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

MAINSTAY

VOLUME 6 NUMBER 03.....MARCH 1987

TI-FEST

SATURDAY
MAY 16, 1987

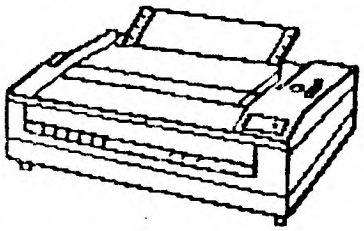
A FESTIVAL CELEBRATING
THE TI-99/4A

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

Need a second Disk Drive? We have Panosonics on at a special price of:

$169.00

Looking For A Printer?



The Siekosha SP-1000 A is a Near Letter Quality printer that can't be beat. It sports a draft mode at 100 cps and a NLQ mode of 22 cps. With both tractor and friction feed the price can't ba beat at

$369.00

Bulk DS/DD Diskettes

.79 EACH

DD-100L Diskette Box

$14.95

HEAD OFFICE
105 O Connor St
Ottawa Ontario
K1P 5M8
(613) 232 5203

1600 Merivale Rd.
Nepean Ontario
K2J 3K3
(613) 727-0180

EDITORS NOTES

by **Marg. O'Connor**

We recently received a disk directly from the author of **TI-FORTH** by **Lutz Winkler** 1540 Corsica St. SanDiego, CA, USA, 92111 ,to replace one that had passed through at least one or two changes. There were some differences from the Sessions already printed in our OTIUG Newsletter. EDITOR, OTIUG.

The differences are:

Session 1 - printed in November 86

The ditty near the bottom of the page should read

```
: SEE 252 22 DO I DUP . 7 VWTR
    KEY 2 = IF ABORT ENDIF LOOP ;
```

Session 2 - printed in December 86

No differences found.

Session 3 - printed in January 87

The last paragraph was not in our original copy so it is reproduced here:

You should be learning how FORTH works. I recommend you obtain Brodie's STARTING FORTH and also programming forth BY Chirlian. These tutorials will continue, but are not intended as a substitute for a text book. It's time to start learning and for that read Chapters 1 and 2 of STARTING FORTH. I went through the book first and annotated each page with the corresponding remarks from the manual's Appendix C (Notes on starting FORTH). This helped to lessen my initial confusion (at least somewhat).

Session 4 - printed in February 87

The first paragraph in our Newsletter version is not in the Authors disk version but we presume it is still valid.
The indexing system in our session 4 is completely different from the authors disk article and his example reads as follows:

```
SCR 28
( TEXT MODE SCREEN-DUMP          -PRINT )

HEX
: SCREEN-DUMP  SWCH
               03C0 00 DO I DOP 28 MOD 0=
                          IF CR THEN VSBR EMIT
                    LOOP CR UNSWCH ;
DECIMAL
```

SESSION 5 - printed this month

As provided on the Authors disk.

I hope this is not too confusing to those of you that are trying to learn Fourth. Also included in this issue are some of the Logos submitted.

CHAIRMAN'S TWO CENTS WORTH

by Berry Minuk

The Geneve 9640 has still not appeared in Ottawa but I have heard rumours that Myarc has begun shipping the new machine to some dealers. I hope that we may have some further information from either Jane or Bob by the time of the next general meeting. Unfortunately, it appears that the IBM compatible system may not be readily available in Canada since the latest rumours are that it does not have CSA approval. For the latest up to date info consult our BBS system.

Speaking of our BBS you will have noticed if you call it that Benoit's Board as modified by Lloyd is now back up and that both downloads and uploads are available. Try it, you'll like it.

In previous newsletters I mentioned that the workshops had begun in Assembly under Art Green. Now I can let you know that Bill Sponchia has started an Assembly workshop for beginners. Further details can be obtained by contacting Bill, but he has let us know that they are on the 2nd and 4th Wednesdays of the month. The C workshop is continueing on the 2nd Wednesday of the month and has almost finished its first project which was the writing of a BBS in C. As previously announced the BBS is working but still needs some fine tuning.

The executive has decided as an experiment to put in the newsletter some of the items of business that will come up at the next meeting. These are as follows:
1) Software Contest;
2) Discussion about Honourary Life memberships;
May we please have your reaction to this idea. What do you think about it?

The Executive has also scheduled the topics for the next few meetings and these are:
April - A Hardware tutorial including cleaning, etc.;
May - A discussion with demos of most popular fairware proorams;
The programs for the May meeting will be picked by you based on your responses to a list of Fairware programs which will be available at the March and/or April meetings.

This brings me once again to a problem which seems to me to be totally unnecessary. I am referring to the difficulty Dick Piche has been having in getting volunteers to bring their equipment to the monthly meetings. At each meeting Dick tries to get names of people who will bring all or part of their systems. However, the results are disappointing.

Remember that we still have a cassette library and tapes are available by contacting Jack McAllister.

I will see you all at the April meeting and let's have a super turnout for it. Remember - April 7.

PS: The software catalog has been updated and Steve will have disk copies available at the April meeting. Don't miss it!!

BROWSING THE LIBRARY

--with STEPHEN BRIDGETT

Interest in the Disk Of The Month has been on the rise and the spin off is new revenue for the club, as well as new software in the hands of our loyal users.

Before I go further, I'll mention that the disk for March was inadvertently (by me) distributed in double sided format. For all those single sided people who purchased the disk, I will offer a 2 for 1 trade at the April meeting, as an act of good faith in the members. It is my intention to keep the library at the lowest

common denominator of the users, that being single sided software, however members should know that the library as it stood as of the time I received it, is double sided. It will be an on going project to both document and reformat the library.

If anyone has had problems with the picture disks of Feb., and I'll say that this is really interesting stuff, remember not to add '_P' to the called file name. These pictures are really something and worth seeing.

A special note to all those people who offered help with the library .... I have your names fondly listed in my notes and am looking forward to speaking with all of you. Up until this point, I would have to say that there has been an element of crisis management in getting the library organized. When I get time to utilize all the offers of help, things will run much smoother. As of this moment there is an overwhelming amount of business to attend to, with little free time to spare. But, by the time of the fest, I will be approaching people for assistance. I thank you for your support and patience.

