRINT "TOTAL INTEREST \$"
 ;TINP
 460 PRINT "TOT PRIN. PAID \$"
 ;TPRIN
 470 END

SPRITE ONE LINER

by: Barron Bartlett
 from: HUG Newsletter, June/83

Want to amaze and frustrate your Atari, Vic-20, Color Computer friends? Then type in the following in X-Basic in the command mode:

```
CALL CLEAR :: CALL SCREEN(5) :: CALL MAGNIFY(2) :: FOR
I=1 TO 28 :: CALL SPRITE(#I,64+I,16,80,80,3&I,8) :: NEXT
I :: FOR J=1 TO 5000 :: NEXT J
```

Press ENTER and watch all 28 sprites do their thing. To see it again, press FCTN REDO then ENTER. Repeat till your pals turn green with envy! Try editing the program (after pressing FCTN REDO) to change colors, characters, speeds, etc. You can even add a line number to make a one line program. This is a good way to begin learning about sprites.

MACHINE LANGUAGE

by: Unknown
 from: DATA STREAM, October/84

The age of the super-intelligent computer is finally upon us. Sees there was a rich Texan from Lubbock who found himself in an electronics exposition. Spotting a talking computer, he decided to check it out for himself. Wondering what the computer would reply, he boldly declared that he was a Texan and that he owned 10,000 acres back home.

To this, the computer almost instantly replied, "That's a lot of dirt Tex".

Not to be put down, the Texan told the machine that he ran 10,000 head of cattle on his range.

"Now that's a lot of bull!", replied the computer nonchalantly.

Being cold shouldered by a box of transistors was not our hero's strong point! Puffing himself up, Stetson pushed well back, he proudly proclaimed that he was rich enough to not only buy the damned computer but that he would likely have ten million (U.S.) dollars left over after he did.

The computer suggested his.

NOVA COMPUTERWARE
 52 Airport Rd. 452-8372
 TI/80 4A Home computer prod.
 CLOSED DURING MARCH
 RE-OPENING APRIL 2



broadmoor stationers ltd.

- Office Supplies
- Office Furniture
- Typewriter Rentals
- Typewriter Repairs

- Business Machines
 - Typewriters
 - Calculators
 - Cash Registers
- Photo Copy Service
- Rubber Stamps

SALES, SERVICE & LEASING

broadmoor stationers

165 Athabasca Avenue Sherwood Park

Mon. - Fri. 9:00 - 5:30 pm Sat. 10:00 - 4 p.m.

SHARP

464-4343

SCM SMITH-CORONA

DATA BASE 99, A SOFTWARE REVIEW

by: Tom Hall

Most reviews I have written in the past have been generally to praise a particular piece of software. Unfortunately, this review will not fall into that category. The DATA BASE 99 package from Quality Software is a significant improvement over the TI MAILING LIST package, and that's about the best thing I can say about it. Getting nicked for \$35 U.S., I expected more than just another EXTENDED BASIC file management system; when you disregard the sophisticated protection scheme employed to make the disk uncopyable and uncatalogable, that's all there really is.

The package does have some nice features, such as allowing the user to define his own input screen. Basically, when you create a file using this package, you are presented with a blank screen, and you can set it up in any way you like, as long as there are no more than 28 fields and no more than a total of 245 characters per record. You can name the fields in any way you like, and the "CTRL_" key places underscore characters where the actual data will be added later.

Once the input screen is created and everything is okay, it is written to disk, and from that point on, whenever you want to add data to the file, you will be shown an input screen exactly like the one you entered, except that the cursor will skip field names and blanks, and will only allow you to write in areas where you earlier put the special underscore character. Each record is written to disk as soon as you verify that the information is correct.

