ZBasic™ Newsletter **Winter 1987/1988**

Zedcor did great this year at the MacWorld Expo in San Francisco. ZBasic sold very well.

The big hit of the show was Andy's latest program DeskPaint.

Here's Karen Moesh, Zedcor's General Manager, during a more relaxing moment.

This Mac show was the biggest ever with over 60,000 people.

Mac ZBasic 4.01 Now Shipping

Incredible new HyperText HELP system makes ZBasic easier to use than ever.

The newest version of ZBasic for the Macintosh has finally shipped. It's packed with many of the powerful new features you've asked for, including:

The Integrated HyperText Help system brings to users a 378K help file with virtually the entire manual on line. An alphabetical index appears when you first invoke the system. Within definitions are cross-referenced keywords in outlined format. Just double click an outlined word and get an instant definition for that keyword. An incredible time saver for expert programmers and a great learning tool for beginners.

Macintosh page 12

ProDOS ZBasic Selling Well

The ProDOS version of ZBasic is selling better than ever. There must be lots of Apple II affectionados out there. The latest Apple II versions are:

Apple // DOS 3.3 ZBasic
Version 3.2 3/18/87

Apple // ProDOS ZBasic
Version 4.1 1/15/88

Apple page 18

Plenty of Useful Examples in this Super Issue

We have an example of doing Bezier curves in two byte integer and more examples of the popular "ShowTyme" graphics program.

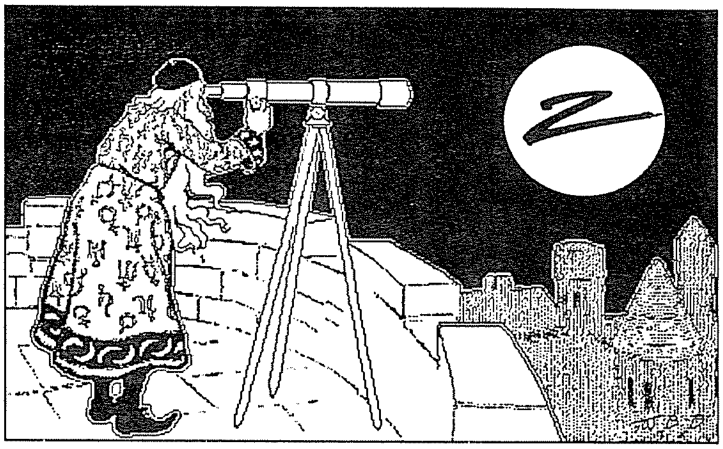
This is a great issue packed with lots of ideas.

Bezier page 8

IBM ZBasic 4.02 Passes the Test

This quarter we have quite a few useful subroutines. Including routines for creating TSR's, reading Cursor positions...

IBM page 22



INSIDE

Example Programs

Michael A. Gariepy
Editor in Chief
Advertising

Andrew R. Gariepy
Technical Editor

ZEDCOR, Inc.

PRESIDENT
Andrew R. Gariepy

CHAIRMAN
Michael A. Gariepy

Winter, 1988

"Z" is published quarterly by Zedcor, Inc., 4500 E. Speedway, Suite 22, Tucson, AZ 85712-5305. (602) 881-8101. Support line: (602) 795-3996. Subscription rates are $19.95 a year, $37.00 for two years (add $5 for overseas and Canadian orders).

To subscribe by phone call toll free: 1-800-482-4567.

Please send address changes to the Zedcor address above.

Advertising offices are at 4500 E. Speedway, Suite 22, Tucson, AZ 85712-5305. 1-800-482-4567

Programs included in this journal may be used for non-commercial purposes without restriction.

Newsletter created with a pre-release version of FullWrite™ and Zedcor's new DeskPaint™.

Letters to the Editor

Gripes, problems or compliments? Send letters to: Editor, 4500 E. Speedway #22, Tucson, AZ 85712.

To the Editor,

In your spring-summer issue(1987) you advertised having routines in the next issue for finding the elapsed time in days for any week. Also the "Support Blues" comic strip and the Support Notes" would be in the following issue. We'll I can't find any of those articles. Is this a case of Vapor-Paper?

Mark Shepard
1773 Creek Dr.
San Jose, CA 95125

Well Mark, I must say I'm a bit sorry about that. The Support Blues died because I just don't have the time to draw one every quarter. I am willing to pay $25 for any good comic strips you folks may come up with. The routines for Day of the week will be in the next issue (promise). Support notes went the way of our last support guy. If you haven't noticed, Greg, Andy and I have been doing the support lately. We figured it gives us a better chance to find and fix the bugs quickly. We work in shifts; Greg from 12 to 1:30, Andy 1:30 to 3:00, Mike 3:00 to 4:30. The phones aren't nearly as busy as they used to be. We must be answering questions to your satisfaction.

To the Editor,

Many thanks for sending the new debugged version of ZBasic. It seems to be OK, without the bugs that were in the original.

The candor in admitting the bugs and in replacing version 4.0 at no charge is a first time experience for me (I've spent close to $1000 for a program that was full of bugs and the author wouldn't even admit to the problems). I've told many of my computernik friends about what you have done and they too feel that it is an unusual thing and absolutely fabulous. Hopefully you will get some ZBasic converts, as I talk up the language and now the company.

Richard B. Mesirov
1700 Riverview Rd.
Gladwyn, PA 19035

We appreciate the feedback. As embarrassing as it is to admit mistakes we believe you have a right to know. When you know about problems you can avoid them and/or write code around them in most cases.

We have taken new steps to insure that future releases don't repeat past problems. In the meantime, we'll continue to work at releasing newer versions of ZBasic that incorporate the features you ask for.

As the guys on TV say; "Thank you for your support."

To the Editor,

Enclosed you will find my ZBasic disk for upgrading.

On the rest of the letter; First your newsletter needs more articles and information in it. Before you respond with a "We're Doing All We can" , consider paying your customers for articles. It takes a considerable amount of time to put together an article with supporting source code. There is a lot of testing and notes that need to be compiled.

David Postler
P.O. Box 1086
Elgin, IL 60121-1086

I have to agree with you. Writing an article is a lot of work. Thanks for your ideas. See box.

To the Editor,

I have been using your ZBasic for the past 16 months and find much to commend it (CP/M version). So I thought I would write a response to your call for suggestions.

The newsletter should be oriented, first, to those materials which apply to all versions. All articles on the first page would be of this type. Second would be a section reporting new additions to some version which makes it more compatible with another version. Please remember that the primary reason many of us bought ZBasic was because it is mostly compatible across different machines.

I have two suggestions for additions to the language. These are a ECHO and INCLUDE.

ECHO# filenumber would echo text sent to a disk file to the screen. The statement ECHO OFF# filenumber would turn echo off.

The INCLUDE "filename" statement would bring in source code at compile time and compile it. This way you could use subroutines as they were needed.

Another feature I miss from other programming languages is a way to show a directory at runtime.

Richard L. Gorsuch, Ph.D.
Professor of Psychology
Director of Research
Fuller Theological Seminary
180 N. Oakland Ave.
Pasadena, CA 91182

Thank you for your suggestions. This new edition of the newsletter should be a little more like you requested. One thing many of you should realize is that just because a program example is in another section doesn't mean it can't be moved to your system with some adjustments. A good example is the Bezier curve function in this issue. It was converted by Robert Strong to a Two Byte INTEGER format from four byte LongInteger format as it first appeared in the newsletter.

We are looking at the INCLUDE statement as it would provide many benefits.

Directory routines are available for most machines. I've sent you a version for CP/M machines.

The ECHO statement does not seem that important in my opinion. On CP/M systems in particular there is extreme memory considerations. To add too many extra statements to ZBasic would severely limit program size. In this respect we have decided to keep commands to a minimum when other commands can do the same thing.

$ Request for Articles $

The "Z" newsletter is open to submissions for articles pertaining to ZBasic. This includes articles about:

- ☐ Subroutines
- ☐ Graphics examples
- ☐ Cartoons
- ☐ File handling
- ☐ Games
- ☐ Programming Ideas
- ☐ Editor routines
- ☐ Fiction/Non Fiction
- ☐ Math puzzles
- ☐ Educational programs

We'll pay $20 to $100 for each article that is accepted (based on size and importance). Program submissions will remain in the public domain for all to use.

Articles and Program submissions must be text files (ASCII) on a diskette formatted for IBM PC (360K), Apple II ProDOS or Macintosh.

BEFORE SENDING ANYTHING call Mike Gariepy at 800-482-4567 to discuss submission. This is done to ascertain the desirability of the submission and the amount to be paid.

To the Editor,

Will ZBasic programs ever run within Windows or OS/2? I have read that OS/2 compatible programs must be written in "C". Could you make a clear "policy statement" on this.

I also read about BASIC-to-C interfacing as a ray of hope for BASIC programmers.

Ronald W. Bryan, M.D.
9330 Park West Blvd. #103
Knoxville, TN 37923

It is hard to make a clear policy statement about OS/2 yet. With the Macintosh the conversion to the Mac Environment was relatively straight-forward but required a considerable amount of work and learning. We are still learning about the Mac ROM.

OS/2 is relatively new and we have to see how widely accepted it will be. There seems to be some hesitancy on the part of PC owners to move over to it. We are, however, "Cramming" on these environments and should be able to tell in the next issue whether an OS/2 ZBasic will be forthcoming.

As far as "C" being required to program in OS/2, I don't see that as being the case. "Inside Macintosh" is entirely Pascal, yet ZBasic provides a complete interface to the "Mac Environment". The biggest problems we encountered were translating the Pascal Jargon to BASIC.

Down with C and Pascal!
LONG LIVE BASIC!

To the Editor,

I have a few suggestions for future versions of ZBasic. I cannot begin to count how many times I have written a program, run it, and realized I forgot to set TRONB. I was just wondering if you could make this the default.

On a different note, a friend of mine has recently put up a BBS and has made me the SYSOP of a ZBasic forum (see below). We call the board Compu-Plane and it is running 24 hours a day on a PS/2 model 80 under OS/2. We were wondering if you could put a small note in your newsletter.

Daniel Neuwirth.

As you request Dan. As far as the TRONB nuisance I have to agree with you. I'll talk to Andrew about that.

