PORT FE

SORCERERS USERS' GROUP

(Toronto)

P.O. Box 1173 Sta. 'B'
Downsview, Ontario,
Canada. M3H 5V6

SORCERER

Newsletter

The Toronto Sorcerer Users' Group was founded in the Spring of 1979, a handful of willing and eager to learn members.

This newsletter shall at all times keep in mind the goal at its conception. To spread the seeds of knowledge.

Articles printed in this newsletter shall be free for all Sorcerer Users' groups to reprint or comment on as they see fit.

Articles submitted for this newsletter must be in no later than the beginning of the 1st of every month.

September 1981 ISSUE

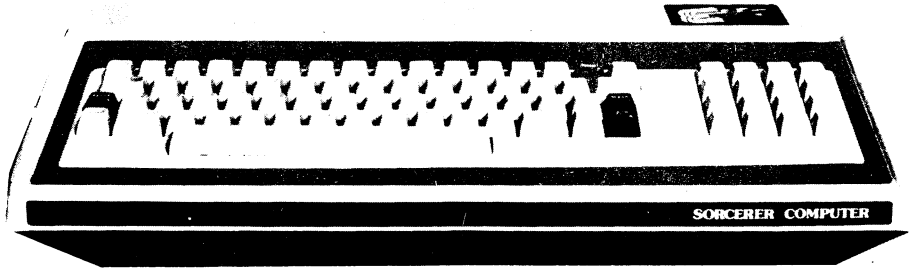
TABLE OF CONTENTS

This issue dedicated to CP/M - Re EXBASIC/MBASIC/EXCAS Ver.5

MEETING PLACE

Location : Bathurst Heights Library - Date: Thur Sept.17 - 7:00 PM
3170 Bathurst St.

One block north of Lawrence on the west side of Bathurst.



Command Name: FIND	Command Name: FINDN
# of Arguments: 3	# of arguments: 1
Location: 9B01H	Location: 9B0EH

This function is used to search through string arrays for a particular array element.

Command Use:- CALL FIND(ARRAY$(1),STR$,INDEX%)

If STR$ is found in ARRAY$, INDEX% is set to the array element in which the array element was found. INDEX% must be an integervariable. If the string is not found, INDEX% is set to -1. Also if ARRAY$ or STR$ are null variables, it also returns -1.

CALL FINDN(INDEX%)

Once an initial call to find has been done, and you wish to search the rest of the array to see if there are any more matches of STR$, call 'findn' and this will do the trick. Make sure that you do nothing drastic to the string array area such as executing FRE(0) or CLEAR.

Special Note

The program will give incorrect results unless it is past the starting element of the array. The program gives the correct result with OPTION BASE 1. With OPTION BASE 1, the element that should be passed to the function is ARRAY$(1).The default format for Exbasic arrays is OPTION BASE 0, so the index will be 1 more than the actual index in the array.

Special Features:- ? can be used to match any character

```
10000 DEFINT A-Z:ADR=-25855 :READ A:WHILE A()-1:POKE ADR,A:
      ADR=ADR+1:READ A:WEND:END
10001 DATA 205,038,155,205,072,155,205,085
10002 DATA 155,205,093,155,201,034,194,155
10003 DATA 042,189,155,125,254,001,202,173
10004 DATA 155,180,202,173,155,043,034,189
10005 DATA 155,205,093,155,201,043,043,126
10006 DATA 050,189,155,035,126,050,190,155
10007 DATA 035,034,187,155,237,083,191,155
10008 DATA 026,050,193,155,237,067,194,155
10009 DATA 033,000,000,034,185,155,201,042
10010 DATA 189,155,125,180,192,225,017,255
10011 DATA 255,195,166,155,058,193,155,254
10012 DATA 000,192,024,241,042,189,155,125
10013 DATA 180,202,078,155,042,185,155,035
10014 DATA 034,185,155,042,187,155,229,035
10015 DATA 035,035,034,187,155,225,058,193
10016 DATA 155,190,040,009,042,189,155,043
10017 DATA 034,189,155,024,215,071,035,205
10018 DATA 178,155,229,042,191,155,035,205
10019 DATA 178,155,235,225,026,254,063,040
10020 DATA 004,190,194,125,155,019,035,016
10021 DATA 243,237,091,185,155,042,194,155
10022 DATA 115,035,114,201,017,255,255,024
10023 DATA 244,235,026,111,019,026,103,201
10024 DATA 000,000,000,000,000,000,000,000
10025 DATA 000,000,000,245,126,047,133,111
10036 DATA-1
```

COLOURFUL ISN'T IT

Ever pulled your hair out with massive frustration from typing in long, big, tedious data statements or REM's just to find out that you made a BOO BOO. Maddening isn't it!! Or what about trying to correct or even restructure a program already written? Get the picture - it's more like trying to walk a mile with shoes 1 size too big-gets a mighty uncomfortable.

This onscreen editor is a master feat of technical genius with only 8 basic functions; it makes easy play of almost any problem. ^E allows you to enter or exit the EDITOR; once in simple use the SHIFT KEY and CURSOR ARROWS allow you to get whereever you want. NOTE once the correction is made HIT return first, then space down to the bottom of the program - TYPE LIST and voila correction done.

For myself I study Computers by day which involves a lot of Structured Programs and these next commands I personally love. Have you ever tried to correct or alter nested IF statements that use the same line number or even try to add to a line without retyping it all over again. Very easy just use these 3 functions ^E, ^B, ^V. Example below.

Example 1

```
10 FOR I=1 TO 25
20 PRINT "*";
30 NEXT I
```

To change the semi-colon in line 20 TYPE ^E, cursor up to line 20 and right till the nice bright square flashing cursor is over the semi-colon type what you want or hit the SKIP key to erase then NOTE hit a carriage return. Space down to the bottom of the program and type LIST and Lo and behold a changed line.

