The Newsletter for ZBasic Enthusiasts

Z™

Upgrade Policies

Dr. Z

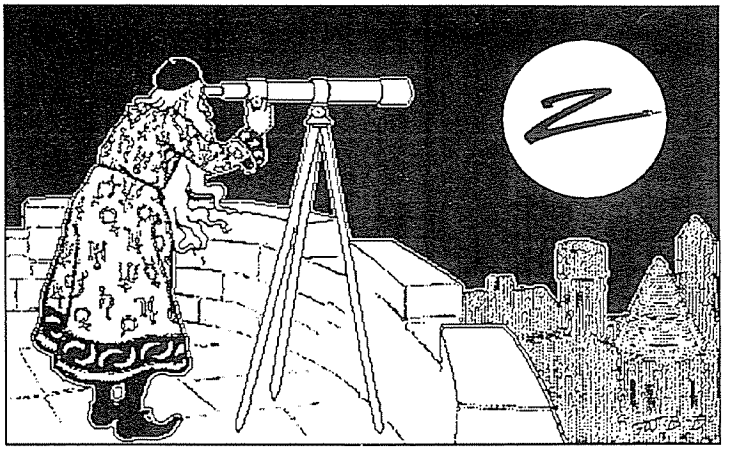
Apple IIGS Paint Program

Many Example Programs

TRS-80 Model 4 Music

How to Hide the Macintosh Menu Bar!

Fall, 1987



INSIDE...

Zedcor Upgrade Policies 2

Macintosh 4

Hide the Menu Bar 5

Bezier Curves 6

Dr. Z 7

Genesis 9

Letters to the Editor 10

Apple II 12

Pathnames at runtime 12

Machine Language in the Main Bank 13

IIGS Paint Program 14

Four New Weapons for programmers 17

MSDOS 18

Bug fixes and patches for version 4.01 18

Patch Program 20

Pallette Demonstration program 22

Z80 (CP/M and TRS-80) 23

TRS-80 Model 4 Sound 23

Michael A. Gariepy
Editor in Chief
Advertising

Andrew Gariepy
Technical Editor

ZEDCOR, Inc.

PRESIDENT
Andrew Gariepy

CHAIRMAN
Michael A. Gariepy

Fall, 1987

"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

"Z" is a technical journal not affiliated in any way with computer manufacturers.

Zedcor Upgrade Policies

New releases of any software product are always frustrating. Regardless of how many hours are spent debugging, alpha-testing and beta-testing, a number of nasty bugs always slip through. It's downright embarrassing. A product with the complexity of ZBasic means that certain combinations of commands, certain syntactical situations and certain problems with not-so-compatible hardware cause problems that are impossible to predict in advance.

But ship a couple of thousand upgrades out and you find the bugs real fast! We fix them as fast as we can find them. We need your help to locate some of them since we may not have the same hardware set-up as you.

Our Goal is Customer Satisfaction

We aim to give you the product you want and upgrades at fair prices. We are continously working on ZBasic to make adjustments and improvements that you ask for.

The following policies will apply to ZBasic upgrades. We believe they are fair and equitable and allow us to sell ZBasic at a reasonable price.

FREE UPGRADES WITHIN 90 DAYS
When available, you may order a free upgrade by sending in your master ZBasic diskette within 90 days from the date of original purchase or within 90 days of receiving a major upgrade.

$19.95 AFTER 90 DAYS
If you've had ZBasic for more than 90 days the upgrade charge will be $19.95. Simply call 1-800-482-4567 to order this upgrade. This also covers the cost of upgrading within the same version. i.e. if you own version 4.01 and the newest release is version 4.03, you may upgrade for $19.95. A new manual is not included with interim upgrades. Usually an appendix with changes or a "READ ME" file is included on the disk.

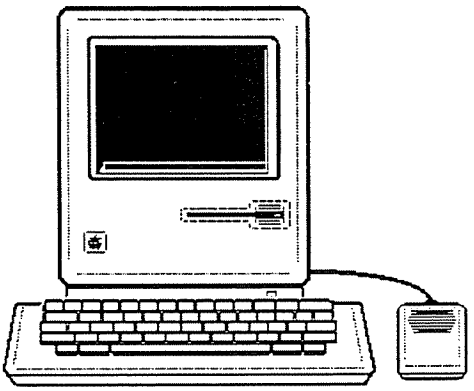
$39.95 FOR MAJOR UPGRADES
When upgrading from version 4.04 to version 5.0, the upgrade charge will be $39.95. This covers the costs of the incidentals mentioned above as well as new documentation.