To the Editor,

I have installed a new BBS just for ZBasic programmers called ZBBS.

The ZBBS is operated by and for all ZBasic programmers, all versions. We are dedicated to the advanced use of ZBasic.

Walt Keifer
819 Ave. F ne.
WinterHaven Florida, 33880

It's great to see the kind of support ZBasic is getting! I hope you all support these guys.

Thanks Walt and Dan. see BBS info below in the box.

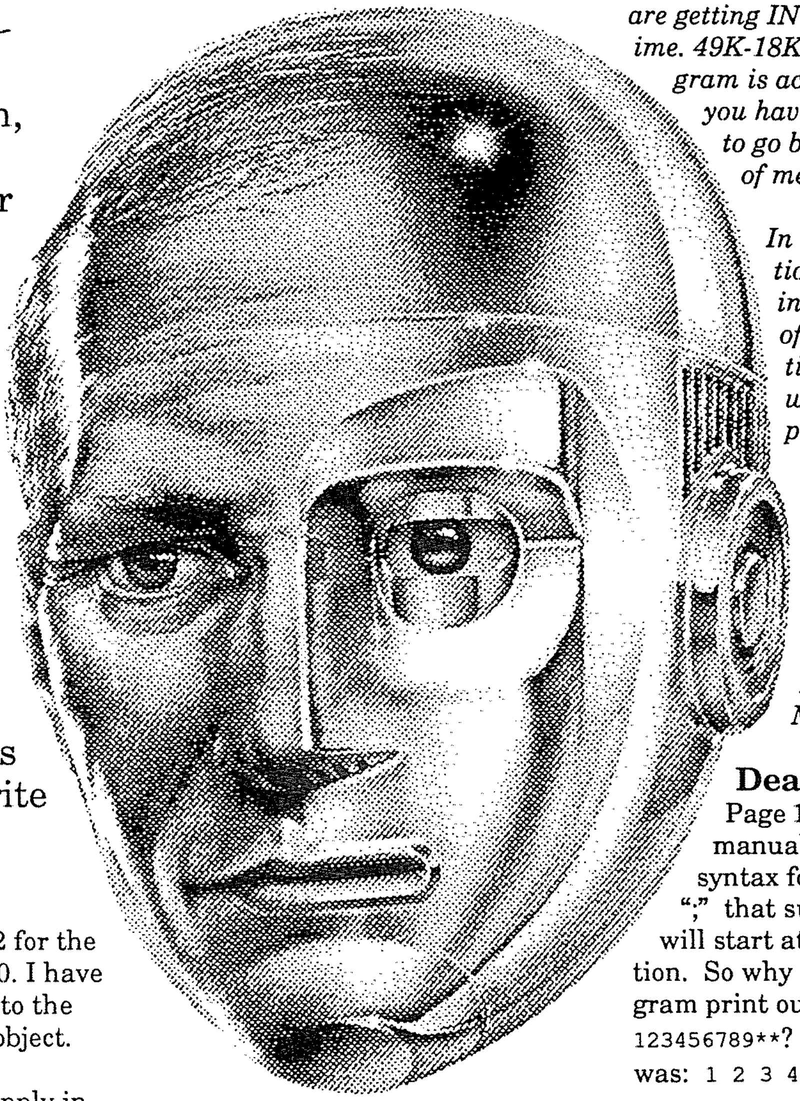
☎ ZBasic Bulletin Boards ☎

For those of you with modems that want more information about ZBasic, try out these two new BBS'.

West Coast	**East Coast**
Los Angeles	Florida
Compu-Plane	ZBBS (ZBasic BBS)
818-843-4874	813-299-5694
voice 818-789-0304	voice 813-299-5650
24 Hours	24 hours
300/1200/2400 baud	300/1200 baud
Dan Neuwirth SYSOP	Walt Keifer SYSOP
Log on for more info.	Operated by and for ZBasic Programmers. $15 year.

Dr. Z

Dr. Z is half-man, half-computer; he'll answer your questions about programming in ZBasic like no other being in the Universe. His brain is a bionic AI device that thinks in ZBasic, making him the most qualified entity in existence to answer questions about your favorite BASIC.



Dear Dr. Z,
I am using version 4.02 for the PC on a Tandy SX 1000. I have two questions relating to the 48K memory limit for object.

1. Does the limit still apply in the 4.02 version?

2. What symptoms or problems occur when a program gets above the limit.

I have a program which measures above 49K without a problem. May I please have some detailed information on this subject.

William Stevenson
513 Lincoln Ave.
Winnetka, IL 60093

Dear Mr. Stevenson,
Your concerns about memory considerations are legitimate. If you look at the Memory Map on page A-16 of the ZBasic manual (in the MSDOS appendix), you will see the maximum size of the final object code is 64K minus any runtime code. Since the Runtime code is about 18K this leaves less than 50K for compiled object code.

In response to question #1: Yes. Approximately. But the 49K you are getting INCLUDES the runtime. 49K-18K means your program is actually only 31K so you have another 15-16K to go before running out of memory.

In response to question #2: ZBasic will inform you if an out-of- memory condition exists. This will happen at compile time.

Note that variables are not included in the Object code. They take up other areas of memory. See the MSDOS Memory Map.

Dear Dr. Z,
Page 126 of the ZBasic manual says under print syntax for the semi-colon ";" that subsequent prints will start at the cursor position. So why doesn't this program print out the line: `123456789**`? The actual output was: `1 2 3 4 5 6 7 8 9 **`

(he owns the ProDOS version of ZBasic 4.02) (Program below)

```
FOR I=0 TO 9
  PRINT I;
NEXT I
PRINT"**"
END
```

Rod Robichaux
8471 Everett Way #D
Arvada, CO 80005-2354

continued next page...

Dr. Z continued...

Dear Mr. Robichaux,

When ZBasic prints the value of a numeric variable, it always reserves one character position to the left of the character for the "Sign" of the number. If the number is negative, ZBasic prints a "-". If the sign is positive a space is printed. This character position is also a part of any string that is produced with the STR$ function.

ZBasic also automatically prints a trailing space when printing a numeric variable. This trailing space is NOT included when converting a numeric to a string with STR$.

If you don't want the spaces inserted by ZBasic, your only resource is to convert the number into a string and then strip the leading space off. Example:

```
FOR I=0 TO 9
  A$=STR$(I)
  PRINT RIGHT$(A$,LEN(A$)-1);
NEXT I
PRINT "**"
END
```

Returns: "`123456789**`".

Voila!

Dear Dr. Z,

[I am] Very satisfied with your improved version 4.02 (IBM PC).

Question; How do you use USR3 to input a string with full ability to back-up [with the delete key], but allowing the user to press the ESC key to escape the routine, at least for the first key pressed.USR3 detects the next key fine but the input routine starts over rather than including the first key pressed.

Scott Stalheim
Box 344
Augusta, WI 54722

Dear Mr. Stalheim,

A simpler way to input keys is with the regular INKEY$ statement although USR3 could also be used.

The following function will allow you to create an input routine that will trap your keys and display them on the screen at the position you tell it (x,y). You can also use the backspace key to delete characters (see box).

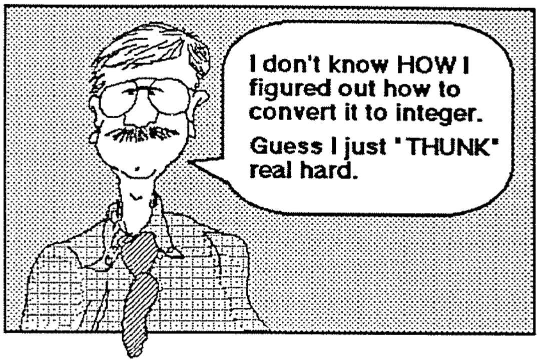
Things that could be added are a blinking cursor routine and more filters for undesirable keys. You could also check for ALT, CTRL and Function keys using USR3.

That's all for this newsletter. Keep those letters coming.

```
'Getkey function (example)
CLS
LONG FN Getkey$(x,y,len)
  PRINT@(x,y);"_";REM unblinking cursor
  str$="": Term=0:  REM keys pressed
  DO
    DO
      key$=INKEY$          REM Get a key
    UNTIL LEN(key$)
    LONG IF ASC(key$)>31   REM Check if ctrl char.
      str$=str$+key$       REM add to other keys
    XELSE
      LONG IF ASC(key$)=8  REM Check if backspace
        str$=LEFT$(str$,LEN(str$)-1)
      END IF
    END IF
    PRINT@(x,y);           REM Print str$ at x,y
    CLS LINE               REM clear line of old
    PRINT str$;"_";        REM print string
    SELECT CASE ASC(key$)
      CASE 13: Term=1    ' Return key
      CASE 27: Term=1    ' ESC key
      CASE 10: Term=1    ' Linefeed key
    END SELECT
  UNTIL Term:    ' Wait til a terminating key is pressed
  REM Exit loop if "RETURN" pressed or len reached
END FN=str$
:
INPUT"Input x,y and len";x,y,len
IF y=0 OR y=0 OR len=0 THEN END
I$=FN Getkey$(x,y,len)
CLS: PRINT:PRINT"The string returned:"; I$
END
```

User Conceives a new Bezier Curve Routine

User sees LongInteger version in previous Newsletter and comes up with a powerful (and Fast) two byte integer alternative.


I don't know HOW I figured out how to convert it to integer.
Guess I just "THUNK" real hard.

Thanks to Robert Strong of Charlottesville, VA for this great Bezier curve routine.

The original appeared in the Fall Newsletter and required four byte Integers as used in the Macintosh version of ZBasic.

This program will run on any version of ZBasic including the IBM PC, Apple II ProDOS and Z80 versions.

Robert wrote; "While this program actually uses some floating point computation, the calculation has been restructured to be faster. The main loop in your LongInteger program executed 100 times, containing 17 floating point multiplications while my version has only 6. Of course the LongInteger version could be improved by similar restructuring."