Since the file is a relative-access type file, sorting is done by placing in each record a numeric variable which "points" to the next record in the sorted list. This provides some flexibility for later editing, because when you delete a record it is not actually removed from the file; it's link pointer is simply set to zero. This means that once a record is written to the file, it will never actually be removed, unless you create a subfile and move only certain records to that subfile. An option called REPAIR FILE allows you to sort a file and restore every record that has previously been deleted. All this does is to return deleted records to the file by changing their pointers from the current value of zero to whatever their rightful place in the file would be.

There are some rather fundamental bugs in the package which I think, considering the price of the software, should have been found and removed prior to marketing it. First of all, there is a routine that allows you to catalog a disk. The program prompts you for the number of the drive in an ordinary BASIC INPUT statement; if your entry is invalid (or null), the program simply crashes right there on the spot. As well, the format used for the cataloging shows you all the information about the disk except the protection status of the files; there's no reason at all that I can see why this important information regarding the contents of a disk should have been excluded.

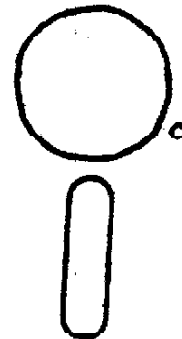
Another annoying bug in the package is in the CHANGE RECORD subroutine. After being taken through the procedure for selecting which record you want to modify, the record is brought to the screen, and the cursor is placed in the first position of the first field in the record. At this point you will discover that there is no way to back the cursor up in a line; once it has passed a position, there's no going back, and you have

to go through the entire procedure again to correct any mistakes you might have made the last time around. Even FCTM S and FCTM D don't work; they are treated as characters, and even though they are invisible on the screen, you probably won't like what they look like on paper! It only took a little checking to discover the reason for this problem: everywhere in the program where CALL KEY is used, the key unit referenced is "5". For some inexplicable reason, just prior to entering the CHANGE RECORD subroutine, there is a CALL KEY(4,K,S). As you may recall, key unit 4 remaps the keyboard in such a way that even the cursor control keys are accepted as valid characters: a blatantly stupid mistake.

One redeeming virtue of the package is that the documentation includes a detailed description of the file layout. Files created by DATA BASE 99 are RELATIVE, INTERNAL, INPUT (or OUTPUT), FIXED 255 format. The last section of the documentation gives specific information on file organization, and this information makes it quite simple to write your own BASIC program to access files created by this package.

I'm hoping that one of these days I'll get to examine a data base package that runs entirely in machine language. But, until that time, I'll grudgingly give DATA BASE 99 a 6 out of 10, since if you disregard the bugs and awkwardness of the package in places, it's actually a fairly decent product.

SS/DD DISKETTES
10 FOR \$16.00
SEE BOB PASS



NEWSROOM

PRINTER ACCESS - FORTH STYLE

NEXT MEETING - The next meeting will be held on the second Tuesday in March (ie Mar 12, 1985) at 7:00 PM in room 849 of the General Services building on the U of A campus. This is building number 16 on the campus map and is located on the east side of 116 St about 2 blocks north of the Jubilee Auditorium.

LAST MEETING - As mentioned in the last news letter, February's meeting included a general business session to call for a vote on the charging of initiation fees. The motion was presented as follows:

Moved by Paul Helwig, seconded by Tom Hall -

That the Edmonton 99'er Computer Users' Society charge a one time \$20.00 initiation fee to all current and future members of the Society. This fee is to be used for the purchase of capital equipment (hardware, software, & educational material) for the use of Society members.

The motion was voted on by a show of hands of members present. Results are: For 27, Against 0, Abstain 1. Motion carried.

Subsequent to this vote, several members immediately paid their initiation fee of 20 bucks to get our capital fund off to a great start with \$300 in the sock. Hopefully, this will be enough for you to see results by the next meeting.

Please bring in your initiation fee as soon as possible as we need \$800 as soon as possible to take advantage of an excellent offer.

FLOPPIES - Bob Pass has obtained the floppy disks ordered at the last meeting and will bring them along to the March meeting. For those who ordered, please remember to bring along your \$16.00. At this price for ten disks, the "Flippie/Floppie" dispute should be settled; it's hardly worth the bother to double-cut your disks now! If anyone else would like to order, please let Bob know at the next meeting; he would appreciate cash up front where possible.