Many people picked up the cataloger program this month. Next month the disk will be the actual catalogue. I'm excited about the prospect of the users having complete access to the catalogue and software. This was my aim when I took over the library. Remember too that the catalogue will be available not only as a disk purchase, but also from the BBS as a download. I recommend that all interested members acquire the DISK OF THE MONTH for March in order to take advantage of the software catalogue coming soon. In keeping with the single sided format, the catalogue will start out on two single sided disks. I'll attempt to have sufficient copies to cover the demand.

NEW DEVELOPMENT...ATTENTION CASSETTE BASED USERS...

installing within the console itself, 32K of memory.!! The cost is minimal, just $50.00 complete !! This will give console-cassette based users access to a greater variety of software, notably some excellent ASSEMBLY LANGUAGE programs. Contact Ralph at work at 596-7410. Many have already made the switch and are happy. It's a whole new world for cassete based owners.

Thats it for this month, see you in April.

SOFTWARE.....STEPHEN...521-3631

CASSETTE BASED SOFTWARE...JACK...225-6989

DOUBLE FOR NOTHING

by **Lou Burrelli** D.D.O. Quebec

Mr. Bobbitt's article in the September Issue of MICROpendium compares GRAPHX with TI ARTIST. His conclusions prove that neither program can be said to be overall superior to the other. Depending on your particular needs, one can be preferred to the other. He says. "No clear winner; each has its own strengths". Let me add that, if you have both programs, however, the winner is YOU! With the Conversions program of TI ARTIST 2.0, one can have the best of both worlds.

GRAPHX has two ways of working with files:
a) Pictures and b)Clipart.

TI ARTIST has three ways of working with files:
a) Pictures, b) Instances and c) Fonts.

Using the GRAPHX program, you can scatter many Clipart figures over the whole screen and save it as a Picture. Similary, using the TI ARTIST program, you can do the same for both the Instances and the Fonts.

The end results will be:
i) your whole Clipart collection saved as various GRAPHX Pictures (I suggest separate Pictures for a) animated figures and b) GRAPHX fonts:

ii) your whole Instances collection saved as various TI ARTIST Pictures:

ii) your whole Fonts collection saved as various TI ARTIST Pictures.

Now the Magic begins. With the help of the Conversions program of TI ARTIST 2.0 (i.e. option 4 of the main menu), load the GRAPHX Pictures and save them as TI ARTIST Pictures and vice versa.

The end results now will be:
i) your whole Clipart collection saved as various TI ARTIST Pictures:
ii) your whole Instances collection saved as various GRAPHX Pictures:
iii) your whole Fonts collection saved as GRAPHX Pictures.

The converted Pictures can now be worked on in their respective Master programs. The scattered figures in the Picture can now be turned into a Clipart, Instance, or Font collection.

If you also have Draw-A-Bit, Draw-A-Bit II, or Draw'n Plot, you can have a field day in building up your library for each program. Have fun!!

FORTH TO YOU, TOO! SESSION 5

When we set up our autobooting system disk I stated that I always include -DUMP. This utility provides a lot more than what I am going to cover here since I want to keep things as simple and understandable as possible. As you may have gathered by now, Forth is a 'stack-oriented' language. There are several 'stacks' but whenever there is reference made to 'the stack' it means the PARAMETER stack. (Any other stack is specified.) The stack's main purpose is for temporary storage of parameters, i.e., numbers. Every time you make an entry from the keyboard (or a word is encountered in the program) Forth checks to see if it is in the dictionary. If it is not found, it goes onto the stack as a number (parameter), otherwise the word is executed. By now you know the stack concept (last thing goes on top and is the first thing taken off). Because there is a limit to your computer's memory capacity the number of parameters which can be stored on the stack is limited. Good programmers let only those get on the stack which are needed. If by chance you define words which leave 'junk' on the stack it will eventually reach its limit and the program will stop with a ?FULL STACK message on your display. Conversely, if you don't leave required parameters on it, you'll get ?EMPTY STACK.

Bring on .S (dot-S). It let's you look at the stack content without touching it otherwise, i.e., it neither adds nor removes anything. For example, let's enter 15 and then 22. Now enter .S and see what you get on the display. It should show ! 15 22. The ! symbolizes the bottom of the stack. In other words, if we use . (dot) then 22 should be printed to the display because it is the top (first-out) item on the stack. Now use .S and see what's left. Only the 15. Another . (dot) will fetch it and if you use . once more Forth will respond with ? EMPTY STACK. (Usually preceeded by a number.) In order to program in Forth you must understand the whys and hows of the stack.

Speaking of stacks, there is another one, though it is never called a stack. It goes by the name of DICTIONARY, but just like nearly everything in Forth it is also a stack. Every time you define a new word it ends up on top of the dictionary (stack). On the bottom reside - you guessed it - the Forth resident words. Our autoboot then piles the words from the load options on top and finally you add your words (or your program's words). Large programs can use up almost all of the memory. Say you have loaded the AAA SUPER-DUPER XY CALCULATOR and there are now 1,500 bytes free. You are through calculating and wish to install the PARAGON XY PLOTTER. You may not be aware of the fact that it takes 4,000 bytes, so as it boots there comes the point where your TI has reached its limit. ?DICTIONARY FULL will be the message to let you know that there is no way you can run the XY PLOTTER with SUPER-DUPER CALCULATOR still in memory. Well, there's always COLD to start over. Not necessary. FORGET is easier and faster. FORGET cccc (as it is stated in the manual) wipes everything out of memory starting with cccc and every word which was added after it.