Be sure to read the section of this newsletter pertaining to your computer to keep tabs on bugs, upgrades and patches (if any).

Free Upgrade if you find a New Bug
If you find a new bug (verified by our support people) and mail us a complete description, you will receive a free upgrade to the next version. This does not apply to old bugs (call our support line to verify that it has not been discovered previously). We will forward you a new version as soon as it has been completed.

We need your help finding bugs and really appreciate it when you report them.

These upgrades policies are subject to change.



Macintosh

As of this date Macintosh ZBasic version 4.0 is out and running! People are saying nice things about us... like we're the best BASIC for the Macintosh (which you already knew).

BUG REPORT

As you guessed we have some bugs in version 4.0. Here's the latest scoop:

• The edit window has problems with System 4.1. Appears there was some kind of problem with TEEDIT in System 3.2 that was fixed in 4.1 which caused our editor problems. It is being fixed now.

• No Help file. This will come out in the next release. We changed the help file significantly and have to create a PICT format file. We didn't want to delay 4.0 any more than we did so we decided to release it without it. The manual covers everything that will be in the Help file.

• Find and Replace is not implemented yet.

• Precision over 1000 digits was not implemented in direct mode (it does work when a stand-alone application is created).

FREE UPGRADE TO 4.01

You should have already received a postcard advising you of these problems. You can receive a free upgrade to version 4.01 by sending us your ZBasic 4.0 master diskette and your correct address. The upgrade will be available in six to eight weeks and will be shipped to you by first class mail.

Changes to Event trapping in version 4.0

As some of you have already noticed there is a slightly different "feel" to the event handling in version 4.0. This is the new "correct way" of doing event trapping according to "Inside Macintosh".

The older versions of ZBasic would handle events in a different manner by "Turning off" events when an event trap occurred. This solved many problems for novice programmers but created others in the area of missed events and lost window updates and such.

The new version corrects the problem nicely and events are no longer lost.

A system error 28 (stack overflow) will occur with some older programs if events are not handled correctly. An example of the error follows:

Old Format (causes System Error 28 with v 4.0)

```
MENU ON: MOUSE ON: DIALOG ON
ON DIALOG GOSUB "Routine 1"
ON MENU GOSUB "Routine 2"
ON MOUSE GOSUB "Routine 3"
'Main Event Loop"
GOTO "Main Event Loop"
'
' Error 28 occurs because events are
' still active and start piling up
' on the stack in version 4.0
```

New Format required in version 4.0:

```
MENU ON: MOUSE ON: DIALOG ON
ON DIALOG GOSUB "Routine 1"
ON MENU GOSUB "Routine 2"
ON MOUSE GOSUB "Routine 3"
'Main Event Loop"
GOTO "Main Event Loop"
MENU OFF: MOUSE OFF: DIALOG OFF
```

As you can see, all that is required is to turn off the events after the event loop so that they are not active when you exit the main event loop. If the event trapping is active at this time, new events are loaded on the stack eventually causing a Stack Overflow error (system error 28).

You figured out how to hide the Menu Bar!??

Wouldn't it be nice to create a window that's as big as the screen? Until now that has been an unknown secret hidden in the gloomy depths of "Inside Macintosh".

Thanks to "Z" reader, Anthony Oresteen, the problem is solved. We tried it on a Mac Plus, an SE with a Radius monitor and a Mac II and it works great. Have fun!