To add to Example 1 , let's say you want to put in a TAB(10) then TYPE ^E to enter the editor cursor up to line 20; space over to the first quote and TYPE ^V which displays a right arrow and type in your addition. REMEMBER to type a carriage return when the change is made. Your program now looks like this:

```
10 FOR I=1 TO 25
20 PRINT TAB(10);"*";
30 NEXT I
```

SIMPLE WASN'T IT.

Simply type in the first DATA line and carriage return when done. Now type ^E to enter the EDITOR and make the changes to the line number or data entries as necessary hit return and repeat while in the EDITOR as many times as necessary. You'll be very surprised at how fast and error free a program can become to enter.

This EDITOR is a "must have" for any avid programmer since the time it saves and ease of use defy anything presently or imminently available. Overall if you have tried and used the present EDITOR available,one look at this MASTERPIECE in operation is all that you need to see how great it is.

The program is available from Northamerican Software in the form of an insert program, which inserts itself into your EXBASIC. The only prerequisites are that you have EXBASIC.COM & DDT.COM or SID.COM. It comes on tape which means all you have to do is load it at 100H and save it to disk. Please note: This EDITOR will only work with EXBASIC and <u>not</u> MBASIC. The surprizing thing was it only cost $49.95 U.S. I thought that was a bargain, for all it can do. Now I don't even mind programming in basic at all.

by: Frank Aylesworth

I often get asked how you access machine language programs from BASIC, which is not suprising since the documentation of the subject is less than sparse. The simplest way to do this is to load your program into memory then execute it via a USR statement. When using machine language and BASIC you will probably end up using the some of the following commands:-

DEF USR CALL USR PEEK POKE VARPTR CLEAR HEX$

1. How to get the machine language into place and make it stay.

This is really a matter of taste. If you are using cassette you can simply load it in starting at an address that will not be eaten by BASIC. This poses some interesting problems because BASIC will eat up all the memory it can find, then do a garbage collection, and probably collect up your routine in the process as well.

In the monitor there is a routine that sets the highest usable ram location. I know that EXCAS uses it but, I am not sure about how the disk versions treat it.

The disk versions have a way that you can protect a program but you have to specify the address in the command line when you call EXBASIC. Not an attractive situation at all. Try telling a novice user that to make the program work, he has to type:-

EXBASIC FOOBAR /F:4 /M:&H9FFF

and see how long he can remember it.

The solution that I came upon was the CLEAR statement. By using the second and third parameters you set the highest available memory location available to BASIC, plus the amount of BASIC stack space.

Now we have the knowledge to make the program stay, we still don't know how to get it into place. If you have any of the Microsoft assemblers you can simply generate code for high memory and just load it there, then load in BASIC. This is a kludgy situation at best, so I don't do it that often. The way I usually load in a machine language file is to convert it to a series of data statements then simply just poke in the data statements. I use a program called MAKEML to do this. (It was published some months ago ... but for new readers here it is again.)

```
100 A$="Make Machine Language => BASIC":PRINT :PRINT A$:PRINT
    STRING$(LEN(A$),137):PRINT
110 DEF FNMS$(A)=RIGHT$(STR$(A),LEN(STR$(A))-1):
    DEF FNP$(A)=LEFT$("000",3-LEN(FNMS$(A)))+FNMS$(A)
120 INPUT"Output Filename";F$:C=10001:OPEN "O",1,F$
130 INPUT"Start Address of Program";ST:INPUT"End Address of
    Program";EN
140 PRINT#1,"10000 CLEAR ,";ST-1;",&H400:DEFINT A-Z:ADR=";ST;":READ A
    :WHILE A(>-1:POKE ADR,A:ADR=ADR+1:READA:WEND:END":A$=""
150 FOR X=ST TO EN STEP 8
160     FOR Y=X TO X+7
170             A=PEEK(Y)
180             A$=A$+FNP$(A)+","
190     NEXT Y
200     A$=LEFT$(A$,LEN(A$)-1)
210     PRINT#1,C;"DATA ";A$:A$=""
220     C=C+1
230 NEXT X:PRINT#1,C+1;"DATA ";-1
240 CLOSE 1
250 END
```

Note line 140. It automatically generates the code necessary to protect the program from BASIC and it also writes a small program that pokes in the data.

2. How to convert the machine language into a BASIC file.

Say I had a program called Fill that I wanted to use with EXBASIC. I have assembled the program and I know that I want it to sit at 9900H. Here is the procedure for making a data file with the program in it.

1. Assemble the file. If you specify relocatable code you can put it anywhere without having to change the ORG statement in the program.
2. Load the program. Using EXLINK it would be EXLINK FILL 9900
3. Exit out of the loader.
4. Call in BASIC and protect the routine so BASIC will not kill it.

```
EXBASIC MAKEML /M:&h98ff
```

5. Give MAKEML the info it needs to write the data file. You can get the ending address from EXLINK. The program will then grind away for a few minutes making the file. You now have a chainable version of your program on disk.
6. At this point it may be a good idea to test out your new program to make sure that everything went smoothly. I also do a SAVE of the data file because it loads faster and takes less space than the ASCII version.
7. Whenever you use the program in future, there is no need to type /M:&h98ff on the command line. You can merely type in EXBASIC FILL and it will load in the machine language and automatically protect itself.

The file generated by MAKEML can be MERGED with another program so you can have a program with machine language subroutine in place.

3. Now that it is in, how do I get to it.

The simplest way to do this is by using one of the USR command hooks. There are actually 10 USR commands available, the default being USR which is USR0. So to execute our fill routine we use:-

```
DEF USR=&H9900:A=USR(0)
```

and the screen is filled. But how do we pass something to the routine like a character or a string so we can fill the screen with any character?

4. USR and parameter passing.

USR is very useful when you only have to pass one piece of information to your machine language routine. The USR routine does a variety of things depending on what type of data that is to be passed. In all case the accumulator A) contains a number specifying what type of data is being passed and HL points to floating accumuator (FAC) where the data is stored. (When passing a string, HL points to the string descriptor, but more about that later.)