One way to always know what to FORGET to get rid of a program, but not the autoboot, is to include on the WELCOME screen a do-nothing word. It should be added as the last word just before the R->BASE word. It can be anything you like, most people use their initials to help them remember to FORGET. (How is that for logic?) Every word compiled prior to my : LW ; remains in the dictionary, every word added afterward is dropped by FORGET LW ENTER>. In the case of very short routines which I may load on top of another one I usually include a : XX ; or similar do-nothing and display a prompt upon exiting to remind me what to forget. In this fashion I leave the underlying program in memory.

RECAP:

The .S word displays the parameter stack's content without adding or removing anything. "!" denotes the bottom of the stack.

FORGET cccc let's you clear from the dictionary entries beginning with cccc and every word added since cccc was compiled.

Placing a do-nothing word on screen 3 makes forgetting easy.

SUGGESTION: Study Chapter 7 of STARTING FORTH.

TIMP Tips and Techniques

by

Steve Zimmerman

This time around, I'm going to talk (or write!) about a variety of small and not -so-small tips to help you avoid trouble with your spreadsheet models. First, if you choose to lock the formulas in a worksheet make an unlocked backup BEFORE YOU LOCK THEM! Locking the formulas in a worksheet is easy-unlocking them requires you to do it piece-by-piece! So, just in case you need to make changes in the future, keep your unlocked backup available! If someone else will be doing data entry on a complex worksheet, it IS a good idea to have them working with a locked copy- this avoids problems such as having someone enter a number or label in a cell which contains formulas or information you use elsewhere in the worksheet.

When building a worksheet, work on one area at a time. This allows you to enter numbers to check to see that each small area of the worksheet does what you want it to do. Using this technique, you can build up each individual area so that it works, and then link the areas to produce subtotals, grand totals, and the like. This is rather like programming in FORTH -in that you define and test a word, and then use that word in further definitions once you have tested it.

Deleting your numbers can help you see if an error condition results, and you can then correct the cause of the error condition in setting up your formulas.

Use relative references wherever possible in building a worksheet (I seldom use absolute references for anything!). The greatest advantage of relative references is that they allow you to easily copy formulas which will be used over and over. An absolute reference cannot be copied and used in another area of the worksheet without being edited-a time consuming process, and one which is prone to errors!

I use relative references for adding up columns of numbers--say, to get a daily total. Adding up R[-1]C+R[-3]C+R[-5]C adds the number 1 row up, 3 rows up, and 5 rows up, in THIS column (and so, can be copied to any column and do the same thing!), and places the total here. A formula of R13C2+R11C2+R9C2, will add the values in those 3 cells and place the total in whatever cell it is in. If copied from column 2 to column 3, it will STILL add up the numbers IN COLUMN 2--NOT IN COLUMN 3! To get it to add up the numbers in column 3, 4, 5, or wherever it is copied to, it must be edited to correct the column references after it is copied. This is, to put it mildly, a pain!

The most accurate way to enter formulas in a spreadsheet is by pointing. Some spreadsheets don't allow this, but we are fortunate because mulTIplan does. In fact, in mulTIplan, we can point not only in entering but also in editing formulas. This is sometimes referred to as "wander mode". Using this method of building a formula is simple--and also has the advantage of creating relative references.

To use this method, begin by placing the cursor or cell pointer in the cell in which you want the formula to be placed and the end result to appear. Key either V or = to begin entering a formula. Use the fctn> E, S, D, or X (the arrow keys) to move the cursor to the first cell you want to add, subtract or whatever. As you do, you will see a relative reference appear on the command line after the word VALUE:--if you move the cell pointer up 1 cell, it will show VALUE: R[-1]C. If this is the first cell you want to use, press the key for the operator (+,-,*,/) you need (in this case, let's say +). The + will appear after the cell reference on the command line and the cell pointer will drop back to the cell you

are building the formula in. To select the cell you want to add to the one just above, move the cell pointer again using the arrow keys. As you move it, you will see the relative reference on the command line change. When you have found the next cell, again, enter the next operator, and continue in this manner until the formula is complete. When you have pointed to the last cell to be in the formula, press enter>. The cell pointer will drop back to the cell you are working in, and the number created by your formula will appear in that cell. In the lower left-hand corner of your screen, you will see the coordinates of the cell you are in, and the first 16 positions of the formula you have created (or the entire formula, if it is less than 16 positions long).

Using this method, you can easily enter long formulas without trying to remember cell coordinates and without taking the time (and memory overhead!) involved in naming cells. Formulas that operate in one column or row can be copied across or down and will work properly in any row or column.

When building a model this way, enter your row or column labels first, then enter data (sample data, if possible), and last, point to each data item to create your formula. Once you have the formula set up, blank the cells with your sample data items, let the sheet recalculate (or tell it to recalculate, if you turn off the automatic recalc as I always do), and check for errors, such as #DIV/0! (meaning that you're trying to divide by 0). Save your backup copy when your formulas are done, then lock the formulas in your working copy (if you want to), and enter your data. This will help you in creating error-free models.

Next month, we'll go over some (potential) problems that can lead to your having different numbers appear in your models than you are expecting! Until then, have fun with mulTIplan!

CHAPTER TWO by Steven Shaw

TI BASIC

In this section we will look at the language in your console. There are a number of general books now available on the BASIC language, and one or two of these may help you if you experience difficulty in handling the language. Many evening classes are also available.

For greater assistance this section will follow as closely as possible the order of the TI Manual.

Do not try to take in all the information in one reading, but go back and read it again a few times.

A computer works as a large number of switches, which are either on or off. Each 'switch' is described as a BIT. In order to pass information more quickly, the computer looks at more than one BIT at a time. The TI99/4A uses a 16 bit processor: it is able to look at 16 bits at a time. For most purposes however, the computer looks at 'words' composed of 8 bits. These words are called BYTES.

A BYTE is a binary number composed of eight numbers, which may be 0 or 1. In digital representation the BYTE has a maximum value of 255 (Binary 11111111).

The computer stores its commands (reserved words) as one byte, rather than a collection of letters. It can only identify the command words if you follow the rules regarding the characters permitted in front of and following command words. In general, you may only use a space, arithmetic operator, or ENTER, but there are exceptions which you will see in the program listings in the manual and in the books of TI programs now available.