This program is quite fast. In fact it is faster than our original. Thanks Robert!

```
'  TWO BYTE INTEGER Bezier Curve Example
'  Submitted by Robert Strong, Charlottesville, VA
'
TRON B
DEFSNG N-Z
DIM X(3), Y(3), U(3), V(3)
' MODE=7: COLOR=15: Set for your system
K=100:TD=1/K
CLS
FOR I=0 TO 3
  X(I)=RND(1024)-1: Y(I)=RND(768)-1
NEXT
FOR I=0 TO 3
  CIRCLE X(I),Y(I),10
  CIRCLE X(I),Y(I),2
NEXT
PLOT X(1),Y(1) TO X(2),Y(2)
U(0)=X(3)-3*X(2)+3*X(1)-X(0)
V(0)=Y(3)-3*Y(2)+3*Y(1)-Y(0)
U(1)=3*X(2)-6*X(1) +3*X(0)  :  V(1)=3*Y(2)-6*Y(1)+3*Y(0)
U(2)=3*X(1)-3*X(0)          :  V(2)=3*Y(1)-3*Y(0)
U(3)=X(0)+0.5               :  V(3)=Y(0)+0.5
T=0:  OX=X(0):  OY=Y(0)
PLOT OX,OY
FOR I=1 TO K
  X=U(0): Y=V(0): T=T+TD
  FOR J=1 TO 3
    X=T*X+U(J): Y=T*Y+V(J)
  NEXT
  LONG IF ABS(X-OX)>1 OR ABS(Y-OY)>1
    PLOT TO X,Y
    OX=X: OY=Y
  END IF
NEXT
PLOT TO X(3), Y(3)
:
RUN
```

A Look At Heap Sort

The question of which sort is faster comes up again.

You decide.

This Heap Sort was submitted by Pat Wojtkiewicz from Shreveport, LA. It is another sort type that we don't currently include on our example disks.

Pat says; "I have noticed you seem to go for the Quick Sort for high speed sorting; however, many people overlook the binary Heap Sort for many applications. It is an O(nLOGn) sort and its best and worst cases are nearly the same. The most impressive point of the Heap Sort is that its memory requirements are O(n), unlike Quick Sort which requires a couple of additional arrays and stacks (although not substantive. Ed.) I enclose a copy of it as a LONG FN".

Thanks Pat.

We compared the sort on a Mac and found the Quick Sort to be significantly faster (10x). Perhaps someone has ideas about optimizing this sort?

```
'            HEAPSORT example program
TRON B   REM DIM a(n) where n=number of elements to sort
DIM a(501): n=500
:
LONG FN HEAPSort(nHEAP)
  REM Transforms the elements into a heap
  :
  FOR iHEAP=INT(nHEAP /2) TO 1 STEP -1
    iADJUST=iHEAP: nADJUST=nHEAP
    GOSUB "ADJUST"
  NEXT iHEAP
  :
  REM The Heap will place the largest value in array position 1
  REM each time the ADJUST is invoked. This loop swaps it to
  REM the nHEAPSORT-iHEAP position
  :
  FOR iHEAP=nHEAP TO 2 STEP -1
    SWAP a(1), a(iHEAP)
    iADJUST=1: nADJUST=iHEAP-1: GOSUB "ADJUST"
  NEXT iHEAP
  :
  GOTO "EXIT HEAPSORT":  REM Finished sorting here so exit
:
"ADJUST"
  jADJUST=iADJUST *2
  itemADJUST=a(iADJUST)
  WHILE jADJUST<=nADJUST
    LONG IF (jADJUST < nADJUST) AND (a(jADJUST) < a(jADJUST+1))
      jADJUST=jADJUST+1
    END IF
    LONG IF itemADJUST >= a(jADJUST)
      GOTO "EXIT ADJUST"
    XELSE
       a(INT(jADJUST/2))= a(jADJUST)
       jADJUST = jADJUST *2
    END IF
  WEND
  :
"EXIT ADJUST"
  a(INT(jADJUST/2))=itemADJUST
  RETURN
  :
"EXIT HEAPSORT"
END FN=0
:
REM Program Starts here
FOR X=0 TO n
   a(X)=RND(2000):' Create random array for speed test.
NEXT
CLS: PRINT%(490,380);"Starting Sort of:";n;" items now..."
START#=TIMER: SOUND 100,100: REM Use TIME$ w/Apple II & Z80
jj=FN HEAPSort (n)
FOR X=n-10 TO n
  PRINT a(X):' Print last 10 to ascertain good sort
NEXT
LAST#=TIMER:: SOUND 100,100
PRINT: PRINT "Sort took ";LAST#-START#;" Seconds"
END
```

ShowTyme Revisited

The ShowTyme Random Graphics shape generator in our summer issue was very popular.

We had several people submit modified versions of it that create some complicated and beautiful shapes.

These programs are added for your amusement.

Like the original ShowTyme program these demonstrate the speed of the integer Sine and Cosine routines in USR 8 and USR 9.

Thanks to Charles Stone and Robert Strong for their submissions.

```
'  Submitted by Charles H. Stone
' Portland, OR
DEFSNG R
CLS
FOR S = 1 TO 127
  CLS: MODE 12 : ' Set mode for hi-res cards
  FIN=256*S
  PRINT%(0,730);"Step";S;
  FOR A=0 TO FIN STEP S
    TRON X
    GOSUB "FORM"
    X=(USR 9(A))*R+600
    Y=(USR 8(A))*R+375
    IF A <> 0 THEN PLOT X1,Y TO X,Y1
    X1=X: Y1=Y
  NEXT A: PRINT:PRINT"    ";
DO
UNTIL LEN(INKEY$)
NEXT S

"FORM"
R=USR 9(USR 8(A))
R=R/200:RETURN
END
```

```
' This program was submitted by Robert Strong
' Charlottesville, VA
CLS:  'MODE=7
FOR N=4 TO 255 STEP 4:' Try changing the steps
  CLS
  FOR D=0 TO 255 STEP 16
  'Add COLOR=RND(15)+1 if you have a color machine
  PRINT%(0,30); : CLS LINE
  PRINT"N=";N"  ";" D=";D;"  "; :T=0:C=0
  WHILE C<256
    TRONX
    A=T: X=(N*A) MOD 256: R=USR 8(X)/2
    PLOT 511+((R*USR 9(A))/128), 383-((R*USR 8(A))/128)
    DO
      A=(A+D) MOD 256
      X=(N*A) MOD 256:  R=USR 8(X) /2
      PLOT TO 511+((R* USR 8(A))/128), 383-((R* USR
8(A))/128)
      C=C+1
    UNTIL A=T
    T=T+1
  WEND
NEXT D, N
SOUND 100,100: SOUND 500,500
DO
  A$=INKEY$
UNTIL LEN(A$)
END
```

The Keyboard Prayer

Our Program
 Who art in memory,
 Hello be thy name.

Thy operating system come,
 Thy commands will be done,
 at the pointer as is on the screen.

Give us this day,
 Our daily Data,
 and forgive us our I/O errors
 As we forgive those
 whose logic circuits are faulty.

Lead us not into frustration,
 and deliver us from power surges.

For thine is the algorithm,
 the application,
 and the solution.

Looping forever and ever.

RETURN

Precision π

Robert Strong has submitted an interesting version of Pi for your mathematical Diversion.

"I ran across something cute which you might enjoy. Recently there have been several mathematical articles about a new way of computing Pi (π) which is supposed to be real fast. It gives fantastically large numbers of digits in no time, but you need a language that carries lots of digits in its arithmetic.

I decided to try this program out using my Apple ProDOS version of ZBasic. I set the configuration to 54 digits of accuracy for Double and Scientific precision and ran my little program.

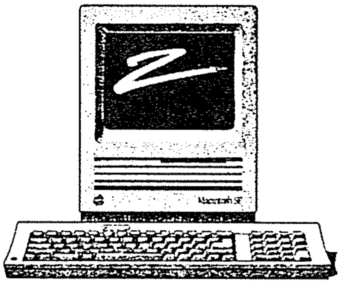
The first 53 digits are correct the last digit is a 2 instead of a 1. Even carrying 54 digits one doesn't expect absolute accuracy due to inevitable roundoff.

Here we have Square roots, Multiplications, Divisions, and all the rest -- and the answer is damn near perfect. It is really pretty."

Thanks Robert.

```
'   Example of High Precision Pi
'   Submitted by Robert Strong, Charlottesville, VA
'   Returns Pi to the maximum precision set
'   Under "Configure" for Double Precision
'
TRON B
DEFDBL M-Z
X=1/SQR(2)
Y=0
P=2
FOR II=1 TO 9
  PRINT P
  S=SQR(X): R=1/S
  P=P*(X+1)/(Y+1)
  Y=(Y*S+R)/(Y+1)
  X=(R+S)/2
NEXT
:
END
```

Macintosh section continued from front page...



Other important new features include COLOR Support for Mac II, MultiFinder™ and Switcher™ support, a SIZE resource has been added to ZBasic and to applications created with ZBasic. This resource passes the memory required by an application. It may be changed with ResEdit™. The editor has been cleaned up (no more bombs in the new ROM and new system). It will even save your window positions and text sizes if you want.

How to Upgrade to 4.01

The upgrade to version 4.01 is free to all owners of version 4.0. To get your free upgrade send in your master disk to ZEDCOR.

Still Have Version 3.0?

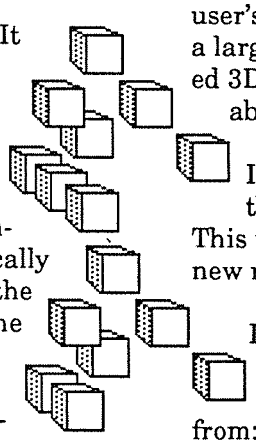
Registered Owners of ZBasic 3.0x may still upgrade to version 4.01 for $39.95 plus $5.00 shipping. This upgrade includes the new 768 page manual. Send us your original ZBasic 3.0x disk and a check for $44.95.

Incredible 3D Life Program written in ZBasic gets Write-Up in Scientific American

Scientific American article details 3D version of Conway's Life program in ZBasic. Program does animated 3D 'Boxes' instead of 2D pixel images like old version of Life.

This incredible program does "Conway's Life" in 3D instead of two dimensions like the old version.

Each 3D cell has 27 neighbors (8 in 2D). It uses "3D Cubes" to represent each cell.



Depending on the relationship of one cell to it's many neighbors, it is mathematically determined whether the form will survive to the next generation.