CONTRIBUTING EDITOR - Giving your newsletter editor (me) a hand now is Greg Sears who is also looking after the P/O box and organizing the corresponding newsletter file from other 99'ers world wide. Greg is sorting through the newsletters and other sources and relaying the cream to me for inclusion in this publication. Greg's contributions will be found all over the place as "Little Seas", Jokes, etc. My thanks to Greg and to all my other contributors for making my job easier and your newsletter one of the best.

*To Evr is Human ; But
to Really Messup, you
Need a Computer!*

by: Michal Jaegermann

This is for all of you with TI Forth and a printer. But even if you do not have your printer yet, you will find something interesting. So read on!

Quite frankly - an inspiration for this note comes from Tom Hall's article in the last newsletter in which he showed how to use BASIC (X-BASIC in fact, but this only makes for a nicer program) in order to send control codes for setting up a printer. He is right. Thumbing through a manual every time you want to use some feature of your printer is a drag. I would like to show how to solve the same problem in Forth. When you see the elegance and flexibility of the solution you will understand why recently I am programming in BASIC only when I have to.

Before we start, let's see how Forth can print something at all. Well, in TI Forth quite a piece of the job was already done. On screens 72 and 73 of the freely distributed disk, a number of printer support words are already defined. As probably everybody knows there is small typo there. In line 5 of screen 72 the word PAB_ADDR should read PAB-ADDR. Also, if your printer is not on the first port of the RS232 with a baud rate of 9600, you will have to edit that screen a little bit. For example - my Gemini is connected to a parallel port. The only change required is in line 4 after a word F- (this is short for File Description). I have the PIO*. Could be as well PIO.CR* if you are printing graphics, or something else if your printer has a different description. Actually any valid device name like RS232* (for telecommunications) or USK1.JUNK* for printing to a file will do, since our computer considers all external devices as files. Only in the last case do you have to be a little bit careful about setting file attributes. You will get an I/O error message if they are of the wrong kind. We will handle this issue some other time.

I also made some changes for aesthetic reasons and convenience. The file switching word, defined in line 2 by FILE, is called >RS232; I have changed it to >PIO. This is immaterial and could be instead known as JOE or BILL as long as the name is unique and the same name is used in line 3 in a colon definition of SWCH (switch). The form will be of some importance when you use any files at the same time. So if you changed the name in line 2 adjust line 3 accordingly. The last small change I have done is in line 4. At the very end of that line sits one 3 which actually belongs to the line below. If you should edit it out during printer experiments your system may crash. So I removed it from line 4 and my line 5 now reads:

PAB-ADDR a 3 OVER VSBW 1 OVER 5 + VSBW ALTOUT ! ;

Now you may 72 LOAD (do not forget to "FLUSH" the above changes to your system disk). Simple SWCH 72 LIST CR UNSWCH will produce listing of your edited screen 72 on your printer. (CR is here to cause printing out of the last printer buffer and may be superfluous if your printer operates in a different way.) Small variations on screen 72 will enable you to send your latest Forth achievement to your friend - via modem and phone line. Actually SWCH will cause an output of any screen printing word to be send to a chosen device.

I added to my source one more convenience feature. To

avoid constant typing SWCH ... UNSWCH - sometimes in blind - I defined a word PID as follows

```
: PID SWCH [COMPILE] ' CFA EXECUTE CR UNSWCH ;
```

Now to list screen # 72 I can type 72 PID LIST. 8000 20 PID JUMP will cause a printout of the contents of 20 bytes starting at the address 8000. And so on. 72 TRIAD does not require PID. Try it!