5. Table of what gets passed to a routine.

Arg Type	Acc Value	HL points
integer	2	at data (Manual is wrong)
string	3	at string descriptor
sfp	4	see manual
lfp	8	see manual

6. What about return parameters?

To return a value to BASIC all you have to do is modify the data that is pointed to by HL when the routine is first called, and BASIC will take care of the rest. Watch what you do though, it is possible to send some very strange stuff to BASIC through the USR command.

7. Where do I go from here.

The only way to really get the feel for something is to try it out. In future articles there will be concrete examples plus the articles on the way EXBASIC stores all the different type of variables.

Exbasic/Mbasic/Excas Version 5 Machine Language Calls

Any Microsoft version 5 BASIC has two means of accessing machine language routine. These two commands are USR and CALL.

CALL Statement

The CALL statement is the most sophisticated of the two of these. With these you can pass multiple parameters to your routines. The intialization of the CALL statement is as follows:-

```
PLOT=&H4567:CALL PLOT(A,B,.....)
```

(The &H specifies that the number is expressed in hex)

The simplest form of the CALL statement would just be something like CALL PLOT which, would simply execute the routine at PLOT, and no parameters would be passed. (An example of this is the GINIT routine in my plotting program).

The next step up in this process is passing of up to three parameters. Exbasic uses the following convention with three or less parameters:-

Parameter #	Register Pair
1	HL
2	DE
3	BC

If there are more than three parameters then the first two parameters are passed in HL and DE, and BC points to a block of data that contains the other parameters. In the back of your Exbasic Manual is a listing of a program called $AT which will transfer these parameters into a local storage area.

VERY IMPORTANT NOTE

The parameters passed in the registers are only pointers to the actual arguments. They are NOT the arguments. Exbasic does not check the number of parameters passed, so it is up to programmer to make sure that his routine gets the right number of arguments. Depending on what is passed, the parameters will point to different things. Page C-3 of the manual gives a fair accounting of what each parameter points to. Another point that causes a fair amount of problem is the fact that you can only pass a variable and not a number or a simple string. They must be in a variable or Exbasic will not take it.

Integer Variable Storage in MBASIC/EXBASIC/EXCAS Ver. 5
USR Parameters

For any of the following:-	USR(A%)	USR(1%)	USR(A%(X))
the parameters are all the same:-	A=02	(Integer Spec)	
	HL	(Ptr to actual data in twos complement notation)	

NOTE: In the manual I have it states that FAC-3, FAC-2 are the actual data. This is an error in the manual.

An actual examples of values presented is:-

```
A=10:C=USR(A)
HL=4EA8   (For EXCASD)
A =02
4EA8: 0A 00
```

Say that some sort of operation is done on A, and the results must be passed back to BAIC. This is done by depositing the result at (HL) and executing a return. BASIC will take care of the rest. This works with all the USR calls, whether they are with strings, or floating point numbers.

eg. If the example above was used, and the final result came to 11 (0B), you could pass 11 back to the program by putting 11 in 4EA8.

CALL Parameters

```
CALL RT(X%)     HL=4F5A      (Pointer to actual variable data)
CALL RT(A%(2)) HL=4F71       (Pointer to actual array element 2)
```

To pass a whole numeric array, pass the first element in the array to the machine language routine. Be sure that you know whether it is the 0th or the 1st element, as this will make a big difference.

Variable Table Storage Format for Integers

eg. NUM% Value of 10

```
02        Integer variable token
4E        N
55        U
01        Length of remainder of variable name
CD        M with bit 7 (MSB) set
0A        Value in twos complement form
00
```

eg. TE%

```
02        Integer variable token
54        T
45        E
00        Length of rest of variable name (none in this case)
05        Value in twos complement notation
00
```

eg. NUM%(5)

```
02        Integer variable specification
4E        N
55        U
01        Length of rest of variable
CD        M with MSB set
0D        Length of rest of array data
00
01        Option base data (0 or 1)
05        Number of array elements
00
01        First value
00
02        Second Value
00
```

```
03          Third Value
00
04          Fourth Value
00
05          Last Value
00
```

A Note about Integers

Integers have a lot going for them. They are short (2 bytes), fast, and they can be used directly in machine language. Integer indexes in FOR...NEXT loops will run far faster than regular (single precision) indexes, so if you want to speed up program execution use a lot of them. If a program does not use any floating point, you can go DEFINT A-Z which will define all the numeric variables to be integers. Most games can be speeded up by putting this in a program.
There is a major difference between the way Ver 4 and Ver 5 handle integers. Ver 4 truncated integers without rounding while Ver 5 rounds integers up. Programs written that use this fact, should use the INT statement so that the variables work correctly. In array indexes this causes a problem because, Ver 5 rounds, then truncates so you may think that you have the correct index, when in fact you have one more than the index you want. I had to use A(INT(X),INT(Y)) in a lot of array specifications to make the program work correctly.

Exbasic 5.04 String Variable Storage

When using the USR command, register convention is as follows:-
For a string variable (simple variable eg. A$)

DE => pointer to string descriptor

Format Of String Descriptor
Byte 0 => String length
Word 1 => Pointer to actual string text

String Variables in Symbol Table (THISIS$)

Byte	Description
03	Indicates next n bytes are a string variable
54	First ASCII byte of string variable (T)
48	Second ASCII byte of string variable (H)
04	Length of variable name - 2 (rest of variable name)
C9	
D3	Remainder of variable name with MSB set (Bit 7)
C9	
D3	
0E	Length of string
4E	
4D	Pointer to string text

Example 2 (T$)

Byte	Descriptor
03	Start of string variable
54	First character of variable name
00	Second byte of string byte (In this case none)
00	Length of rest of variable name
0E	Length of string variable
4E	
4D	Pointer to actual string text

Variable Storage for String Arrays (THISIS$(20))

Byte	Descriptor
03	String identifier
54	First character of name
48	Second character of variable name
04	Length of rest of variable name
C9	
D3	Remainder of string variable name with MSB set
C9	
D3	
15	Length of array pointer from starting subscript
00	
01	Base subscript (set by OPTION BASE)
06	Current number of active elements in the array
00	
0E	Length of first element in array
5F	Pointer to text for first element
4F	
0E	Length of second array element
56	Pointer to text for second element
4F	

All the remaining pointers in the array are set to 00 unless used.