The life forms are created and move on the Mac screen in real time. Life forms settle in to a set pattern after several generations of existence or eventually grow into an unstable condition (the interrelationships of cells) and perish. It is fascinating to watch this animated display.

The article was in the February, 1987 issue of Scientific American. The Life program is available as a stand-alone application. Cost is $28 + $2 Shipping [$30 total]). The price includes a user's guide, 400K diskette, and a large variety of already created 3D Life shapes. Execution is about .5 to 3 generations/ sec.

I implore Mr. Bays to make the "Source Code" available. This would spur some incredible new releases.

I had a chance to try the program and was quite favorably impressed. Order from:

Carter Bays
Dept. of Computer Science
LeConte Bldg.
University of South Carolina
Columbia, SC 29208

Important Notes on Macintosh ZBasic 4.01

Making the Help System Work with MultiFinder

New HyperText Help requires special installation to work under MultiFinder.

To make the new HELP Desk Accessory recognize the ZBasic™ HELP file when you're using ZBasic under MultiFinder, you'll need to install the DA directly into ZBasic using Font/DA Mover.

The procedure is identical to installing it into the System except that when you are in Font/DA mover you hold in the "Option and Command keys" and select ZBasic instead of System. This installs the Help DA directly into ZBasic and allows it to recognize the help file correctly.

Nested Event Trapping Differs from Version 4.0

Old 4.0 programs missing Events in 4.01? Here's an easy way to solve the problem.

If you encounter problems with your old 4.0 programs missing events here's the reason.

When doing an ON Event GOSUB in version 4.0 and the user failed to turn off events at the routine containing the event handing, an inevitable System Error 28 would occur. To fix this, version 4.01 disables event trapping as soon as an Event takes place with an ON Event GOSUB. Event trapping is not re-enabled until a RETURN is encountered.

The big problem occurs in routines where you go to another event loop based on an event that took place at a previous event loop. In this situation events will not be trapped. If you press buttons or select menus or edit fields nothing will happen. This is because the program is waiting for a RETURN to reactivate event trapping.

Fortunately you don't have to totally rewrite your programs to fix this. Since the events are handled based on information in a global memory location, you can simply poke a value into that location to solve the problem. The solution is to put this statement:

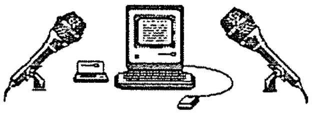
```
POKE PEEKLONG(&904)-&907,0
```

in front of subsequent event loops. Here's a simple example:

```
Program start...

ON DIALOG GOSUB "Dia"
ON MENU GOSUB "Men"

"Main Event Loop"
DIALOG ON: MOUSE ON
DO
UNTIL Loop
DIALOG OFF: MENU OFF

program continues..

"Second Event Loop"
POKE PEEKLONG(&904)-&907,0
ON DIALOG GOSUB "Dia"
ON MENU GOSUB "Men"
DIALOG ON: MOUSE ON
DO
UNTIL Loop
DIALOG OFF: MENU OFF
```

Note: Use this statement at the start of each subsequent event loop.

It doesn't hurt to use it whenever you like.



Modify Example Sound Program to obtain Asynchronous Sound Effects

Create Special Sound Effects for your Applications that can play in the background. This is a modification to "PlaySound.BAS" in the Program Example Folder.

We've included a program in the Sound example folder of version 4.01 to play "beep" sounds.

This is a way for you to add digitized sounds to your programs. This can come in handy for Games, Educational and Business applications.

For those of you that don't have "Beep" sounds try looking in the Public Domain "Sound Files" and "HyperCard Sound Effects Files" on GEnie or Compuserve.

Other sources are Public Domain software houses like EduComp and BudgetBytes.

Unfortunately the program didn't play in the background and a program would grind to a halt until the sound was finished playing.

```
' This is a program that plays digitized "CheapBeep"
' Start-up Beep Sound files. Just run the program and
' select a valid start-up sound or cancel to exit


DO
  F$=FILES$(1,"FSSD",,V%)      : ' OPEN A SOUND TYPE FILE
  IF F$="" THEN END            : ' CANCELED FILES$
  TEXT 0,12 : CLS              : ' SET TEXT SIZE/CLEAR SCREEN
  OPEN "I",1,F$,1,V%           : ' OPEN A SOUND TYPE FILE?
  L&=LOF(1)                    : ' GET THE LENGTH OF FILE
  IF L&=0  THEN STOP           : ' IF NO LENGTH THEN STOP
  P&=FN NEWPTR(L&+6)           : ' ROOM FOR SOUND RECORD
  IF P&=0 THEN STOP            : ' COULD NOT GET THE MEMORY
  READ FILE #1,P&+6,L&         : ' READ INTO MEMORY
  CLOSE 1                      : ' CLOSE THE FILE
  PRINT "File Name: ";F$       : ' SHOW FILE NAME
  PRINT "Length:";L&;"bytes"
  POKE WORD P&,0               : ' FREE FORM SYNTH
  POKE WORD P&+2,0             : ' RATE OF PLAY (INTEGER)
  POKE WORD P&+4,32768         : ' RATE OF PLAY (FRACTIONAL)
  A&=VARPTR(A$)                : ' GET A PARAMETER BLOCK
  POKE LONG A&+12,0            : ' NO COMPLETION ROUTINE
  POKE WORD A&+24,-4           : ' SOUND DRIVER REF-NUMBER
  POKE LONG A&+32,P&           : ' POINTER TO SOUND RECORD
  POKE LONG A&+36,L&           : ' LENGTH  OF SOUND RECORD

_____( Add the following 4 lines to example )_________
  MACHLG A$, &A403             : ` Asynchronous Write
  DO
    PRINT ".";                 : ` Talking and printing at the
  UNTIL PEEKWORD(A&+16)=0      : ` same time!!
___________________________________

  X=FN WRITE(A&)               : ' PLAY THE SOUND
  X=FN DISPOSPTR(P&)           : ' DISPOSE OF SOUND MEMORY
UNTIL LOOP                     : ' Endless Loop

END
```

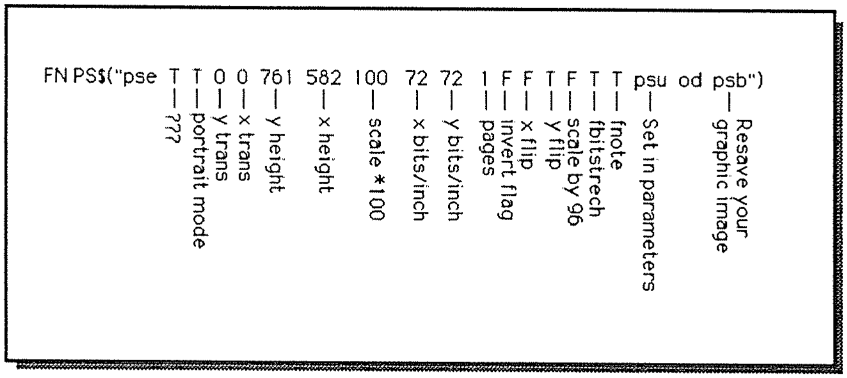
Add the changes noted in this example program to allow these sounds to play in the background. You will still realize about a 20% speed degradation.

Digitized sounds really add a lot to a program.

New ideas for using PostScript

Ken Jenkins sent us some enlightening observations about using PostScript.

Here's some information pertaining to the PostScript example you published in the new ZBasic manual (4th edition). I thought it might be useful to other users who are attempting to write PostScript output for ZBasic programs. I would be very interested if you come across an example of how to send PS through the AppleTalk network without encountering the Print Manager at all (and subsequent download of Laser Prep). I don't have the knowledge of AppleTalk needed to write such a routine. If you know of one, please let me know. (*Anyone out there? ed*) Another problem is the 32K limit.



```
FN PS$("pse T T 0 0 761 582 100 72 72 1 F F T F T T psu od psb")
```

On page E-25 in the ZBasic manual is an example of how you can use the PICTURE statement to send PostScript code directly to the LaserWriter (or other PostScript devices).

Unfortunately since Apple's print manager is used to do this, the code doesn't quite go "directly" to the LaserWriter. Apple's LaserWriter drivers are called up, and output to the LaserWriter is done through these drivers. If Laser Prep has not been downloaded to the printer already, it is downloaded at this time.

What is Laser Prep? PostScript was published after QuickDraw (the graphics routines in the Mac's ROM) were already established as the new graphic standard at Apple. The Mac generates QuickDraw primitives, the LaserWriter only understands PostScript... "what we have here is a failure to communicate." What to do?

Enter Laser Prep: Laser Prep is a PostScript prologue which is generally downloaded to the LaserWriter outside the "Server Loop" so it remains in the printer and is in effect from job to job. The reason for this approach is , Laser Prep is about 30K and we don't want to have to download it every time we send a job to the printer.

If you would like a text copy of Laser Prep to work with, hold-in the <Command K> keys immediately after pressing the OK button from the Print Dialog box. Instead of sending the document to the LaserWriter, the drivers will create a PostScript text file called "PostScript 0...9" on the desktop.

The PostScript file will include the Laser Prep prologue followed by the postscript for your document (if you print a blank page there won't be much). If you don't want the Laser Prep prepended, then do the same as above but press the <Command F> keys instead.

Problems with Apple's Laser Prep:

1. It has been notoriously "buggy". So much so, that Aldus decided to chuck it altogether and write their own prologue, Aldus Prep, for PageMaker™.

2. Since RESOURCES are altered via the print manager dialogs and written back to the driver, it is possible for the drivers themselves to be corrupted.

3. Laser Prep puts the LaserWriter in a default mode which causes a couple of problems for us if we want to send PostScript code to the printer and have it behave properly (that is unaffected by Laser Prep). The two problems are;

a. The PostScript "y" axis is inverted. This is what causes the example in the ZBasic manual to be up-side-down.

b. PostScript 0,0 original offset from where it should be.

There are two ways to remedy these problems.

#1. The first is straight forward but dependent on the version LaserWriter drivers you are using. Simply select "Options" in the "Page Setup" dialog box (DEF PAGE) and then select "Flip Vertical".

Modify Laser Prep!