After this long introduction - when we know how to make the printer cooperate with Forth - let's return to our original problem, how to send control codes to your printer. The simplest solution which comes to mind is as follows: define a word (or words) which will do the following: SWCH type string of characters UNSWCH. We have in Forth the word for typing strings - namely ".". The only problem is that putting Ctrl-A in a type string is - expressing it mildly - inconvenient, and for Ctrl-A this is impossible. Try it yourself. And for printer control we have to generate a lot of codes of that kind. To be sure there are some indirect ways to doctor type strings, but such a solution would be really kludgy and not at all "high level". There is a much better way and it is called defining words. This is an extremely powerful feature of Forth which is absolutely unique for that language, so let's have a closer look.

As you know Forth is extensible. To some extent this is also true for a number of other languages. On this common level this means that you may add new words to the language understood by the system. A new word is compiled on the top of the existing dictionary. Most of the time this is done in "colon definition" by a subroutine known as a colon compiler. But there are also other compiling words - like, for example, VARIABLE. A uniqueness of Forth stems from the fact that you may also add your own compiling words creating new specialized mini-compilers tailored to your needs. The most important thing to remember about those words is that they are "double-action" words and you have to specify both actions. In fig-Forth, hence also in TI Forth, one is doing that with the pair <BUILDS

```
: (name of "mini-compiler")
  <BUILDS recipe for compilation of a new word
  DOES> what to do with the contents of
  the compiled word in a run-time ;
```

DOES> by itself specifies a run-time action. It returns, namely, an address of the code of the compiled word. It is possible to stop in this moment. This is actually a run-time action of VARIABLE which is defined as : VARIABLE <BUILDS DOES> ;. But let's have another example, say something like this:

```
: PUTIN <BUILDS DUP C, 0 DD C, LOOP HERE =CELLS DP !
DOES> COUNT TYPE ;
```

This compiling word requires on stack some numbers in the range 0-255 followed by a count. Lets try it first.

```
84 55 67          3 PUTIN DOG
69 83 95 79 77   5 PUTIN CAT
69 83 69 69 72 67 6 PUTIN MOUSE
and something more crazy
53 8              2 PUTIN 2+2=
```

Now execute DOG CAT MOUSE 2+2= and have fun. What happened? A "mini-compiler", or - officially - defining the word, PUTIN when compiling duplicated count and compiled it on the top of the dictionary moving up HERE by 1. This was done by C, . Next count was used for a loop which repeated a compiling action with subsequent numbers from the stack. A blurb HERE =CELLS DP! is necessary on our machine which uses a 16-bit processor. Consequently 2 insists on an even address. If you omit this nothing wrong will happen initially. But if you

leave an odd value of HERE and try to compile the next word you will find yourself in big trouble. FIND will not be able to find anything and the only response you will get from the system will be " ? " .

Now run-time. DOES> returns the address of the count byte, COUNT fetches it and moves the address up by 1 and TYPE happily types 'count' bytes interpreting them as ASCII characters. Why do we need that? Forth already has a nice word "." which is more or less performing that job. Well, one thing is that PUTIN does not necessarily have to only TYPE in run-time; it may also run around and wave flags. Also, please note that in 2+2= I have put in a backspace as easily as any other character. And this is something which we need for a printer control. So we are nearly done? It is enough to extend run-time action adding redirection of output to a printer and that is that? Yes and no. If you look at the examples, you will see that I have put ASCII codes on the stack in reverse order so that they were compiled properly for output. But I am too lazy to remember to type those printer codes in reverse order when I want to create a new printer control word. They are surly enough by themselves. So they have to be compiled or outputted in a reversed order. Since you are compiling only once and running the result any times, keep it as simple as possible. To be sure, there are a number of ways to reverse an order in a compiled word and I am sure that after a short while you will possibly find a better method. My solution is presented below. The listing is using a "\ " which means 'treat the remainder of the line as a comment'. You may add it to your system, if you wish, defining as it follows: \ IN 240 + FFC0 AND IN ! ; (constants in HEX).

```
: PRINTING <BUILDS \ codes count --
  HERE \ leave on stack low loop limit
  SWAP DUP \ put on the top two copies of
count
  C, \ compile count
  ALLOT \ move dictionary pointer up
  HERE 1- \ leave on stack upper loop limit
  DO I C! -1 +LOOP \ store ASCII codes from top down
  HERE =CELLS DP ! \ ensure even dictionary pointer
  DOES> \ run time - leave address of codes
  SWCH COUNT TYPE UNSWCH ;
  \ send your codes to printer.
```