Notes

Since there are two bytes allocated for the number of elements in the array, the probable maximium number of elements in a string array is 32768.

When the VARPTR function is used on a string variable it returns a pointer to the 3 byte string descriptor.

When using the CALL statement with strings, the parameter given in any one of the registers is only a pointer to the string descriptor or the FAC; therefore it is up to the programmer to know what parameters will be passed to the machine language program.

A useful function that can be used from BASIC is:-

```
DEF FNADR(A$)=(PEEK(VARPTR(A$)+1)+256*(PEEK(VARPTR(A$)+2)))
```

This will return the pointer to the actual string text, so from a program you can change the contents of a string in a different way. I use this technique for storing machine language in a string.

(see USR a simple application on Page #11)

WHY CANADA IS A COPYRIGHT HAVEN FOR US SOFTWARE

EDP and the Law. The anomaly that American software can get better protection here than at home arises from several facts, notes lawyer Dan Mersich. Here's a look at the implications.

Although not intentionally so designed, it seems that the Canadian Copyright Act as it applies to software provides a fine haven for registra
tion of foreign
packages, notably those from the United States.

The anomaly that American software can get better protection here than at home arises out of four facts.

First, that the publication of an original work destroys any protection it might have had under the laws of trade secrecy. One cannot claim to be the holder of a valuable trade secret while, at the same time publishing it for all the world to see.

And even though trade secrecy is probably the best vehicle for protection of software, many houses like to get as much protection as they can and therefore look to copyright law also.

The second fact is that a formal registration is mandatory under U.S. copyright law, and that procedure requires the author to submit (and thereby reveal) portions of the work to be protected. Because this arguably constitutes publication, American software owners must choose between copyright and trade secrecy as methods of protection.

The third fact is that Canadian copyright law allows for the registration of an unpublished work, and does not demand a sample; hence it does not impair its status as a trade secret. Incidentally, the licensing of software to a user under a contract with a confidentiality clause in it does not constitute publication.

The fourth fact is that copyright protection under Canadian law is available to any person who is a citizen of a country which adheres to either the Berne or the Uniform Copyright Conventions (which the U.S. does). This means that American software can enjoy world-wide copyright protection merely by being registered in Canada. And, it can do so without being stripped of the protection it has under American trade secrecy laws.

On top of all that, the cost of registration is a trifling $25 along with completing a simple descriptive form. Not since the days of the home-steaders when free land was given away in the west has there been such a bargain.

What does registration buy? For starters, it buys a dominant position in a court room. In particular, a court will presume that the software is a valid and protectable work under the Copyright Act simply because it has been formally registered. In so doing the court places a heavy burden of disproof on the shoulders of anyone who challenges the copyright-say for example on the grounds that the work was not original; and therefore should not have been given a copyright in the first place. Infringers usually try to establish this as their first line of defence.)

Registration also buys a certain amount of deterrence. A piece of software which carries the added endorsement of formal registration has a greater sobering effect on would-be pirates. True, some pirates are not frightened off by anything, but most people who toy with the idea of misappropriating software will likely be dissuaded. Considering the extremely low cost of registration, it is well worth it, even if only a single case of piracy is avoided.

Would registration in Canada foreclose registration in the U.S. at some later date if the author so desired? No. The main requirement is that the work be original ,i.e. not copied.

The fact that it has been previously registered elsewhere or licensed to users does not affect its originality.

The rest of the world views the national character of Canada as being mostly grey and perhaps slightly mid-Victoriao in its moral outlook.(Although one must admit that Pierre and Margaret Trudeau have taken large steps to dispell that image.)

It is therefore a comical irony that circumstances have made Canada a first-class copyright haven just as the Grand Cayman and Liechtenstein are first-class tax havens; Canada is akin to the new school marm who unwittingly takes up residence next to a house of pleasure. Despite her protestations she probably gets a secret thrill out of it all. Credits to Dan Mersich

For further information contact Tony Bagshaw at Port FE, P.O. Box.

NOTE TO MEMBERS

The up and coming meetings are listed below for your convenience.

October	:	Wed 14th	7.00 p.m.
November	:	Wed 18th	7.00 p.m.
December	:	Wed 16th	7.00 p.m.

Unfortunately we have no choice about the meeting in November with regard to it falling on a Wednesday as well. This completes the scheduling for the balance of this year.

It seems as if in the August issue of PORT FE we made inroads to one of the Remote Bulletin Board Systems in Canada and the U.S.A. with the use of a MODEM.

I received a phone call from a close colleague who lives in Troy, Michigan. He goes by the name of Ralph LaFlamme and is secretary & editor of the "Sorcerer's Apprentice" newsletter, P.O. Box 1131, Troy, Michigan 48099. They agreed some time back to have reciprocal exchange privileges with us. Well it seems that our last PORT FE interested him in respect of their own BBS system operating under the name of the SORCERER'S APPRENTICE COMPUTER BULLETIN BOARD SERVICE (Tel. 313-535-9186). This board is one of the variety that has a ringback feature which means the following, i.e. You first dial the number then listen for one or two rings and hangup. Now redial the same number (this is what ringback means) and wait for the high pitch tone indicating the CBBS is ready to communicate, place the phone into your modem cradle and type (CR)'S until the CBBS requests a number (0-9) specifying how many nulls your system needs. This CBBS system supports the following baud rates 110,300,450,600 & 710 (the latter via the "NEWBAUD" program) and requires 8 bits and no parity - 2 stop bits at 110 baud. If you get this far your next operation is the password. (SORCERER) You have five attempts to type the password in correctly. At this point the system will inform you that it is booting CP/M and an "A)" prompt will appear to indicate that you are logged into the "A" disk and CP/M is ready to receive your command (the "B" command brings up the "B" drive.) The "DIR" command requests that CP/M print a directory of disk files to the CRT. The "HELP" command lists all system instructions, which includes an "R" command whereby other members leave messages for one another or for ALL members. Finally to add cream to the pie, any programs that are listed can be downloaded in ASCII files from this CBBS or any other BBS system using the correct TERMINAL program and a modem.