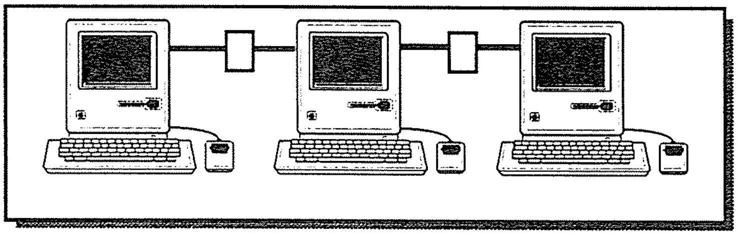
#2. Calling Laser Prep routines from our program to defeat default. Since Laser Prep establishes a dictionary of its own when downloaded, there is no reason why we can't use the operators in it ourselves.
Of course, since Laser Prep is not documented anywhere, you have to do a little digging around. If you add the line shown below as the very first line in the PostScript example in the manual (or any PostScript program you write based on the same model) it will correct the problems caused by Laser Prep (and without having to force the user to select all the correct options under DEF PAGE).

If you were to do the above and then capture the PostScript output in a file using the <Command F> method, you would come across a line like this: T T -8 -18 784 594 100 72 72 1 F F T F T F psu. This line calls routines in Laser Prep to adjust the graphics state of the LaserWriter. We can create our own "psu" line and override Laser Prep.

Feel free to publish or use this information any way you wish.
Sincerely,

Ken Jenkins
P.S. The ZBasic manual is one of the very finest examples of software documentation I have ever seen (and I've seen an awful lot), great job!

(Ah shucks Ken! Thanks! Ed.)



Special Version of ZBasic offers Network Support

ZBasic 4.01N opens Shared Read/Write files

Version 4.01 was out the door only 48 hours when we discovered a new parameter in the Apple System software that allows sharing files in both Read and Write modes. (Previous versions of ZBasic allowed shared read files when you set the "Shared bit" of a data file using ResEdit". Ed).

Well we thought about it a bit and decided to release version 4.01N. The N stands for Network. The only difference between this and 4.01 is that a new parameter is offered in the OPEN statement: OPEN "[R]N". The N signifies the file may be opened for shared read/write.

Since we don't have a network here to try it out on, we are awaiting feedback from end users. We are not sure if the user will have to do internal record lock-out or what sort of other problems may be encountered. This parameter was offered in the latest Volume "IV" of *Inside Macintosh*.

If you want this new version send your master disk back with $10 to cover our costs. We'll return the "N" version by First Class mail.

Add Libraries to ZBasic 4.01 for the Macintosh?

Previously Secret toolbox resource information Now Revealed in example program "Show toolbox.BAS"

In the "Program Examples" folder is a new example program in the "Toolbox Folder" called "Show Toolbox.BAS".

This interesting little program will show experienced BASIC programmers how the toolbox routines are added as resources to ZBasic.

With a little investigation you should be able to figure out how to add your own commands.

When you run the program, the entire toolbox routine tokens will be printed with their proper parameters. Just create your own libraries the same way and add them using ResEdit.

Watch for a future release of a Library Editor so you add and delete your own commands to ZBasic!!!

64K, 128K or 256K ROMS?

Ever Wonder how to Determine if a Program is running on a Mac II, SE, Plus or Other type of Macintosh?

Be sure to see the example program in the "Mac II folder" called "Which ROM?.BAS".

This little program will tell you if the program is running on a 64K, 128K or 256K ROM.

64K is Mac XL, 128K or Mac 512K. These versions do not support the new HFS.

128K is the Mac 512KE, Mac Plus and Mac SE.

256K is the Mac II.

Note: ZBasic windows on a Mac II now use PIXMAPS instead of bit maps. Avoid PEEK/POKE of direct graphics on a Mac II.

End Macintosh Section

AppleTalk Example Programs Wanted

DEAD OR ALIVE!

We need good working examples of AppleTalk programs for the Macintosh written in ZBasic. We will pay a $100 reward to the first person sending us an example program that illustrates many of the facets of using Appletalk (and works).

This $100 reward is in addition to the $75 fee you will get for an article about AppleTalk.

Don't hesitate ...

Send it today.

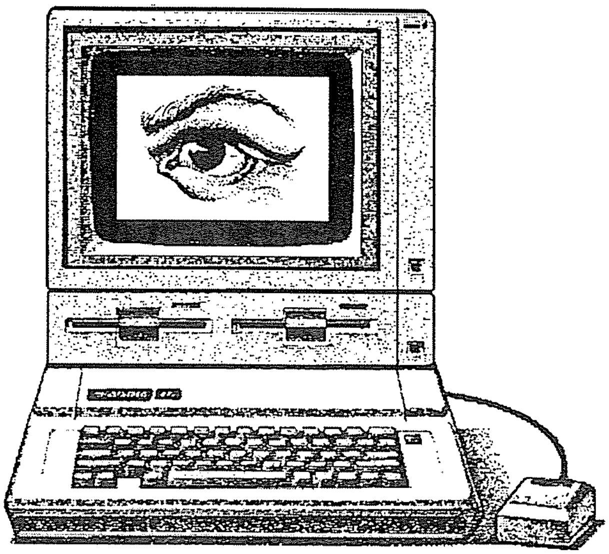
Apple continued from page 1...

If you don't have the latest version you can upgrade by sending us your master ZBasic diskette and $19.95 to:

Zedcor, Inc.
Apple II upgrade
4500 E. Speedway, #22
Tucson, AZ 85712-5305

Greg says he's working feverishly on the Apple //GS version. He says maybe this summer (we have our fingers crossed Greg).

He says if you want any special features, send your ideas now so he can plan on implementing them.



Do Screen Dumps to a Printer with this handy Routine.

For those of you who'd like to have a PAGE LPRINT command like the Mac and IBM versions of ZBasic, here's another useful routine by Greg Branche.

Just add this routine to your program and do: GOSUB "PAGE LPRINT".

```
"PAGE LPRINT"
:
REM Performs an 80-column screen dump to the printer
REM Adapted to ZBasic from Apple Miscellaneous Tech Note #1
:
ROUTE 128
MACHLG &A2,&00,&8A,&20,&C1,&FB,&A0,&00,&8D,&01,&C0,&8D,&55,&C0
MACHLG &98,&48,&4A,&90,&03,&8D,&54,&C0,&A8,&B1,&28,&8D,&54,&C0
MACHLG &20,&ED,&FD,&68,&A8,&C8,&C0,&50,&90,&E2,&A9,&8D,&20,&ED
MACHLG &FD,&E8,&E0,&18,&90,&D2
ROUTE 0
RETURN
```

How to Get Free Space on ProDOS Disk Volumes

Use this function in your programs to get the free space for ProDOS volumes.

Here's a LONG FN you can use in your programs to determine the amount of free space in a ProDOS volume.

This can come in handy quite often, especially when you don't want your programs to run out of disk space.

Thanks again to Greg Branche.

GETSIZE.BAS

```
REM includes and demonstrates a LONG FN that can be used
REM to retrieve the block usage information of a volume
:
LONG FN GETSIZE(VOL$)
  REM VOL$ contains pathname of the volume to get info
  POKE WORD &1F01, VARPTR(VOL$):REM Set pointer->string
  POKE &1F00, 10  : REM 10 parameters for GET_FILE_INFO
  MACHLG &A9, &C4
  REM MACHLG &20, &803 : REM Use this line for 64K ver.
  MACHLG &20, &865 : REM Use this line for 128K version
  MACHLG &90, 3
  REM MACHLG &4C, &809 : REM Use this line for 64K ver.
  MACHLG &4C, &87F : REM Use this line for 128K version
  MAXSIZE% = PEEK WORD (&1F05) :REM total blocks avail.
  USED% = PEEK WORD (&1F08)  : REM Get # of blocks used
  FREE% = MAXSIZE% - USED% : REM  # blocks free
END FN = FREE%
INPUT "ENTER VOLUME NAME -> "; A$
Blocks_Free% = FN GETSIZE(A$)
PRINT UNS$(MAXSIZE%) " TOTAL BLOCKS, ";
PRINT UNS$(USED%) " BLOCKS USED, ";
PRINT UNS$(Blocks_Free%) " BLOCKS FREE"
END
```

APPEND function for use with ProDOS ZBasic

Use this Function to Append data to an existing file.

Lots of people have asked for this function so here it is. Many thanks to Greg Branche.

To use the example (right), just add it to your program with MERGE or APPEND and then use the following function in your programs:

```
FN Open_Append
```

Any subsequent output to an existing file will be appended without changing the data.

```
LONG FN Open_Append(FileNum%, Path$)
  REM FileNum% is the file # you wish to use: (1-8)
  REM Path$ is the pathname of the file to open for append
  OPEN "O", FileNum%, Path$: REM Open the file
  REF% = PEEK(&1F05)        : REM Get ProDOS refnum
  POKE &1F01, REF%          : REM Set into parameter block
  POKE &1F00, 2             : REM Number of parameters
  MACHLG &A9, &D1           : REM LDA #$D1 (GET_EOF)
  REM MACHLG &20, &803      : REM JSR $803 (64K VERSION)
  MACHLG &20, &865          : REM JSR $865 (128K VERSION)
  MACHLG &90, 3             : REM BCC *+3 (check file error)
  REM MACHLG &4C, &809      : REM JMP $809 (64K VERSION)
  MACHLG &4C, &87F          : REM JMP $87F (128K VERSION)
  MACHLG &A9, &CE           : REM LDA #$CE (SET_MARK)
  REM MACHLG &20, &803      : REM JSR $803 (64K VERSION)
  MACHLG &20, &865          : REM JSR $865 (128K VERSION)
  MACHLG &90, 3             : REM BCC *+3 (check file error)
  REM MACHLG &4C, &809      : REM JMP $809 (64K VERSION)
  MACHLG &4C, &87F          : REM JMP $87F (128K VERSION)
END FN
:
OPEN "O", 1, "TESTFILE.DAT" : REM Create the file initially
FOR I = 1 TO 10             : REM Just dump stuff to file
  PRINT #1, "This is line number"; I
NEXT I
CLOSE #1
                         : REM Now try the append function
FN Open_Append(1, "TESTFILE.DAT")
FOR I = 11 TO 20         : REM Lines 1 - 10 already in file
  PRINT #1, "This is line number"; I
NEXT I
CLOSE #1
                        : REM See what's actually in the file
OPEN "I", #1, "TESTFILE.DAT"
ON ERROR GOSUB 65535    : REM Turn off ZBasic's error check
WHILE ERROR = 0
  INPUT #1, A$          : REM Get a line of input
  PRINT A$              : REM Display it
WEND
ERROR = 0               : REM Must clear ERROR ourselves
ON ERROR RETURN         : REM Give error checking back
CLOSE #1
END
```

Changing Hard-Drive Volumes with the Apple *//* DOS 3.3 ZBasic

A Helpful program for Hard-Disk users "plagued" with DOS 3.3.