For discussion of the error tracing commands (Trace, Untrace,Break) see the section on How to use TI Basic.

LIST

The command LIST is used to list the contents of a program. Used on its own,it will list the program on the screen. You may use CLEAR (FCTN 4) to halt the LIST. To start again at sav line 400, you type in LIST 400- the hyphen indicating 'to the end', or LIST 400-600

LIST can also be used in many other ways. These are described in the section on modules and peripherals.

SAVE

Please refer to the section on cassette handling.

LET is optional. but uses up one byte of memory every time you use it. It is better not to.

```
LET A=2  is the same as
A=2
```

END is also optional. but in this case it is good practice to use it. By adding END to your program. you may be certain when you list it in the future, that you have the complete program. and not a 'working copy'.

IF...THEN...ELSE

TI BASIC may appear to be slightly limited in its use of IF...THEN compared to some other computers. TI do however allow the ELSE alternative.

The problem arises because TI insist that you use the construction only to transfer to another line. You cannot add commands such as:

```
IF X=B THEN B=C
```

to do this you need Extended Basic.

However, TI BASIC does have 'relational operators' which will often help you out of this problem. These may be found described in the section on Advanced Programming.

FOR TO STEP

Note that in TI Basic you must always use the variable name after NEXT . NEXT on its own is in error. In some early computers you were not allowed to transfer to another line once a FOR NEXT loop had been established, but with the TI99/4A you need not worry. You may leave a FOR NEXT loop before the loop has been completed.

Sample use:

```
100 FOR FREQ=110 TO 200
110 CALL SOUND(100,FREQ,0)
120 NEXT FREQ
```

For..to..step may also be used to provide delays:

```
100 FOR DELAY=1 TO 300
110 NEXT DELAY
```

will take a little over a second to complete in TI Basic.

INPUT

Try to use a separate INPUT for each variable. It IS possible to input more than one variable eg by using INPUT A,B but this requires the program user to input two numbers separated by a comma.
The TI form of input. INPUT "HOW MANY?";N uses a colon separator (:), most other Basics use a semi colon (;).

VARIABLES

When you wish to refer to a number, you may use that number, or a 'label' representing the number. For instance, if we tell the computer:

```
A=2
```

Then whenever the computer comes to 'A' (without other letters, that is, with spaces or brackets on either side), it will treat it as the number 2.

'A' is a VARIABLE, and can be allocated to any number. The TI99/4A may have variable names up to fifteen letters long: you may for instance use:

```
HIGHSCORE=12000
```

A variable representing a number is a NUMERIC VARIABLE and a variable representing a letter,a word, or a group of words is called a STRING VARIABLE. A string variable always ends with the dollar sign:

```
MESSAGE$="YOU WIN"
```

Strings (as they are called) are dealt with later.

READ...DATA...RESTORE

TI Basic is slow at reading DATA lines, and if you need to use a number of READs, it is essential that you do not do it more often than absolutely necessary. It is a good idea to fill a variable array, and refer to that.(ARRAYs are dealt with at some length later in this chapter).

```
eg FOR I=1 TO 5        READ A
   IF A=1 THEN 200
   NEXT I
   DATA 2,3,1,0,6

if used often, could be replaced with:
   FOR I=1 to 5
   READ B(I)
   NEXT I
then when a check is required
   FOR I=1 TO 5
   IF B(I)=1 THEN 200
   NEXT I
   DATA 2,3,1,0,6
```

It is worth mentioning that DATA causes more problems in debugging a program than any other command. There must be enough DATA to fill all the READs in the program, and they must be numbers if a numeric variable is READ.

Be careful how many commas you use in your DATA lines : too many or too few can cause many hours searching for errors. The error messages you will receive may be some distance from a READ line, if you have loaded an incorrect value into a numeric variable due to missing out just one comma.

PRINT

TI Basic has a fairly slow screen scroll,but your information will appear more quickly if you use the print separators instead of a number of separate PRINT lines. You will also save memory.

```
eg 100 PRINT "PRESS"
   110 PRINT "1.  TO START"
   120 PRINT "2.  TO TERMINATE"
   130 PRINT
   140 PRINT "H FOR HELP"
```

Will appear more quickly if you use:

```
100 PRINT "PRESS":"1.  TO START":"2.  TO TERMINATE"::"H FOR HELP"
```

TI Basic allows you to key in a program line up to 4 screen lines long, so use this facility. Notice that instead of a single PRINT to scroll one line, an extra colon has been used in our single line amendment. Each colon causes the screen to scroll once.

MYARC EXTENDED BASIC Vn 2.11

A REVIEW BY STEVEN SHAW - ENGLAND

This new version of Extended Basic is a far more than a TI-compatible : it is largely compatible with TI ExBas but has a number of enhancements which make it a worthy successor, a definIte must for anyone who programs in Basic - and it could even win converts from other languages.
One major drawback: none of it is in GPL. In other words, it is not possible to fit even the TI-compatible bits into a module. With the extensions... therefore you MUST have at least a 128k ram card to run it. At present it will run with the Myarc and Foundation cards only, and a new EPROM for the cards is required - included in the price for a Myarc card and at very low cost for a Foundation card (Foundation have now ceased trading by the way).
So here is what you need in order to use Myarc ExBas: at least 128k ram, preferably 512k, and at least one disk drive, with disk operating system.

TI BASIC:

Myarc Extended Basic allows you to run 99% of programs written in TI BASIC. You may need to use CALL FILES(1) to load them from disk - but a clever bit of sidestepping allows the Myarc XB to load a longer TIB file than TI's XB can.