I wish to acknowledge with thanks all pertinent details herein supplied by Robert Hageman of the Sorcerer's Apprentice for the inclusion of this article.

by: Tony Bagshaw

The following graphic was received from a Jean Pickett of Princeton N.J.

Jean Pickett

THE WORD PROCESSOR SORT

At last a speedy Z-80 sort routine that integrates with the Exidy Word Processor as a new command. Menu driven, extremely flexible, allowing multiline records in many formats. Sorts on any word in the record. Turns the Word Processor into the easiest-to-use mailing list and data base system available to Exidy users. Specify memory size, cassette or disk. Supplied on cassette for both. If ordering for disk use, specify where you locate your DOS and cold boot programs so that we may supply a version which does not conflict. $47.50 U.S. with instructions.

ROGER HAGAN ASSOCIATES
THE DECISION EDIT
1019 Belmont Place E.
Seattle, WA 98102

The examples presented below are very simple things that you can do with the USR function. The source code is included so that the linking mechanism can be seen. This in conjunction with my other article should provide a starting point for experimentation. One last thing about USR, if USR is called with a string parameter, the parameter it will return will be a string so A=USR(A$) will give a TYPE MISMATCH ERROR.

The sample program was assembled and tested on a 48k Sorcerer II running Exidy CP/M 1.43. The code was linked to sit at 9b00h (this should be around 7b00h for a 32k system). The BASIC data file gives the code that sits at 9b00h. All the code is totally relocatable so if it doesn't work where it currently sits, put it somewhere else. If you do so, don't forget the CLEAR statement at the beginning of the program!

If you have an questions about the USR/CALL interface you can contact me through Port FE or at home. My home address is:-

Jacques Giraud
26 Arlington Ave
Toronto, Ontario
Canada M6G 3K8

Tel:(416)-656-9646
I am usually
home after
6 pm most days.

1. Program Assembly Source Listing

```
          title     An Example of the USR Function
          name      ('USR')
          .z80
;
          .comment *
```

This is the source file for 2 small USR functions. The two functions are a FILL command that will fill the screen with a given character and keyboard input routine that will get a character from the monitor without the user having to hit return.

Fill Command

There are two usages of the Fill command. You can call it with either the ASCII number of the character or you can give it the character you wish to fill the screen with in a string.

Format 1: A$=USR((string exp))

Expected Parameters

A =3
DE=Pointer to string descriptor
Note: A string descriptor is 3 bytes that gives the length of
the string and location of where the string text is.

First off we have to decide whether or not it is the string or the ASCII version of the Fill command.

```
*
;
string    equ       3          ;value A has when string passed
int       equ       2          ;value A has when passed a number
screen    equ       0f080h     ;start of screen on Sorcerer
;
fill::    cp        int        ;check for number passed
          jr        z,filla    ;use ASCII version
          cp        string     ;check for string version
          ret       nz         ;none of the above so return
;
```

```
fills::   inc       de          ;ignore length of string
          ex        de,hl       ;hl=pointer to string descriptor
          ld        e,(hl)      ;move pointer to string text into
          inc       hl          ;   De
          ld        d,(hl)
          ld        a,(de)      ;get first character in string
;
fillscr:  ld        hl,screen   ;set up for fill
          ld        de,screen+1
          ld        bc,1920     ;1920 character on screen
          ld        (hl),a
          ldir
;
          ret                   ;back to BASIC
;
          .comment  *
```

Format 2: A=USR((int exp))
Expected Parameters
A= 2
HL=pointer to ASCII number of character
*

```
g
fillX::   ld        a,(hl)      ;get the character
          jr        fillscr     ;use the fill screen routine
;
          .comment  *
```

The final example is how to return a parameter to BASIC. This function will read a character from the keyboard. If a character is present, it will return the ASCII value of the character. If no character is present it will return 0.
Expected Parameters
A= 2
HL=don't care (must be saved though for return parameter)
*

```
;
keybrd    equ       0e009h      ;get a character from keyboard
;
getc::    push      hl          ;save HL for return parameter
get1:     call      keybrd
          jr        z,get1      ;wait until we get a key
          pop       hl
          ld        (hl),a      ;put character in low byte
          sub       a           ;zero the acc
          inc       hl
          ld        (hl),a      ;zero out the high byte
          ret                   ;back to BASIC with value
;
          end
```

2 BASIC Data File of Assembly

```
10000 CLEAR ,-25857 ,&H400:DEFINT A-Z:ADR=-25856 :READ A
      :WHILE A<>-1:POKE ADR,A:ADR=ADR+1:READ A:WEND:END
10001 DATA 254,002,040,022,254,003,192,019
10002 DATA 235,094,035,086,026,033,128,240
10003 DATA 017,129,240,001,128,007,119,237
10004 DATA 176,201,L26<024,240,229,205,009
10005 DATA 224,040,251,225,119,151,035,119
10006 DATA 201,255,255,255,255,255,255,255
10008 DATA -1
```

LAST MINUTE CORRECTION~Page12
Run 10000 first. This puts the machine language in place
Line 10004 should read:

```
                                v  v
10004 DATA 176,201,L26<024,
                           126,
```

3 BASIC Sample Program

```
100 REM
110 REM SAMPLE PROGRAM TO TEST THE USR FUNCTION
120 DEFINT A-Z
130 REM DEFINE WHERE THE ROUTINES ARE
140 REM
150 DEF USR=&H9B00
160 DEF USR1=&H9B1D
170 REM
180 REM *** FLASH THE CHARACTER SET ON THE SCREEN ***
190 FOR X=0 TO 255
200 A=USR(X)
210 NEXT X
220 REM *** FILL THE SCREEN WITH THE CHARACTER TYPED ***
230 REM *** IF A=3 TESTS FOR ^C TO STOP PROGRAM
240 REM
250 A=USR(USR1(0)):IF A=3 THEN STOP ELSE GOTO 250
260 REM *** GIVE EXAMPLE OF CHARACETER USAGE ***
270 FOR X=0 TO 255
280 A$=USR(CHR$(X))
290 NEXT X
```

CONTENTS OF THE MACHINE CODE TUTORIAL PACKAGE as reviewed by J.L. Neale of SCUA.