A ZBasic user was nice enough to send us source code that lets you change volumes using the old DOS 3.3 version of ZBasic;

Dear Mike,
Enclosed is a sample program that changes the volumes on an Apple hard drive formatted in DOS 3.3. The program assumes that you are currently in the slot and drive of the hard drive.

I am currently developing software using ZBasic that runs on these hard drives [see box]. Some of the programs access as many as thirty different volumes. This method has proven most successful.

This program should be helpful to those hard disk users still plagued with DOS 3.3.

Fran Breit
University of Wisconsin
(at Madison)
Vocational Studies Center

P.S. Thanks for your technical support. It is appreciated.

(*Thank you Fran for sending this listing. Ed*)

Changing Volumes on Hard Drives Formatted with Apple DOS 3.3

```
REM Program demonstrates changing
REM volumes on CORVUS and SIDER hard
REM drives. It assumes these drives were
REM formatted with DOS 3.3.
:
REM ZBasic allows one to configure the
REM slot and drive but not the Volume.
:
REM This program lets you get around that
REM by poking the volume you want to
REM access. I suspect this method will
REM work on other hard drives formatted
REM in DOS 3.3 (no guarantees).
:
:
REM This LONG FN requires the file name
REM (filename$), the file number (Fnum%)
REM and the Volume number (Vol%)
:
REM If 0 is returned then no disk error
REM occurred.  If a non-zero is returned
REM it is the disk error number.
:
:
LONG FN OpenVol% (Filename$, Fnum%, Vol%)
  x%=0
  ON ERROR GOSUB 65535:REM Error check on
  :
  POKE 43622, Vol%   :REM Change volume
  OPEN "I",Fnum%, Filename$
  :
  IF ERROR THEN x%=ERROR: ERROR=0
  ON ERROR RETURN:REM Error check off
END FN=x%
:
REM Example...
:
Error%=FN OpenV% ("Fred.Txt", 1, 3)
:
LONG IF Error%>< 0
  PRINT "A Disk error occurred"
  PRINT ERRMSG$(Error%)
  DELAY 2000: STOP
END IF
:
REM  Program continues...
:
END
```

End Apple Section

IBM continued from page 1...

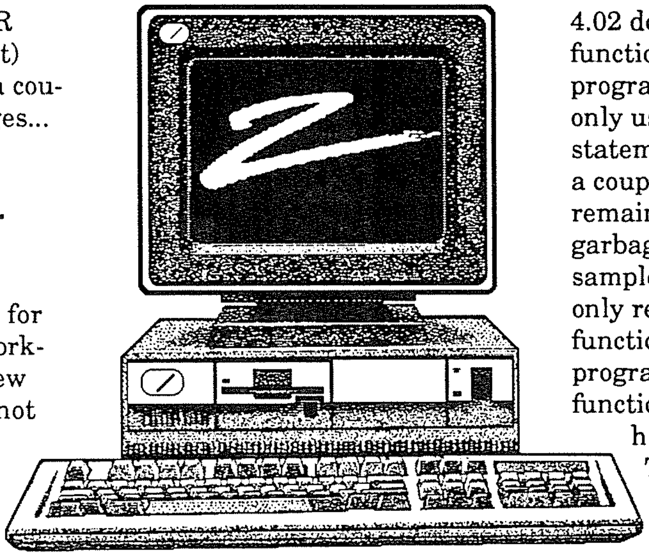
from BIOS; a delay adjustor, modem routines, a Special routine that lets you create TSR (terminate and stay resident) programs with ZBasic and a couple more. See following pages...

ZBasic 4.02, IBM version Bug Report

The latest version of ZBasic for MSDOS and IBM PC's is working well. There are still a few quirks (don't worry they're not nasty ones this time). The following patches fix the problems:

1. The first problem will show up if you have a program line containing an IF statement followed by a blank line (null line). When the compiler converts your source code into object code, the incorrect address is stored at a JMP instruction. To fix the problem simply don't put a blank line after an IF statement. Future versions will have a fix installed.

2. The second bug will appear if you happen to type a quote (") in place of a colon (:) as a statement separator in a program line. When you compile the program, the compiler will flag the quote as the incorrect character, highlight the quote, and then continue to highlight the rest of the line. When you edit the line, the first character of the line will then be converted into a random line #. To correct this problem, use ZBasic's patch utility to change the byte at address &644C from &B4 to &B2.



3. Next, we have a slight problem with the CASE statement when used with floating point arguments and relational operators. For example:

```
FOR I = 1 TO 3
  READ X#
  PRINT X# "IS ";
  SELECT X#
    CASE > 0
      PRINT "GREATER THAN 0"
    CASE = 0
      PRINT "EQUAL TO 0"
    CASE < 0
      PRINT "LESS THAN 0"
  END SELECT
NEXT I
DATA -2.25, 0, 5.75
```

Produces:

```
-2.25 IS GREATER THAN 0
 0 IS EQUAL TO 0
 5.75 IS LESS THAN 0
```

Oops! To fix this, patch location &9A85 from &73 to &7B.

4. The next problem appears if you attempt to use the ON INKEY$ statement to set up function key vectors. Version 4.02 does not properly clear the function key vector table during program initialization. If you only use the ON INKEY$ statement to set the vectors for a couple of the function keys, the remaining vectors contain garbage. For example, the sample program on page A-67 only recognizes the F1 and F2 function keys. If you run this program and press any other function key, the system will hang, forcing you to reboot. To correct this problem, use the following subroutine in your program, and then call the subroutine prior to executing an ON INKEY$ statement:

```
"FIX ON INKEY$"
FOR I = 0 TO 24
  POKE WORD &187 + I, 0, MEM C
NEXT I
RETURN
```

5. The next bug will appear if you attempt to get an address of an integer array element using VARPTR. The incorrect address will be returned by the function. Install the following patch to correct the bug:

Address	Old Value	New Value
&BD72	&52	&E8
&BD73	&BA	&CD
&BD74	&C3	&FA
&BD75	&07	&BA
&BD76	&E8	&C3
&BD77	&61	&07
&BD78	&FA	&E8
&BD79	&5A	&5F
&BD7A	&E8	&CA

continued next page...

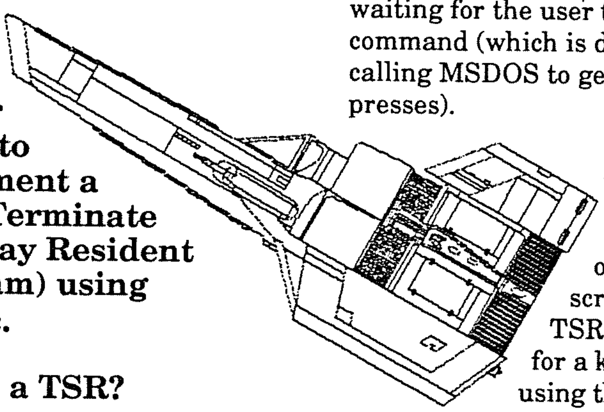
&BD7B	&C5	&E9
&BD7C	&FA	&3C
&BD7D	&E9	&FE
&BD7E	&3A	&90
&BD7F	&FE	&90
&BD90	&80	&F6
&BD91	&FB	&C3
&BD92	&10	&C0
&BD93	&74	&74
&BD94	&DD	&E0
&BD9C	&E0	&E2

Once you are sure that the patches have been installed correctly, you can save the corrected code to disk using the S)ave option from the ZBasic startup screen.

A TSR WRITTEN IN ZBASIC!

by Greg Branche

It seems like everybody has been asking for a way to implement a TSR (Terminate and Stay Resident program) using ZBasic.



What's a TSR?

Ever use programs like SideKick™ from Borland? This is a TSR (Terminate and stay resident in memory). These programs are neat because they are loaded into memory and are always there by pressing one or two keys (Control Alt for Sidekick). When you exit a TSR you are back in your original program. To purge TSR's from memory you must reboot. The ESC key is most often used to exit a TSR. *Ed.*

Well, I finally did it! It wasn't easy, but this particular TSR demonstrates some useful techniques that you can use when you write your own.

While writing this, I also ran into some problems which you need to keep in mind when writing TSR's: MSDOS is NOT reentrant. This means that a TSR cannot make use of any MSDOS functions if the TSR is called while MSDOS is currently executing a function.

I discovered the consequences while I was testing the program. Normally, the TSR was being activated when the system was waiting for the user to enter a command (which is done by calling MSDOS to get the key presses).

After displaying a message on the screen, the TSR would wait for a key press by using the INKEY$ function. ZBasic implements the INKEY$ function by calling MSDOS to get a key press (Bingo!). When it did, of course, MSDOS overwrote the previous values of it's system variables, trashing the values it was using in the previous call. When the TSR exited back to the system, since MSDOS' variables had now been trashed, the system just locked up. I had to turn the machine off and back on again to regain control. So, keep this in mind when writing your own TSR's.

You must also be VERY familiar with architecture of the IBM and 8088 assembly language.

TSR's usually deal with the machine at a very low level, and some things just cannot be done from a higher level language. You can see all of the MACHLG statements in the accompanying listing.

Now, for some program specific comments. This particular TSR can only be executed on machines that have true IBM text compatibility. This is because of the saving and restoring of the text screen. The program uses machine-language block moves to copy the contents of screen memory to and from an array. The screen memory must be where it's "supposed" to be.

There are no provisions in this program for the TSR to "unload" itself from memory. Once installed, the only way to get rid of it is to reboot the machine. In addition, there is no provision to prevent the installation of more than one copy in memory. I'll leave it to the curious hacker to implement these two features.

Once installed, the TSR checks the keyboard for a CTRL-ESC key press sequence. When it sees this key sequence, it grabs control of the machine and displays a simple message on the screen. In TSR lingo, this CTRL-ESC sequence is known at the "hot key."