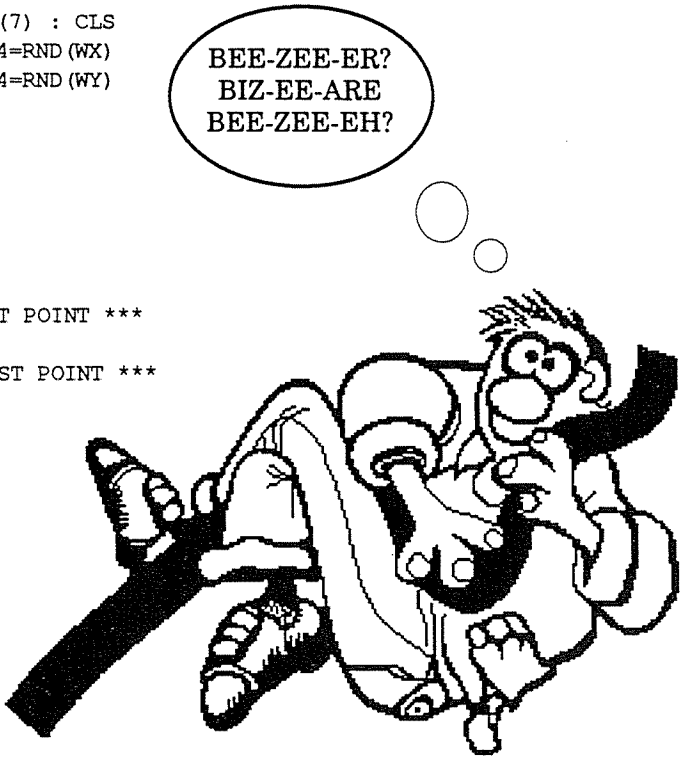
```
REM   This program draws a window over the menu region
REM   so you can use the whole Macintosh screen for your
REM   programs.
REM   Program submitted by Anthony Oresteen, August 1987.
'
CLS: COORDINATE WINDOW: WINDOW OFF
'
VSIZE%= PEEK WORD(PEEK LONG(&904)-116)       ' Vertical size of screen
HSIZE%= PEEK WORD(PEEK LONG(&904)-114)       ' Horizontal size of screen
'
GRAY&=PEEK LONG(&9EE)                        ' Gets gray region of desktop
'
CALL OPENRGN                                 ' Saves region data
OLDDESKTOP&=FN NEWRGN                        ' Gets a new handle for region
CALL COPYRGN (GRAY&, OLDDESKTOP&)            ' Gets a copy of standard desktop
CALL SETRECTRGN(GRAY&,0,0, HSIZE%, VSIZE%)   ' Sets desktop to full screen
'
WINDOW#1 ,, (0,0)-(HSIZE%,VSIZE%),3          ' Sets Window to full screen size
COORDINATE 1023, 767
BEEP
BOX FILL 0,0 TO 1023,767:
TEXT,24,,2: PRINT : PRINT "FULL SCREEN WINDOW!!"
DO
UNTIL LEN(INKEY$)
'
WINDOW CLOSE #1                              ' CLOSE FULL SCREEN WINDOW
CALL COPYRGN(OLDDESKTOP&, GRAY&)             ' Restores normal desktop region
CALL PAINTRGN (GRAY&)                        ' Draws region
CALL DISPOSERGN(OLDDESKTOP&)                 ' Throw away the old handle
CALL DRAWMENUBAR                             ' Restores menu bar
'
WINDOW#1 ,,(0,40)-(HSIZE%,VSIZE%-20),257     ' Sets Window to full screen size
TEXT 0,12
PRINT :PRINT "SMALL WINDOW WITH MENU BAR BACK AGAIN!!!"
DO
UNTIL LEN(INKEY$)
END
```

BEZIER CURVES

A ZBasic programmer sent in a program he had created to do Bezier Curves using floating point math. This type of graphic curve calculation routine is used in programs like Adobe's Illustrator and other CAD/CAM and graphics applications. The problem was that the curve calcuation took too much time in floating point. Andrew provided a LongInteger version of the routine that operates about ten to fifteen times faster. Try it out.

Converting to two byte integer is a problem. The resolution would be limited to 256 points so it would look pretty nasty. Any ideas?

```
"BIG LOOP"
  COORDINATE WINDOW : WX=WINDOW(6) : WY=WINDOW(7) : CLS
  XC1=RND(WX) : XC2=RND(WX) : XC3=RND(WX) : XC4=RND(WX)
  YC1=RND(WY) : YC2=RND(WY) : YC3=RND(WY) : YC4=RND(WY)
  KM%=100 : REM *** # OF POINTS ***
  CIRCLE XC1,YC1,10 : CIRCLE XC1,YC1,2
  CIRCLE XC2,YC2,10 : CIRCLE XC2,YC2,2
  CIRCLE XC3,YC3,10 : CIRCLE XC3,YC3,2
  CIRCLE XC4,YC4,10 : CIRCLE XC4,YC4,2
  PLOT XC2,YC2 TO XC3,YC3
  FOR SS=1 TO 2 : COLOR SS=1
    OX=XC1  : OY=YC1 : PLOT OX,OY : REM *** 1ST POINT ***
    ON SS GOSUB  "OLD BEZIER",  "NEW BEZIER"
    PLOT TO XC4,YC4                : REM *** LAST POINT ***
    BEEP : BREAK ON : COLOR -1
  NEXT SS
  BREAK OFF: DELAY 500
GOTO "BIG LOOP"
'
"OLD BEZIER"
  NBS!=1./KM% : T!=NBS! : RU!=.5
  DO
    T2!=T!*T!
    T3!=T2!*T!
    NC1!=1-3*T!+3*T2!-T3!
    NC2!=3*T3!-6*T2!+3*T!
    NC3!=3*T2!-3*T3!
    NC4!=T3!
    X=NC1!*XC1 + NC2!*XC2 + NC3!*XC3 + NC4!*XC4 + RU!
    Y=NC1!*YC1 + NC2!*YC2 + NC3!*YC3 + NC4!*YC4 + RU!
    IF ABS(X-OX)>1 OR ABS(Y-OY)>1 THEN PLOT TO X,Y:OX=X:OY=Y
    T!=T!+NBS!
  UNTIL T! >= 1
RETURN
'
"NEW BEZIER"
  RU&=S&>>1 : S&=32768 : KX&=(S&*S&+(KM%>>1))/KM%
  T&=(KX&+RU&) >> 15
  DO
    T2&=(T&*T& + RU&) >> 15 : T3&=(T2&*T& + RU&) >> 15
    NC1&=S& - 3*T& + 3*T2& - T3&
    NC2&=3*T3& - 6*T2& + 3*T&
    NC3&=3*T2& - 3*T3&
    NC4&=T3&
    X=(NC1&*XC1 + NC2&*XC2 + NC3&*XC3 + NC4&*XC4 + RU&)/S&
    Y=(NC1&*YC1 + NC2&*YC2 + NC3&*YC3 + NC4&*YC4 + RU&)/S&
    IF ABS(X-OX)>1 OR ABS(Y-OY)>1 THEN PLOT TO X,Y:OX=X:OY=Y
    T&=(T&*S& + KX& + RU&) >> 15
  UNTIL T& >= S&
RETURN
```