1. Introduction - required reading - aims - hexadecimal briefing - the byte - MONITOR commands; DUmp, ENter, GO.

2. What is OBJECT code anyway - Assembler - hand assembly - the A register(accumulator) - Load a constant into register - Load a register into an addressed location - Return(and its use to give control to MONITOR) - Load contents of a byte into a register - Jump relative and Jump absolute.

3. The zero and carry flags and how to set them with the Compare instruction - Conditional jumps - SORCERER MONITOR routines - The MONITOR SEt commands for O & I - the Call instruction.

4. The BASIC USR function; how to call your machine code program - The MONITOR PP command - Add and the other registers - More Load instructions - register pairs (16 bit HL,DE,BC,AF) - INC & DEC.

5. PUSH; the STACK and what it does - POP - Conditional Calls and Returns - MONITOR commands; SAve, LOad, SEt X & F, Find - The cassette routine in MONITOR explained - BASIC CLOAD compared to MONITOR SAve - discussion on Video RAM.

6. BASIC statements PEEK,POKE and CHR$() - Cursor movement, some graphics information - MONITOR commands; MOVE and TEST - The reset keys, explanation of the MONITOR work area, "cold", "warm" and "user" starts.

7. BASIC work area - some useful routines - Bit, Set and Reset; AND, OR, XOR; NOP - IX and IY registers.

8. How to Join (end to end) two BASIC routines - BASIC statement structure - parallel and serial data port - cassette file format - a sound generation program comparing BASIC to machine code - OUT and IN instructions - a non-stop keyboard demonstration program.

References: A Guided Tour of Personal Computing by Exidy
A Short Tour of Basic by Exidy
Z-80 Op-codes by Zilog.

Northamerican Software – Limited Time Offer

(This offer is only valid on orders postmarked on or before November 30, 1981)
All items marked 'SPECIAL OFFER' will, after November 30, 1981 revert back to regular prices.

GOLF — Anybody for a round? Play 18 holes of GOLF (fully graphic) and complete with all the usual things, such as sand traps, trees, rough ground, streams, lakes and greens with close-up view. Better be prepared to get thrown off the course if you're not good enough or if you hack up the greens with anything but the putter. A single player game in BASIC and machine language. (BASIC & ML)
SPECIAL OFFER **$19.95** U.S.

NEW GAMES WITH SOUND EFFECTS

ECHO — Can you remember sequences of sight and sound? Enhance your ability to remember and train your mind. Master this skill with ECHO. Just a few minutes a day will improve and sharpen your mind. Four levels of difficulty. (ML)

SPIDER — WHO'S AFRAID OF SPIDERS? Imagine yourself in a very large room with obstacles all around you. Your objective is to reach the opposite corner and retrieve some money and get back again. Sounds simple enough, doesn't it? Well, you might just get eaten or surrounded. These spiders come out of the woodwork like ants. No matter how many you destroy, they just keep on coming. Fully graphic and with sound. (ML)

Both the above games come complete with a digital-to-analog sound generator (printed circuit board assembled) that plugs directly into the Sorcerer Parallel Port. All that is required is an audio amplifier and you're away to the races. Fully compatible with other software being currently sold. There is also provision to hook TWO joysticks for game input control. NOTE: The two programs supplied do not utilize joysticks.

TOTAL PACKAGE PRICE which includes the two programs ..... SPECIAL OFFER **$39.95** U.S.

DISK EXTENDED BASIC 'EXBASIC' ON-SCREEN VISUAL EDITOR

This is the EDITOR that you have been waiting for. This program will modify your EXBASIC and do away with the cumbersome Microsoft editor. You can modify your BASIC programs to your heart's content, just like on some of the other computers. The only pre-requisites are that you have CP/M and are running a 48K or larger system and have of course EXBASIC Ver. 5+. The program comes with full instructions in cassette fashion so that it can be loaded by any disk user. NOT compatible with MBASIC Ver. 5. (ML)
SPECIAL OFFER **$49.95** U.S.

EDOS — MICROPOLIS OPERATING SYSTEM

This system will enhance and allow you full use of Exidy ROM PAC BASIC with disk R/W routines. Are you doing it the old way? Very cumbersome to save BASIC programs. Well, at least you now can update and do it the easy way. EDOS comes on 5¼ inch disk (16-sectored Micropolis). (ML)
$59.95 U.S.

ZETU — CASSETTE-BASED Z80 ASSEMBLER

Look, an easy-to-use Z80 assembler for the thrifty-minded individual who can't afford those biggies. It will do what the big ones can, and guess what, it does it all in memory. No more two passes to the cassette. A very powerful little guy. Fully documented manual. With on-screen editor. *Disk drives not required.* (ML)
SPECIAL OFFER **$29.95** U.S.