The first part of the program contains function definitions and program initialization. The Get_Vector function simply makes a call to MSDOS to retrieve the current segment:offset of an interrupt handler. Set_Vector performs the inverse function (it sets the vector of an interrupt handler).

The SaveScreen function performs a machine-language block move routine that copies the current contents of screen memory to a ZBasic array. It returns the current cursor position as a 16-bit integer for storage in a ZBasic integer. Make sure you DIMension a large enough array to hold the contents of the screen (see line 30). PutScreen (530-670) simply copies the contents of the array holding the saved screen back into screen memory. It also repositions the cursor to it's saved position. The function calls to SaveScreen and PutScreen both assume that a CGA video adapter is currently installed. Change the &B800 in both lines if you are using a different display adapter. Both functions also assume that page 0 is the current page. The functions will have to be modified slightly if you are using a different display page.

I save ZBasic's Data Segment into the compiled object code so that when the TSR is invoked, it can find it's variables. I save the current contents of the interrupt &16 vector (the keyboard BIOS interrupt), point the vector to the TSR, and set up some internal pointers. The last few lines perform an MSDOS Terminate & Stay Resident function call to return control back to the system. MSIZE% is the number of paragraphs to reserve, and is calculated by subtracting MEM C (ZBasic's lowest segment in memory) from MEM I (the INDEX$ segment, which is the highest segment in memory that ZBasic uses).

When the TSR actually starts, it is called each time a program executes an INT 16h call. The first thing it does is check the function number requested. If it's function 1 or 2, it simply passes the function call on to the original interrupt vector (normally BIOS). If it is function 0, it calls the original vector itself to get the key press. It then checks the keypress for an ESC code. If it's any other keypress, it simply returns the keypress back to the caller.

If an ESC is pressed, the TSR checks to see if the CTRL key was pressed at the same time, and, if not, returns the keypress back to the caller. Only if both CTRL and ESC are pressed at the same time does the TSR "take control" of the machine.

First, it must save any registers that it's going to use (and a compiled ZBasic program usually uses ALL of them) so that they can be restored prior to exiting back to the interrupted process. Then the program picks up the correct Data Segment, and continues on.

After performing it's function, the program restores all registers, and then re-CALLs the keyboard interrupt to get the keypress that was originally requested.

As I said before, it's not easy, but it is possible.

Hopefully, this will satisfy your appetites for Terminate & Stay Resident style programs.

TSR Example Program listing starts on next page...

TERMINATE AND STAY RESIDENT

Page 1

```
' ONLY WORKS WITH TRUE IBM COMPATIBLE TEXT SCREENS
CLEAR 0 'DON'T USE INDEX$ ARRAY WITH TSR'S
DIM Oldoff%, Oldseg% ' MUST be in sequence
DIM Addr%, SSAVE%(79,24) ' for screen save
POKE &342,1 ' Sets to IBM compatible text
Adr% = VARPTR(Addr%) + 2 ' Calculates address of SSAVE array
:
LONG FN Get_Vector(Vector%, Adr%)
  ' Vector% contains interrupt #
  ' Adr% contains address of double word variable to store vector in
  MACHLG 6                     ' PUSH ES             ;SAVE ES
  MACHLG &B4, &35              ' MOV  AH,35h         ;FUNCTION #
  MACHLG &A0, Vector%          ' MOV  AL,[Vector%]   ;INTERRUPT VECTOR TO GET
  MACHLG &CD, &21              ' INT  21h            ;CALL DOS
  MACHLG &93                   ' XCHG BX,AX          ;SAVE ADDRESS IN AX
  MACHLG &8B, &1E, Adr%        ' MOV  BX,[Adr%]      ;GET ADDRESS OF VARIABLE
  MACHLG &89, 7                ' MOV  [BX],AX        ;SAVE OFFSET OF VECTOR
  MACHLG &83, &C3, 2           ' ADD  BX,2           ;ADJUST TO VAR ADDRESS
  MACHLG &8C, 7                ' MOV  [BX],ES        ;SAVE SEGMENT OF VECTOR
  MACHLG 7                     ' POP  ES             ;RESTORE ES
END FN
:
LONG FN Set_Vector(Vector%, Seg%, Offset%)
  ' Vector% contains interrupt #
  ' Seg% contains segment of new interrupt vector
  ' Offset% contains offset of new interrupt vector
  MACHLG &1E                 ' PUSH DS               ;SAVE DS
  MACHLG &8B, &1E, Seg%      ' MOV  BX,[Seg%]        ;GET SEGMENT OF NEW VECTOR
  MACHLG &8B, &16, Offset%   ' MOV  DX,[Offset%]     ;GET OFFSET OF NEW VECTOR
  MACHLG &B4, &25            ' MOV  AH,25h           ;FUNCTION #
  MACHLG &A0, Vector%        ' MOV  AL,[Vector%]     ;INTERRUPT VECTOR TO SET
  MACHLG &8E, &DB            ' MOV  DS,BX            ;PUT SEGMENT INTO DS
  MACHLG &CD, &21            ' INT  21h              ;CALL DOS
  MACHLG &1F                 ' POP  DS               ;RESTORE DS
END FN
:
LONG FN SaveScreen%(SaveAdr%, Seg%)
  ' SAVE THE CURRENT SCREEN INTO SSAVE, RETURNS CURRENT CURSOR POSITION
  MACHLG &1E                ' PUSH DS              ;SAVE DS
  MACHLG &31, &F6           ' XOR  SI,SI           ;CLEAR SI
  MACHLG &8B, &3E, SaveAdr% ' MOV  DI,[SaveAdr%]   ;POINT TO STORAGE
  MACHLG &B9, 2000          ' MOV  CX,2000         ;2000 BYTES IN SCREEN MEM
  MACHLG &A1, Seg%          ' MOV  AX,[Seg%]       ;GET SCREEN SEGMENT
  MACHLG &8E, &D8           ' MOV  DS,AX           ;PUT INTO DS
  MACHLG &F3, &A5           ' REPZ MOVSW           ;COPY FROM SCREEN TO ARRAY
  MACHLG &1F                ' POP  DS              ;RESTORE DS
  MACHLG &B4, 3             ' MOV  AH,3            ;MAKE BIOS CALL
  MACHLG &B7, 0             ' MOV  BH,0            ;(CURRENT PAGE #)
  MACHLG &CD, &10           ' INT  10h             ;TO GET CURSOR POSITION
  MACHLG &89, &16, a%       ' MOV  [a%],DX         ;SAVE POS IN a%
END FN = a%
```

TERMINATE AND STAY RESIDENT

Page 2

```
LONG FN PutScreen(SaveAdr%, Seg%, CursorPos%)
  ' RESTORE THE SCREEN
  MACHLG 6                    ' PUSH ES                 ;SAVE ES
  MACHLG &8B, &36, SaveAdr%   ' MOV  SI,[SaveAdr%]      ;POINT TO ARRAY
  MACHLG &31, &FF             ' XOR  DI,DI              ;CLEAR DI
  MACHLG &B9, 2000            ' MOV  CX,2000            ;2000 BYTES IN SCREEN
  MACHLG &A1, Seg%            ' MOV  AX,[Seg%]          ;GET SCREEN SEGMENT
  MACHLG &8E, &C0             ' MOV  ES,AX              ;PUT INTO ES
  MACHLG &F3, &A5             ' REPZ MOVSW              ;RESTORE SCREEN MEMORY
  MACHLG 7                    ' POP  ES                 ;RESTORE ES
  MACHLG &B4, 2               ' MOV  AH,2               ;BIOS CALL
  MACHLG &B7, 0               ' MOV  BH,0               ;(CURRENT SCREEN #)
  MACHLG &8B, &16, CursorPos% 'MOV  DX,[CursorPos%] ;GET SAVED CURSOR POS
  MACHLG &CD, &10             ' INT  10h                ;SET CURSOR POSITION
END FN
:
Dseg% = MEM D
POKE WORD LINE "POKE LOC" + 1, Dseg%, MEM C ' Save ZBasic's Data Segment
FN Get_Vector(&16, VARPTR(Oldoff%)) ' Get vector for keyboard interrupt
FN Set_Vector(&A0, Oldseg%, Oldoff%) ' Save in unused vector (&A0)
POKE WORD LINE "INTVEC", Oldoff%, MEM C ' Set up jmp pointers
POKE WORD LINE "INTVEC" + 2, Oldseg%, MEM C
POKE WORD LINE "ADDRESS FIX 1" + 3, LINE "INTVEC", MEM C
FN Set_Vector(&16, MEM C, LINE "TSR") ' Set new keyboard vector
PRINT "ZBasic TSR installed!" ' I'm here!
MSIZE% = MEM I - MEM C ' Calculate size of ZBasic memory used
MACHLG &B8, &3100             ' MOV  AH,3100h      ;FUNCTION 31h, EXIT = 0
MACHLG &8B, &16, MSIZE%       ' MOV  DX,[MSIZE%]   ;SIZE OF PGM IN DX
MACHLG &CD, &21               ' INT  21h           ;Perform TSR call
"INTVEC" MACHLG 0,0,0,0 ' Storage for original interrupt vector
"TSR"
MACHLG &0A, &E4               ' OR   AH,AH         ;FUNCTION 0?
MACHLG &74, &05               ' JE   +05           ;YES, POSSIBLE HOT KEY
"ADDRESS FIX 1"
MACHLG &2E, &FF, &2E, 0, 0    ' JMP  L,CS:[INTVEC] ;ELSE PASS TO ORIGINAL
MACHLG &CD, &A0               ' INT  A0h           ;PERFORM ORIGINAL CALL
MACHLG &3C, &1B               ' CMP  AL,1Bh        ;ESC?
MACHLG &74, &01               ' JE   +01           ;NO, RETURN TO CALLER
MACHLG &CF                    ' IRET
MACHLG &50                    ' PUSH AX            ;SAVE KEYPRESS
MACHLG &B4, &02               ' MOV  AH,2          ;GET KEYBOARD STATUS
MACHLG &CD, &A0               ' INT  A0h           ;CALL ORIGINAL INTERRUPT
MACHLG &25, &04, &00          ' AND  AX,0100b      ;CTRL KEY PRESSED TOO?
MACHLG &75, &02               ' JNE  +02           ;YES, HOT KEY!
MACHLG &58                    ' POP  AX            ;ELSE RESTORE KEYPRESS
MACHLG &CF                    ' IRET               ;RETURN TO CALLER
MACHLG &58                    ' POP  AX            ;FIX STACK
MACHLG &53, &51, &52, &56     ' PUSH BX,CX,DX,SI   ;SAVE ALL REGISTERS
MACHLG &57, &55, &1E, &06     ' PUSH DI,BP,DS,ES
"POKE LOC"
MACHLG &BB, 0, 0              ' MOV  BX,MEM D      ;GET ZBASIC'S DATA SEGMENT
MACHLG &8E, &DB               ' MOV  DS,BX
```