TI EXTENDED BASIC:

One major problem was found with running commercial programs: some commercial producers not only supply programs in PROTECTED format, but the program also checks the CPU RAM flag to see if protection has been removed, and if it has, it locks out the console (there are a range of CALL PEEKS that can do this).
Bad news: in an effort to increase the protection of PROTECTION, Myarc have moved the flag - so these programs take a look, fail to find the marker, and crash!
You will note the version number above! In the last issue I reported on Vn 2.0, and then received Vn 2.1, just as the last issue of Rambles went to press! I duly reported the problems with Vn 2.1 to MYARC, and in due course - very quickly actually - received Vn 2.11, in which most of the problem areas had been fixed. Note by the way that Myarc have included these two upgrades as part of the original price, something Texas Instruments NEVER did in the U.K.
The prescan check of for-next loops does not function correctly- and this can make debugging very interesting! If a NEXT is missing, the error message:
FOR-NEXT NESTING IN NNNN is given. Unfortunately the number NNNN has little resemblance to any line number in your program! Remember to check that every FOR has its NEXT yourself if you have problems! For existing TI XB programs this change to pre-scan makes no difference.
ACCEPT AT and DISPLAY AT also differ from TI XB, in an undocumented manner which could actually be helpful!
For instance:
ACCEPT AT(11,12)SIZE(32):A$ does what?
In TI XB it blanks from column 11 to the end of line 12, and accepts A$ from column 11 to column 28.
In Myarc XB however, it blanks from column 11, line 12, to column 11 of line 13, and the ACCEPT cursor appears on LINE 13!!!
This can cause problems! Fortunately, you are unlikely to come across this problem in many TI XB programs!
DISPLAY AT(11,12)SIZE(32):A$ makes the same deletion, spreading over two lines, but at least the display starts in the right place! Again you are unlikely to have problems with this.
Interesting... ACCEPT AT(24,12)SIZE(32):A$ will place the cursor on ROW 1!!
OK ... now to the useful bit. Lets be really odd and try:
ACCEPT AT(12,2)SIZE(255):A$
NOW... Beginning at row 12, column 2, count 255 screen characters... including the edge characters. THEN the cursor appears! BUT those 255 screen characters may not be blanks! They seem to be taken from the input buffer but I could be wrong. The result is not useful!
Now...ACCEPT AT(12,2)SIZE(-255):A$. Nothing is blanked, but the input field for A$ measures 255 chars from (12,2), including any positions outside current margins (eg edge characters). You can therefore type in quite a lot of text, and have it accepts with one key push into one string. This could be very useful. You can of

course use HCHAR first to blank out that section of screen.
As this is undocumented it may not stay through to any subsequent versions, but I hope it does, as it answers a question asked often in Micropendium other TI-related mags and newsletters...
And indeed, in the 2.11 variation, Myarc HAVE kept in the above, at my request. The ability to load a string with 255 characters with one press of the ENTER key is not to be dropped lightly!
Here is my comment on Vn 2.1:
BLACK MARK: I may be the only person to OPEN files with the APPEND declaration, but Myarc XB fails to support it. It will open the file without an error message, but then refuses to write to it! Bye Bye APPEND.
Myarc claimed that Vn 2.11 had been fixed- unfortunately, only to the point of not issuing an error message when you try to write to the file- so you may think everything is going well - it isn't. Do not use APPEND!
The manual I have bears the following claim: "VASTLY IMPROVED ERROR HANDLING SUPPORT" - so of course I have had a close look at this area, and find the claim entirely false (sorry). In most cases, the same error messages are given as in TI XB, but in several instances, quite different messages appear (eg SYNTAX ERROR instead of BAD VALUE). In one case, an abbreviated error message is reported - eg only SYNTAX ERROR instead of SYNTAX ERROR : RECURSIVE SUBPROGRAM CALL.
In checking error messages out I came across an undocumented TI XB error message- Myarc have also not documented it but worse, have omitted the error trap entirely. Try this one:

```
100 DEF T(N)=T(N)-1
110 PRINT T(N)
```

Short program isn't it! In TI XB I received the message: UDF REFS ITSELF (almost an X-cert message eh!)
BUT in Myarc XB, these two lines crash the system!!!
And also an undocumented error message in Myarc XB, where a standard error message should be:
INVALID ERROR NUMBER IN NNNN
Thats a good one - this was produced by VERSION 2.1 with the following code:

```
100 GOTO 120
110 SUB TRAP
120 PRINT "ERROR MESSAGE="
130 SUBEND
```

However, version 2.11 produced:

```
WARNING
NUMERIC OVERFLOW IN 130
WARNING
NUMERIC OVERFLOW IN 130
WARNING
NUMERIC OVERFLOW IN 515
WARNING
NUMERIC OVERFLOW IN 515
```

Yes, FOUR warning messages - and yes, I know there is no line 515 in the code! After throwing these messages at you, you keyboard becomes unresponsive- only the QUIT key will function.
You CAN stop the WARNING messages with an ON WARNING NEXT, but the console will still become unresponsive. The problem area seems to be the console being busy looking for somewhere to go to when it hits the SUBEND!
And in checking out this area of errors, I found something very interesting - not just a difference between the two XBs, but an unreported feature of TI XB. Read the manual on SUB PROGRAMS and it tells you that SUBEND can ONLY be followed by: Another subprogram, REM and END.
Try this:

```
100 CALL HEH
110 SUB HEH
120 SUBEND
130 PRINT "PROGRAM END"
```

Does it work?
No it doesn't, you get an error message in TI XB. whereas Myarc XB prints "READY".
SO TRY THIS ONE in TI XB:

```
100 CALL HUM
110 SUB HUM
120 SUBEND
130 !P-
140 PRINT "PROGRAM END"
```

and now TI XB works just like the Myarc XB. The word SUB appears

to act in the same manner as STOP, and forces the READY message.
Now try this one - in Myarc XB the !P- is not required:

```
100 PRINT "START"
110 CALL MIDDLE
120 PRINT "RETURNED FROM CALL"
130 GOTO 180
140 SUB MIDDLE
150 PRINT "IN CALL"
160 SUBEND
170 !P-
180 PRINT "THIS IS AFTER SUBEND"
```