EXBASIC — Full disk-extended BASIC	DP 7310	**$325.00** U.S.
Z80 Disk Development System	DP 7260	**125.00** U.S.
Exidy precision PRINT driver routines	DP 7221	**125.00** U.S.
WPP PAC to disk conversions	DP 7220	**99.00** U.S.
Exidy ROM PAC BASIC to Extended 19K BASIC	DP 7272	**49.00** U.S.
Exidy 19K BASIC to EXBASIC disk BASIC	DP 7271	**99.00** U.S.
MICROHOME programs	DP 7100	**30.00** U.S.
SORCERY cassette programs	DP 3003	**50.00** U.S.
SPELLBINDER — Word Processor		**375.00** U.S.
TOOLKIT for Sorcerer ROM PAC BASIC (2.3K ML)	SPECIAL OFFER	**29.95** U.S.
SWORD — Sorcerer Word Processor (4K ML)	SPECIAL OFFER	**29.95** U.S.
SUPER GRAPHIC SCRATCH PAD Ver. 2.2 (BASIC & ML)	SPECIAL OFFER	**19.95** U.S.
KNOW YOUR SORCERER I, II, III — For beginners (BASIC & ML)	SPECIAL OFFER	**19.95** U.S.
BRICKS — Block your opponent and wall him in (BASIC & ML)		**9.95** U.S.
GALAXIANS — A favorite with all (4K ML)	SPECIAL OFFER	**18.95** U.S.
MACHINE CODE TUTORIAL PACKAGE — 8 exercise programs (ML)	SPECIAL OFFER	**24.95** U.S.
SOUND GENERATOR & JOYSTICK CARD by Northamerican Software ..... (Assembled and tested)	SPECIAL OFFER	**19.95** U.S.

DELIVERY: Software sent out within 10 days from receipt of your order.

TERMS: **Orders are payable in U.S. funds by certified cheque or money order. C.O.D. orders are not possible. VISA and MasterCard accepted.**

NOTE: *North America only:* Postage & Handling minimum $1.50 (up to 3 items) Each additional item $0.50 extra. **DEALER ENQUIRIES WELCOME**
Overseas only: $5.00 inclusive of Registration / Insurance

SEND TO:

Northamerican Software
P.O. Box 1173, Station 'B'
Downsview, Ontario
Canada M3H 5V6

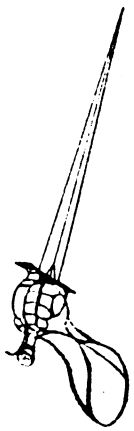
SORCERER

N o american software h

Northamerican Software is proud to announce

the newest addition to the range of programs in their library

Sword : A Sorcerer **WORD** processor by R L. HENNE Northamerican Software



This program must be rated as one of the most versatile (yet simple) Word Processors available on any computer today. It will handle and control from the simplest of printers to the most complicated programable printers available now or even in the future. This 4K machine language program will and can dump 2K of text properly formatted to the printer in an instant (for those with buffers). Output either Centronics parallel or Serial routine , included or (an area for custom serial driver routine) is provided for those wishing to use their own drivers.

The program utilises a compression technique which conserves memory space and will allow full use of memory without filling it with needless spaces. The program features the following:
(all functions are detailed in manual)

(a) Full cursor Editing
(b) Character delete and insert
(c) Move to top of text
(d) Move display window 20 lines forward
(e) Transfer blocks of text
(f) Preview text before printing
(g) Paragraph indent
(h) Margins indent left/right
(i) Auto text centering
(j) Right margin text justification
(k) Programable line length
(l) Auto page numbering
(m) Programable page length
(n) Manual/auto paging
(o) Special output mode control
(p) Two programable control initialization characters
(q) 23 - Programable control function characters output to printer
(r) Tape save & load commands (saves & loads all printer control functions and text)

The programable control function characters allow for full control of programable printers. **SWORD** was designed around the I.D.S. **460 PAPER TIGER** and all of the features (graphics excepted) that this printer is capable of. The 460 Paper Tiger sports proportional character printing, among the multitude of other programable functions.

Full details will be supplied to those who order the **SWORD** program and specify that it will be used with the I.D.S. 440 or 460.
Registration of the program to the buyer (manual included).

TOOLKIT - For the SORCERER by Paul Grimshaw through Northamerican Software

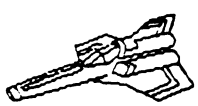
One of the most valuable programs that a Sorcerer owner could and should have in his/her collection. Once you've had a taste of the power and ease of this program, you'll probably never turn your machine on without loading this program in first, when writing programs. This program tidies up up the Exidy Basic 'bugs' and now gives you these additional features and commands.

List a,b	Renumber	10 Programable function keys
Randomize	Link	CRC errors drop back to basic not monitor
Kill a,b	Variables	Clear/home keys do not cause SYNTAX errors
Find/Find $	Move a,b,c	Character input buffer overflow gives error
Step a	Old	message and does not crash the program
Auto a,b	Close	RUN/STOP key halts execution or listing until another key is pressed

ON SCREEN EDITING - no more retyping

Registration of program to the buyer (manual included).

SUPER GRAPHIC SCRATCH PAD by H.A. Lautenbach Northamerican Software
Version 2.2 (five additional function commands added to version 2)



Create the graphic (images/pages) that you always wanted to but refrained from doing so because of the time it would have taken in basic or machine language. This program allows you to save on tape any of 5 (five) keyboards (128 characters each) and any of 7 (seven) video pages. The programing style is one of Machine language and also Basic. Learn how to manipulate graphic pages, as in Machine Language programs.

This program is a perfect match with the I.D.S. 440 or 460 to give you also Graphics Print-out of whatever is shown on the screen. A most educational program for those who wish to update their programing methods.

With manual

ORDERS ARE PAYABLE IN U.S. FUNDS ONLY BY CERTIFIED CHECK, MONEY ORDER, VISA OR MASTER CHARGE.

TO: NORTHAMERICAN SOFTWARE,
P.O.BOX 1173, STATION 'B',
DOWNSVIEW, ONTARIO, CANADA M3H 5V6

ADD $5.00 FOR REGISTRATION ON OVERSEAS ORDERS ONLY.
NORMAL POSTAGE & HANDLING ADD $1.50 (MAX 4 CASSETTES) NO C.O.D. ORDERS PLEASE.