$100 REWARD

We need example programs that use ZBasic AppleTalk commands. We will pay $100 for the best program that completely demonstrates the use of AppleTalk. Send programs on disk to: AppleTalk/Z, 4500 E. Speedway, Suite 22, Tucson, AZ 85712-5305. Programs become the property of Zedcor. All submissions of useable code will get next ZBasic upgrade for free.

Dear Dr. Z,
I am especially pleased with the way the new MSDOS version of ZBasic is evolving, having just received my 4.01 update. It will be absolutely super when it is completely debugged and probably meet 90 percent of my programming needs.

However, I find that some of my most useful programs are TSR utilities, such as SuperKey. *(TSR=Terminate and Stay Resident. ed.)*

I could greatly benefit for a number of special needs with dedicated TSR utilities, especially if they would not require as much memory as some of the commercially available multi-purpose utilities. At this point my only alternative appears to be to learn assembly (or perhaps C) language so that I can write my own TSR programs.

My question is this; Is there a way to make an MSDOS ZBasic program into a TSR activated by a "Hotkey"? If that is possible, ZBasic would probably meet 99 percent of my needs.

Thank you for your kind attention to this matter. I look forward to your response.

Bob Chubon
Columbia, SC

Dear Bob,
This is an interesting question. Having talked with my fellow programmers we have come to the conclusion that it is indeed possible. To explain it however will take us until the next issue. Hold on until then and we will have a special column about this.

Dear Dr. Z

Dear Dr. Z,
I have several questions:

1. When we are in a BASIC program is there any way to dump the screen without pressing a key?:

```
10 CLS
...
300 CIRCLE
....
400 DUMP?
```

2. When will you offer a version for the Atari ST or Amiga?

3. Is there a command or technique to use the whole screen without viewing the menu bar?

4. Will you be offering a special version for the Mac II or the Apple IIGS?

5. You should support "Keys" in file searching like in Pascal and Cobol.

6. We use the TOPS network to transfer files between Macintosh and IBM. We have a problem, the files transferred have

extra characters. TOPS is a fantastic and cheap network. Try it.

We distribute your ZBasic in Greece and believe your newsletter is getting better with each new issue.

Manolis Milonakis
Athens, Greece

Dear Mr. Milonakis,

Your answers in order...

1. With Macintosh and IBM you can use the PAGE LPRINT command. With the MSDOS version make sure you have loaded the "GRAPHIC.COM" program that comes with MSDOS.

With Macintosh you will have no problems except when using a LaserWriter. The screen dump facility has problems with some versions of the system. In this case you could store your text and graphics in a "PICT" by modifying your example program and then "dumping" it to the printer with ROUTE 128:

```
CLS
PICTURE ON: CALL SHOWPEN
  CIRCLE 250, 300, 300
PICTURE OFF
'
A&=PICTURE          REM Store in A&
'
DEF PAGE
DEF LPRINT:         REM Printer setup
'
ROUTE 128:          REM Route to printer
  PICTURE , A&
ROUTE 0:            REM Back to screen.
```

2. When the economics dictate that we spend the time to create versions for these computers, we will. The news that we have is that neither computer is going to survive the Apple / IBM war. (Do you agree? ed.)