Now is the time to get your printer manual. Let see - to print italics you have to send to the printer ASCII 27 and 52. (These are Gemini codes. Lots of printers will use the same, but check your manual). So we are defining 27 52 2 PRINTING ITALICS. Now when your printer is on and printer words are loaded, it is enough to type ITALICS and your printer is printing italics. To get back to standard, you type STANDARD and presto - you have standard. Of course standard was defined as 27 55 2 PRINTING STANDARD. Other examples are:

```
27 66 1 3 PRINTING PICA
27 66 2 3 PRINTING ELITE
27 66 3 3 PRINTING CONDENSED
27 69 2 PRINTING EMPHASIZED
27 45 1 3 PRINTING UNDERLINE
27 68 4 14 28 35 57 0 8 PRINTING MY-TABS
```

- - and so on. You could rewrite the whole printer manual. Often used printer switches may be added to your printer screens. Actually on screen 73 there is enough space left to add comfortably all of your extensions. More exotic switches may be created for a one time use with the printing job on hand.

Well, explanations were lengthy since this is the first time we were presenting here the defining of words. But say, do you still want to program in BASIC?

DISK MANAGER IIIA SOFTWARE REVIEW

by: Tom Hall

I'm sure that just about all of us can relate to the following scenario: You've just finished doing some extensive modifications to a BASIC program on which you've been working off-and-on for quite some time. After completing the latest round of revisions, you save the updated version to disk — only to discover that you forgot to unprotect the file on disk. And, to make matters worse, there isn't even room enough on the disk to save the program under another name. So, as you frantically hunt around for another disk to save your evening's work on, you utter to yourself about how nice it would be if you could unprotect a file without leaving BASIC.

Well, our prayers have been answered: from Quality Software comes a neat little utility called DISK MANAGER III. The only hardware requirements are EXTENDED BASIC and 32K Expansion Memory. It autoloads in EXTENDED BASIC and, once loaded, will always be available until either CALL INIT is executed, some absolute assembly code is loaded into low expansion memory, or until you turn off the expansion system.

Once the DISK MANAGER III has been loaded, it is accessed through CALL LINK statements. All the major features of the DISK MANAGER module are included, except for the copy functions. Additionally, if you have an EXTENDED BASIC program in memory after loading DISK MANAGER III, that program will not be affected by MANAGER operations. With DISK MANAGER III you can catalog a disk to screen or printer, protect and unprotect files, rename files and/or disks, and initialize a blank disk in any format compatible with your hardware. File deletions are handled in the usual BASIC manner, i.e., DELETE "DSK2.FILENAME".

One of the most powerful features of DISK MANAGER III is that all of the commands it provides can be invoked from an EXTENDED BASIC program. For instance, it is now possible to write a 2 or 3 line program to protect (or unprotect) every file on a disk, or to check for the occurrence of a certain string in file names and change that to another string. The authors give one word of caution: because of the large amount of memory involved in initializing a disk, it is not recommended that the CALL LINK("INIT") command be used from a BASIC program, as there is every possibility — especially in a larger program — that part of your program's symbol and/or data tables may be wiped out in the process.

In my opinion this is an indispensable utility for anyone who works a lot with BASIC.

THE GAMES GANG**New****Low****Prices****Full****repairs****on T.I.****equipment**OUT OF WARRANTY REPAIRS**FAMILY COMPUTERS**

435-4636

3872-63rd AVENUE, EDMONTON, ALBERTA T6E 0B6