NOW... I am not advocating that you should place your SUB PROGRAMS in the MIDDLE of your programs - let's accept they are better at the end - BUT - they really do not HAVE to be at the end!!!
That covers normal TI XB, now onto the enhancements. The Myarc manual referred to is the one I have : it may be subject to revision!
The manual suggests that:
PROTECTED programs will auto-run and erase if CLEARed or at termination. WRONG. They act just like TI XB progrs!
DEF can be used with more than one parameter - wrong, it works exactly like TI XB.
RUN and OLD will load a program LISTed in DV80 format- wrong.[But Myarc say their new computer the GENEVE can do this]. DEFINT can be used in subprograms: wrong. it cannot- you CAN use the format suggested but the results are not those desired. Dont try it! [Myarc responded to this with the comment "?" - but it really does NOT work!
Original review of Vn 2.1: DEFINT can be used for numeric arrays- wrong. Most unfortunate as this is the area that using INT can really save memory! Myarc's reply on this one was to tell me how to do it! as the manual failed to do so and my guesses were not right!
The way to DIM an array for INTEGER variables only is:

```
100 DEFINT DIM A(20)
```

The original manual suggests a command to force a garbage collection, which can be useful to prevent irritating problems with music and sprites when you least want them, but the command has not been implemented.- Myarc tell me they ran out memory space!
Now we move onto the GOOD bits of the ExBas side of things... and some of these are a lot better than some the ads mention...
First you have commands strangely similar to UNIX commands... PWD and CHDIR ... used in COMMAND mode. these:
CHDIR: sets the name of a default i/o device. If you do not use the command. the default device is DSK1. To use it you type in: CHDIR RD. or similar.
PWD: prints the current default device name.
What use are they? Suppose you are working on a program called PROGRAM, which is on a disk in drive 1 and the default device is DSK1. To save PROGRAM to DSK1 you only type: SAVE PROGRAM and to reload it you only type OLD PROGRAM The default device name is inserted between OLD, RUN, and SAVE and the file name! Neat. RUN when used in a PROGRAM uses a QUOTED string, and you can now have in your program:

```
100 INPUT "PROGRAM NAME?":A$
110 RUN "DSK1."&A$
```

or similar - something TI wouldn't let us do.
Also, in your program you can use a line like this one: RUN "DSK2.PARTTWO",CONTINUE Know what this does? It retains all the variable values from program one for use in program two - including string variables!!! Think about it...
In COMMAND mode, RUN seems to use an unquoted string- quotation marks are not obligatory. They are also optional with OLD and SAVE.
Thus RUN DSK1.LOAD and RUN "DSK1.LOAD" are equivalent.
PRINT and DELETE still require quotation marks.
You can also use RUN or OLD to load and run a machine code program which is on disk in memory image format. This may not always work though - see the section on Machine code below!
GRAPHICS are a real delight, and screen displays are set up SO QUICKLY! You can use the 32 column screen, the 40 column screen, and bit map mode, all very easy to use with simple EX BAS CALL commands.
The 32 and 40 column screens use identical commands, except for colour: in 40 column mode, text colour is set by using two parameters with CALL SCREEN!
In BIT MAP mode you have the extra CALLS:
CIRCLE, RECTANGLE, DRAW, DRAWTO, FILL, POINT, WRITE, DCOLOR

where WRITE is used to place text onto the bit map screen, using the 8x8 pixel character size. CALL HCHAR and CALL VCHAR are also available in bit map mode.
Another undocumented feature you will like- remember TI's canard that you could not have sprite automotion in bit map mode? This is repeated in the Myarc XB manual I have - but in truth, you CAN switch Myarc XB to bit map mode, and then call sprites with auto-motion. Neat.
In the 32 and 40 column text modes, you have the ability to "window" by setting margins for screen top, bottom, left and right. I recommend you do not LIST a program in a window two characters wide....! The margins can be changed several times, and HCHAR and VCHAR let you place characters outside the margins anyway..
In default, the margins are set for two unused columns on either side of the screen, allowing 28 and 36 printable columns respectively. Unlike standard TI XB, PRINT does NOT scroll characters in the margins (eg outside the window).
The reason you can run TI Basic programs without worrying about character sets 15 and 16 is that Myarc XB actually gives you 256 definable characters in 32 character sets - and 32 sprites for good measure. That is enough for most people! (In bit map mode Chars 216 to 255 are not available).
CALL CHARSET restores characters 32 to 95 only, maintaining compatability, but CALL GRAPHICS(N) restores all predefined characters, as well as restoring default colours and clearing the screen....
VARIABLES can now use lowercase, so that you can use A,a,b,B=2 and have FOUR variables set. CALL color is accepted and remains unchanged when you LIST but has exactly the same effect as CALL COLOR.
If you use Myarc XB to write a program and use lower case variables, the program WILL run in TI XB - but if you edit a line with a lower case variable, TI XB will re-crunch the line and change the lower case to upper case!
Here is one Myarc have kept quiet about.... MERGE format is a remarkably USEFUL thing that TI gave to us. BUT it was so slow... a long program could easily take an hour or so to load! MYARC have performed a miracle, and MERGE now saves and loads almost in a twinkling (especially if you use a RAM disk!). I was astonished at the speed up. Amazed even! Thanks Myarc!
INTEGERS... so often we hear that using integer math makes a CONSIDERABLE difference in speed. Maybe, but not what Myarc has done! You have the ability to define simple variables to be integers. If you do this, any decimal values are ROUNDED not decimal-stripped. The increase in speed? In a simple benchmark I clocked a 20% increase in speed. Handy but nothing to rave over in ads, when there are so many other improvements! The other change is that your integer variable eats up only two bytes instead of 8.
Nice new command: VALHEX. Ever wondered what >A000 was? Then use this command, as follows: PRINT VALHEX("A000") and there you are. Unfortunately the reverse is not available!
You can now use a variety of keys to terminate a input, as well as ENTER and FCTN E and X. Whenever you do enter data, a predefined variable called TERMCHAR is set and can be referenced in your program at any time until the NEXT input! Then your program can branch depending on whether ENTER or AID or BEGIN!!!!..... catch the drift? Unfortunately REDO has not been implemented. Pity.
The various ram card commands can be used in your EXBas program now, such as CALL PART, EMDK, VOL, and RDDIR--- but for some reason these must each be on a program line of their own. Not serious, but undocumented.
Extra commands include PEEKV and POKEV.