TERMINATE AND STAY RESIDENT

```
Curpos% = FN SaveScreen%(Addr%, &B800) ' Save the screen
LOCATE ,,0 ' Turn cursor off
DEF PAGE 20,5 TO 59,12 : CLS ' Define window and clear it
DEF PAGE 20,5 TO 59,13 ' Increase bottom to prevent scrolling
COLOR ,11 ' Light cyan border color
PRINT CHR$(201); ' Print window border

FOR I% = 21 TO 58
  PRINT CHR$(205); ' Top line
NEXT I%
PRINT CHR$(187);

FOR I% = 6 TO 11
  PRINT CHR$(186); TAB(39); CHR$(186); ' Window sides
NEXT I%

PRINT CHR$(200);

FOR I% = 21 TO 58
  PRINT CHR$(205); ' Bottom line
NEXT I%
PRINT CHR$(188);

DEF PAGE 20,5 TO 59,12 ' Back to actual size of window
COLOR ,14 ' Yellow text color
PRINT @(28,8) "This is your ZBasic TSR"
COLOR ,4 : PRINT @(36,9) "CHARGE!"
RESTORE ' Start at the beginning of the DATA statement
USR 2(725) ' Set correct system speed (8 Mhz 80286)

FOR I = 1 TO 6 : ' Charge!
  READ T, D: SOUND T,D
  DELAY 10
NEXT I

DATA 792,100,880,100,990,100,1188,300,1024,150,1188,500
DELAY 1000 ' Wait a little longer
DEF PAGE 0,0 TO 79,24 ' Back to full screen window
COLOR ,7 ' Restore to white text
FN PutScreen(Addr%, &B800, Curpos%) ' Restore screen
:
LOCATE ,,1 ' Turn cursor back on
MACHLG &07, &1F, &5D, &5F    ' POP  ES,DS,BP,DI    ;RESTORE ALL REGISTERS
MACHLG &5E, &5A, &59, &5B    ' POP  SI,DX,CX,BX
MACHLG &B4, &00              ' MOV  AH,0           ;PERFORM ORIGINAL CALL
MACHLG &CD, &16              ' INT  16h            ;RE-CALL KBD INTERRUPT
MACHLG &CF                   ' IRET                ;THEN RETURN
END
```

ZBasic too Fast for Some PC Direct Modem Commands

A few people are having problems getting ZBasic to talk to their modems correctly.

When trying to PRINT an "AT" command, they say that the modem never receives the command.

After playing with this extensively, it has been discovered that ZBasic's PRINT routines are simply too fast for the modem to keep up.

The following program demonstrates a LONG FN that can be used to "slow down" the output to the modem. For the function to operate correctly, the modem must be able to echo the command characters back to the sending device (usually set with the "ATE" modem command).

In addition, modem commands must start at the beginning of a line, hence the CHR$(13) at the beginning of AT$.

Greg Branche

Example for Using Modems

```
AT$ = CHR$(13) + "AT"
OPEN "C", -1, 1200, 0, 0, 1, &20 ' Inits the com port

LONG FN Comprint(a$)
  FOR I = 1 TO LEN(a$) ' Loops through a$
    b$ = MID$(a$,I,1)   ' Grab one character at a time
    WRITE #-1, b$;1     ' Write the character to the port
    DO
      READ #-1, b$;0    ' Wait til char echoed by modem
    UNTIL LEN(b$)
  NEXT I
  b$ = CHR$(13)         ' Write terminating carriage return
  WRITE #-1, b$;1
  DO
    READ #-1, b$;0      ' And wait for it to be echoed
  UNTIL LEN(b$)
END FN

INPUT "Please enter the number to dial -> "; N$
Dial$ = AT$ + "DT " + N$
FN Comprint(Dial$)

"Fall into a simple Terminal Program"
DO
  READ #-1, B$;0
  IF LEN(B$) THEN PRINT B$;
  B$ = INKEY$
  IF LEN(B$) THEN WRITE #-1, B$;1
UNTIL B$ = "]"

PRINT "Thanks for using ZTerm!"
END
```

Get Drive Info on IBM PCs

Returns Logical, Floppy and Current Drives.

If you ever had a need to find the Logical drive, Floppy Drive or Current drive from within your application these routines are for you.

Greg Branche

```
MACHLG &B4,&19,&CD,&21,&8A
MACHLG &D0,&B4,&0E,&CD,&21
MACHLG &98,&A3,LDRIVES%

MACHLG &CD,&11,&B1,&06,&D3
MACHLG &E8,&25,&03,&00,&40
MACHLG &A3,FDRIVES%

MACHLG &B4,&19,&CD,&21,&98
MACHLG &A3, CURDRIVES%

PRINT LDRIVES%;"Logical"
PRINT FDRIVES%;"Floppy"
PRINT CURDRIVES%;"Current"
```

How to Get the Amount of Free Disk Space for MSDOS

Returns bytes available on specified drive.

Ever need to find the free space on a disk? Well Greg was nice enough to include a good example of fixing this on the ZBasic Master diskette in the Samples Subdirectory called DiskFree.FN.

Try it out. It could come in handy some time.

COMPILE filename

You've asked for routines and stuff, but how about being able to compile programs using a simple .BAT file. I like the feature where you can compile programs using:

```
ZBasic filename.BAS filename.COM
```

but I Hate typing all that! So I wrote this batch program:

```
File: COMPILE.BAT
Echo off
Cls
IF '%1'=='' goto nofile
If not exist %1 goto badexit
Echo Compiling: %1
ZBasic %1 %1.COM
Goto exit
:nofile
Echo No file name has been specified.
Echo Please retype the command.
Goto Exit
:badexit
Echo File %1 does not exist.
Echo Please retype command.
:exit
```

So now when I want to compile a program I just type:

```
COMPILE filename
```

(from the DOS prompt). This time saver submitted by: Peter Bennett, Bank of Nova Scotia, 44 King Street West, Suite 1321, Toronto, Ont. Canada M5H 1H1. *Thanks Peter!*

Get and Set Cursor Positions for IBM PCs

These routines will get and set cursor positions for different screens.

The following program demonstrates a couple of LONG FNs to get and set the cursor position using BIOS functions.

Feel free to use it in your programs.

End IBM

Get and Put Cursor Information

```
' This program demonstrates two functions that can be used
' to save the current cursor position and restore it.
:
LONG FN Getpos%              ' Returns current position in 16-bits
  MACHLG &B4, 3              ' MOV  AH,3
  MACHLG &B7, 0              ' MOV  BH,0      ; SCREEN NUMBER
  MACHLG &CD, &10            ' INT  10h
  MACHLG &92                 ' XCHG DX,AX
END FN                       ' Returns a value even w/o the "= var"
:
LONG FN Setpos%(Curpos%)     ' Restores cursor position to FN Getpos%
  MACHLG &B4, 2              ' MOV  AH,2
  MACHLG &B7, 0              ' MOV  BH,0      ;SCREEN NUMBER
  MACHLG &8B, &16, Curpos%   ' MOV  DX,[Curpos%]
  MACHLG &CD, &10            ' INT  10h
END FN
:
LOCATE 40, 20                ' Position the cursor
CUR% = FN Getpos%            ' Call function to get cursor position
X% = CUR% AND &FF            ' Low order byte is column position
Y% = CUR% >> 8               ' High order byte is row position
PRINT
PRINT "X,Y ="X%","Y%
FN Setpos%(CUR%)             ' Restore to previous position
PRINT "HELLO THERE!"
LOCATE 0,24                  ' Position to bottom of screen to exit

END
```

is YOUR Newsletter! Whether you own a PC, a Macintosh, an Apple II or a Kaypro 4 you owe it to yourself to subscribe to time saving programming ideas.

Subscription is only $19.95 a year ($37 for two years). Overseas orders (outside U.S.) add $5.00. To subscribe call Toll Free 800-482-4567. Have your credit card ready. To subscribe by mail please fill out and and return this card with Credit Card#, Check or Money order:

Name______________________________

Company____________________________

Address ____________________________

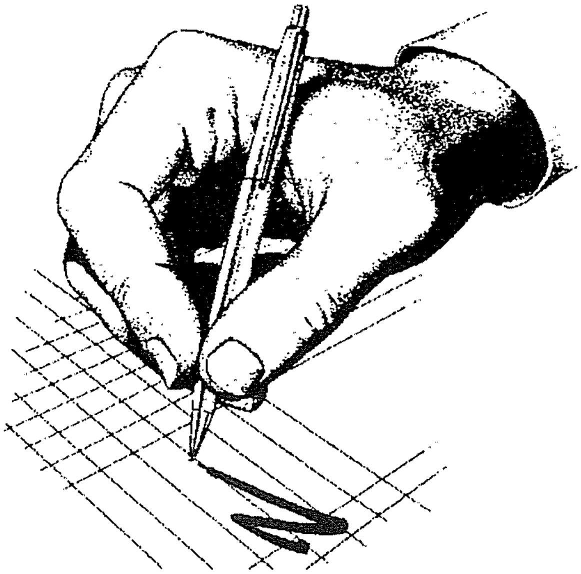
City __________________ ST____ ZIP_________

Daytime Phone (_____)_______________________

Credit Card#_________________________

Expiration date_______________________

Signature___________________________



ZEDCOR INC.
4500 E. Speedway, Suite 22
Tucson, AZ 85712-5305
(602) 881-8101
(800) 482-4567

Bulk Rate
U.S. Postage
PAID
Tucson, AZ
Permit Number
2220