NORTHAMERICAN SOFTWARE - LIMITED TIME OFFER

SORCERER SOFTWARE ORDER FORM

Offer valid on orders postmarked on/before - Nov.30/81

			Price U.S.$	Qty	Amount
8105	GOLF (B/ML).....	*	19.95	___	___
8103	SPIDER & ECHO plus SOUND GENERATOR and JOYSTICK CARD (Assembled & tested) (ML).....	*	39.95	___	___
8108	DISK EXTENDED BASIC 'EXBASIC' ON-SCREEN VISUAL EDITOR (ML).....	*	49.95	___	___
8109	EDOS - MICROPOLIS OPERATING SYSTEM (ML).....		59.95	___	___
8110	ZETU - CASSETTE-BASED Z80 ASSEMBLER (ML).....	*	29.95	___	___
8101	TOOLKIT - THE ON-SCREEN EDITOR FOR BASIC ROMPAC(ML)	*	29.95	___	___
8005	SWORD - SORCERER WORD PROCESSOR (ML).....	*	29.95	___	___
8115	SUPER GRAPHIC SCRATCH PAD VER.2.2 for IDS 460G Paper Tiger Printer and other graphic dot matrix printers (B/ML).....	*	19.95	___	___
7912,13,14	KNOW YOUR SORCERER I,II,III for new Sorcerer owners only (B/ML).....	*	19.95	___	___
8106	BRICKS(B/ML).....		9.95	___	___
8111	GALAXIANS (ML).....	*	18.95	___	___
8112	MACHINE CODE TUTORIAL PKG.(ML).....	*	24.95	___	___
8104	SOUND GENERATOR & JOYSTICK CARD(Assembled & tested)	*	19.95	___	___
8102	SPIDER & ECHO (ML).....		24.95	___	___
DP7310	EXBASIC - Full disk extended BASIC.....		325.00	___	___
DP7260	Z80 Disk Development System		125.00	___	___
DP7221	EXIDY precision PRINT driver routines.....		125.00	___	___
DP7220	WPP PAC to disk conversions.....		99.00	___	___
DP7272	EXIDY ROM PAC BASIC to Extended 19K BASIC.....		49.00	___	___
DP7271	EXIDY 19K BASIC to EXBASIC disk BASIC.....		99.00	___	___
DP7100	MICROHOME programs (Home utility programs).....		30.00	___	___
DP3003	SORCERY cassette programs (Exidy's competition)..		50.00	___	___
------	SPELLBINDER - Word Processor.....		375.00	___	___
------	SPELLGUARD - Dictionary program for SPELLBINDER..		285.00	___	___

NOTE: N. America only: Postage & Handling minimum $1.50 (up to 3 items). Each additional item 50¢ extra ___

Overseas only: Postage & Handling inclusive of Registration/Insurance $5.00 ___

TERMS: ORDERS ARE PAYABLE IN U.S FUNDS BY CERTIFIED CHEQUE OR MONEY ORDER. C.O.D. ORDERS ARE NOT POSSIBLE. VISA AND MASTER CARD ARE ACCEPTED. MAIL TO: TOTAL ___

NORTHAMERICAN SOFTWARE, P.O. BOX 1173, STATION 'B', DOWNSVIEW, ONTARIO, CANADA M3H 5V6

VISA: Card No. _____ Expiration Date _____

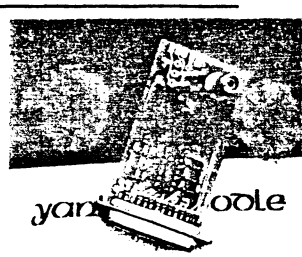
MASTER CARD: Card No. _____ Expiration Date _____

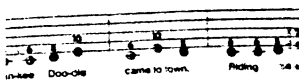
SHIP TO: Name _____

Address _____

City,State,Zipcode _____

Signature: _____ Date: _____





* DENOTES SPECIAL OFFER

JOYSTICK CARD

SORCERER USERS' GROUP (TORONTO)

Membership Application Form — Covering Jan. to Dec.1981

Membership to the group is not restricted to the TORONTO area. All persons willing to participate are invited to join.

As a member of the Sorcerer Users' Group (Toronto), I enclose the annual membership fee and agree to the following Terms.

1. That I will not, without the authorization of the board of directors, represent myself or take any action as agent, or representative or become spokesperson of the group.

2. That I will not use any software obtained from the SUGT library for any commercial purpose or financial gain. The library shall be available to me should I wish to obtain programs donated by other members. These programs shall not be distributed without the owners consent and/or the consent of the board of directors.

3. That I have the right to vote for the officers and directors of the organization at the annual general meeting.

4. That any breach of the above conditions and any other restrictions that the board of directors may invoke in the future on my part may result in suspension or termination of my membership without refund.

Annual Membership Rates : (Jan - Dec):Effective May 1981

Canadian - $15.00 Cdn - U.S.& Foreign $15.00 (U.S Funds) PLUS $8.00 Postage

Payable to - SORCERER USERS' GROUP (TORONTO) - by Cheque or Money Orders.

The SUGT program library is available to all members in the following manner.

You may send $6.00 for each volume as they become available and we shall supply the cassette/s. Program cassettes shall be sent via Air Mail.

All issues of PORT FE shall be mailed first class, in the case of non local issues, they are mailed via Air Mail. Past issues of PORT FE are only available for the current calendar year. Contact the editor, he will advise the amount of payment for previous issues.

NAME(print):.....
ADDRESS:.....
CITY:.....
POSTAL CODE:.....
TELEPHONE: Res..... Bus.....

Payments enclosed (membership):..... Library tape/s.....

Signature:.....

Please list the type of equipment you are using etc...
Sorcerer size: 8... 16... 32... 48... other..... S100... Graph board.....
Disk system - Micropolis..... Discus.... Exidy.... other... Size.....
Other Equipment .....
.....

If you belong to any other Sorcerer Users' Group please list it below.
.....