TECHNICAL STUFF

Myarc ExBas comes in the form of a module, a disk , an eprom and a manual. Not bad for the price. Good value!
Everytime you QUIT you must reload the whole system from floppy.- now remedied in Vn 2.11 - Myarc are quick to react to minor comments like this! - in VN 2.11, the file 128KOS is modified and instead of slavishly loading everything everytime, checks memory to see what is there. In general, provided you have not switched the ram card off, the system will only need to reload two files from the system disk, a great saving in time!

WRITER-WROUTES

by **Jane Laflamme**

This month I am going to talk about indents and outdents (what a word!). The command for the Formatter is .IN (+ or -)(n). If you specify .IN 10 the formatter will place the first character of the first paragraph of text on the 11th column of the printer (remember column 0 is the first column) and every first character after a CR symbol on the 11th column (providing you are in word wrap, of course). If you have specified the left margin as .LM 10, you won't have an indent! They both are set at the same column. If you wish to make it relative to the left margin, then the command should be .IN +10; your indent then will be at the 21st column; you can of course, use .IN 20, with no plus sign: it is not relative to the left margin but produces the same results. The plus (or minus) sign can be used on either the .LM or .IN, among others.

There have been a few times I have wanted to do the following:

(a) Now is the time for all good men to learn how to use the formatter (women also, for that matter!)

(b) It is very easy once you learn how to do it; but best if you try out all these things for yourself!

(So the prose is a little less than inspired!) This was done with outdents. Before the first paragraph (a), I inserted the formatter commands ".LM +4;IN -4". .LM +4 told the formatter to set the left margin four more than it was before, or plus 4; the indent command, minus four relative to the Left margin. With the plus and minus signs added to the indent command, it is always relative to the left margin setting. At the end of the two paragraphs, I reset my Left margin with the command .LM -4 and continued my text as before.

Thank you to all those who gave me suggestions for my column. I will incorporate them in the future. Just keep those letters and cards coming - to the address on the Newsletter, or phone 837-1719 evenings, or 745-2225 days!

>

TI-WRITER+PLUS

LA TRANSLITERATION

par Michel Johnson

La translitération, une des plus puissantes instructions du TI-WRITER, permet au programme d'adresser des commandes à l'imprimante. La **TRANSLITERATION** permet les changements de caractères à l'imprimante, les accents français et tout caractère graphique issu de votre imagination. **TRANSLITERER** veut dire "remplacer un caractère par un ou des autres".

Le mois dernier, nous avons vu la commande **"RS"** ou **"Replace String"**; cette commande de l'**EDITEUR** permet de remplacer **A L'ECRAN** un caractère ou une expression par une autre. La **TRANSLITERATION**, elle, s'adresse **A L'IMPRIMANTE**. Comme tous les commandes du **FORMATEUR**, la commande de translitération s'écrit avec un "."(point) suivi de deux lettres, d'un espace puis des instructions spécifiques. Une commande de formatage doit toujours être placée au début d'une ligne.

Prenons l'exemple du mois dernier : on tape notre texte en utilisant "=" (égal) en prétendant qu'il s'agit de l'apostrophe (pour taper plus vite). Evidemment, à l'écran, chaque fois qu'il devrait y avoir une apostrophe "'", on voit un "="
ex.: j=écris à l=écran

Nous avons alors utilisé la commande **"Replace** String" pour remplacer par après tous les "=" par des "'". Nous aurions aussi pu procéder en utilisant la translitération selon l'exemple suivant

:

ex.: .TL 61:39

61 : le caractère "="
39 : le caractère "'"

Placé au début d'un texte, cette commande dit à l'imprimante de remplacer tout caractère "=" **(61)** par le caractère "'" **(39)**. Evidemment, on continuera dans ce cas à lire des "=" à l'écran:

ex.: j=écris à l=écran

alors que l'imprimante écrira :

ex.: j'écris à l'écran

Jusqu'à présent, rien de bien sorcier, voyons maintenant les caractères de controle de l'imprimante pour comprendre la vrai puissance de la translitération.

LES CARACTERES DE CONTROLE SIMPLES

Les codes de caractères 32 à 127 sont les caractères visibles à l'écran et correspondent aux lettres, chiffres et symboles connus. Les codes de 1 à 31 sont les caractères dits de controle : ils sont utilisés par plusieurs logiciels comme les logiciels de communication et TI-WRITER. Parmi les codes "connus", notons :

- 7 signal sonore "BEEP"
- 8 retour arrière ou "BACK SPACE"
- 13 retour de chariot ou "ENTER"
- 27 échappement ou "ESCAPE"

C'est donc dire qu'on peut intégrer au texte des "back space", des "beep", etc... Une des utilités de ce controle est la possibilité de jouer avec les accents français.

LES ACCENTS FRANCAIS A L'IMPRIMANTE

Toutes les imprimantes à points sont équipés d'option permettant le choix de différents caractères correspondant aux langues courantes (anglais, français, allemand, etc...). Parmi les caractères français préprogrammés, nous retrouvons principalement les "**éèî!**". Par contre, les "ê" ou autres lettres nécessitant l'emploi de l'accent circonflexe ne sont pas disponible et doivent être créés. Rien de plus simple avec la tranlitération.

Prenons l'exemple du "ê" ; ce qu'on veut que l'imprimante fasse, c'est qu'elle écrive un "e", recule d'un espace et place un accent circonflexe sur le "e" avant de continuer. Par convention, la version française de FUNNEL-WRITER utilise **FCNT T** (accolade droite) pour le "ê" ; on pourrait prendre n'importe lequel des caractères mais on a intérêt à prendre un caractère qu'on n'utilise pas souvent comme **FCTN T**. On procède donc comme suit :

ex.: .TL 93:101,8,94

93 : accolade droite
101 : caractère "e"
8 : "back space"
94 : accent circonflexe

Donc, à chaque fois que l'imprimante rencontrera le caractère **93** (accolade), elle imprimante le caractère **101** (e), reculera (back space) et imprimera le caractère **94** (accent circonflexe).

On peut faire la même chose avec n'importe quel accent sur le même principe. Il faut seulement se rappeler que ce que nous verrons à l'écran ne correspondra pas au résultat de l'imprimante. Il existe par contre des versions de FUNNEL-WRITER qui affiche A L'ECRAN (et à l'écran seulement) des caractères français prédéterminés tels le "ê".

LE CHOIX DU TYPE DE CARACTERE

En consultant votre guide d'imprimante, vous constatez que vous avez la possibilité de différents types de caractères comme par exemple :

LE MODE ELARGI

LE MODE CONDENSE

LE MODE SUPERSCRIPT

etc...

Ces modes peuvent être appelés en tout temps par les codes identifiés dans votre guide. J'utilise par exemple dans mes rapports officiels un caractère foncé pour une meilleure présentation : le mode condensé. Le code de ce mode est "**Esc** E" ou "27,69".

"**Escape**" (le caractère 27) va indiquer à l'imprimante qu'elle se trouve en mode d'échappement et que les caractères qui vont suivre (même s'ils sont supérieurs à 32) ne devront pas être considérés comme du texte.

"**E**" (le caractère 69) complète cette instruction et l'imprimante s'exécute immédiatement.

Pour envoyer cette instruction à l'imprimante, on utilise l'instruction ".TL" comme suit : on prend un caractère quelconque, par exemple la lettre "Z" (caractère 90) :

```
ex.:0001 .TL 90:27,69
    0002 Z
    0003 .TL 90:90
```

Le caractère "90" (Z) sera donc remplacé par les caractères "27" et "69", lesquels feront réagir l'imprimante. A la deuxième ligne, on met justement un "Z" parce qu'on veut immédiatement envoyer ce code à l'imprimante avant que celle-ci ne commence son impression. A la troisième ligne, on dit à la lettre "Z" de revenir elle-même puisque le message est passé et que l'imprimante le conservera tant et aussi longtemps qu'un contre-ordre ne lui sera pas donné. Dans le présent exemple, on a donc utiliser temporairement la lettre "Z" comme messager.

CE DONC BEN COMPLIQUE...

Pas du tout! Lorsque vous avez compris le principe, vous pouvez vous batir des fichiers types comme :

```
0001 .TL 90:27,69      (mode emphasé)
0002 Z                 (exécution)
0003 .TL 90:90         ("Z" à nouveau "Z")
0004 .TL 93:101.8,94   (le "é")
0005 .FI               (ces codes seront
0006 .AD                expliqués plus tard) ect...
```

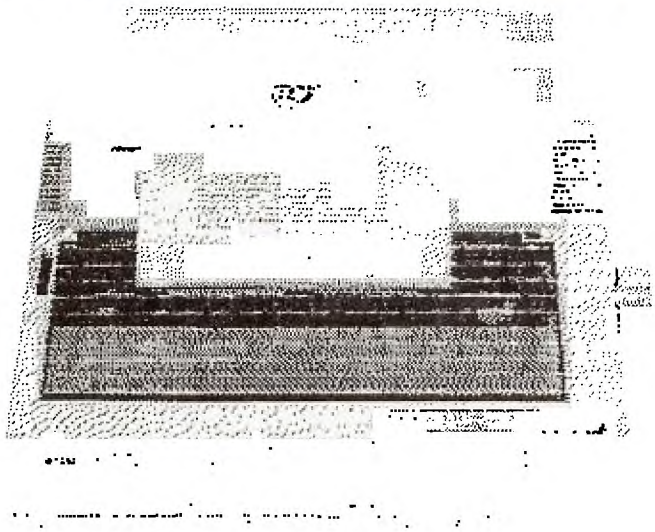
Vous sauvez ce fichier sous un nom tel DSK1.TYPE. Chaque que vous avez une lettre à écrire, entrez d'abord en mémoire ce fichier, tapez votre texte puis sauvegarder le tout et le tour est joué. Mieux encore, il existe des fichiers types que vous pouvez appellere de votre texte et qui vous simplifie davantage cette opération, mais là, c'est le sujet de discussion du mois prochain...

Introducing the

NX-10

star micronics inc

THE POWER BEHIND THE PRINTED WORD



Only the Star NX-10 gives you full front panel control in a 120 cps draft/30 cps near letter quality printer.

This printer offers features which are simply unavailable on other printers in it's class. Features like a complete front panel which does not require the user to fumble with DIP switches to use the printer's features. And when you and your program can't agree on printer settings, our unique panel-lock mode makes sure you win.

You can even set margins and align forms right from the front panel.

When you don't need the appearance of near letter quality, but time is of the essence, the NX-10 provides a quick 120 cps draft mode.

Other features include a built in tractor, automatic single sheet feeder (bin type cut sheet feeder optional) an easy-to-change cartridge ribbon (no dirty fingers) and switch selectable IBM PC graphics compatibility, plus a wide variety of fonts:

Headline mode lets you shout out loud!!

Or even louder!!!!!

Condensed elite (20 characters per inch) is great for the fine print on contracts and legal documents, or those wide spreadsheets.
For the mathematician, superscripts and subscripts are easy.
Eleven foreign character sets are also standard, along with 81 IBM special characters -- including block graphics. For non-IBM computers, an ESCAPE P standard command set is included.

Print pitches of 5, 6, 8.5, 10, 12, 17 and 20 plus proportional characters and seven graphics modes makes the NX-10 the best value for all of your printing needs.

For more information about Star Micronics NX-10 printers, contact:
EXCELTRONIX INC. 217 Bank Street, Ottawa, Ontario K2P 1W9
Telephone: (613) 290-9000

Exceltronix

FROM

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