3. See the article in the Mac section of this newsletter.

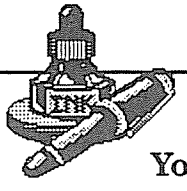
4. The Mac version will encompass the Mac II color and sound capabilities early next year. We have not decided on whether to persue a complete Apple IIGS version with toolbox calls and such. What do our readers think? We're going to the AppleFest in San Franciso this September. We'll let you know after that.

5. We are looking at providing special BTree file support as an option in a short while. Will keep you informed.

6. We've heard a lot of good things about TOPS. The problem you are having is that some computers provide an extra Linefeed or Carriage return at the end of a line. To strip off this extra character create a small ZBasic program:

```
OPEN "I",1,"OriginalFile"
OPEN "O",2,"NewFile"
WHILE EOF(1)=0
  LINEINPUT#1, A$
  A$=LEFT$(A$,LEN(A$)-1)
  PRINT#2, A$
WEND
CLOSE#1: CLOSE#2
```

That should do the trick.



You can write Dr. Z, c/o Zedcor, 4500 E. Speedway, Suite 22, Tucson, AZ 85712-5305.

Genesis

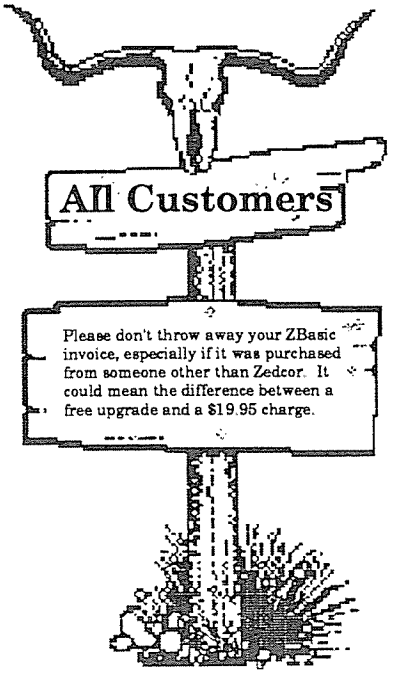
In the beginning the Project manager created the Programming staff. The programming staff was without form and structure. And the Project Manager said, "let there be Organization"; and there was Organization. And the Project Manager saw that Organization was good; and the Project Manager separated the workers from the Supervisors, and he called the supervisors: "Management", and he called the workers : "exempt".

And the Project Manager said, "Let there be a mission in the midst of the Organization, and let it separate the workers, one from another." And the Project Manager created the mission and he called it : "The System". And the Project Manager separated those who were to benefit from The System, from those who were to build it. And he called the former: "Users", and he called the latter: "Programmers".

And the Project Manager said, "Let all the Programmers in the Organization be gathered in one place, and let a Chief Programmer be brought up to lead them". And it was so. And the Project Manager saw that he was competent.

And the Project Manager said unto the Chief Programmer, "Create for me a schedule, so that I may look upon the schedule and know the Due Date." And the Chief Programmer went among his staff and consulted with them. And the staff was divided into two parts; one part was called: "Analysts", and the other part was called: "Application Programmers". And the Analysts went back to their desks and estimated,as was their custom. And it came to pass that each Analyst brought his estimate to the Chief Programmer, whereupon he collected them, summarized them, and drew a PERT chart.

And the Chief Programmer went unto the Project Manager and presented to him the estimate saying "It shall take ten months." And the Project Manager was not pleased and said, "I have brought you up from the depths of the staff; but you have not grasped the Big Picture." And the Project Manager hired consultants, and authorized overtime, and he said to the Chief Programmer, "Behold, see all I have done! The Due Date will be in five months." The Chief Programmer was much impressed and went from the Project Manager and proceeded to implement The System.



And the Chief Programmer sent his Analysts to the Users and said, "Let Specifications be written." And there were meetings, and lunches, and telephone calls. And the Specifications were written. And there was a Payday and the Happy Hour; one month.

And the Chief Programmer examined the specifications and saw that they were too ambitious. And he separated the mandatory features from the optional features; and he called the mandatory features: "Requirements", and he called the optional features: "Deferred", and the Users called him names. And the Chief Programmer gave the Specifications to the Analysts and said, "Let the Requirements be analyzed and let the files be designed." And it was so. And the Chief Programmer said, "Let the Software Houses put forth their Salesmen, and let us have a Data Management System." And it was so. The Software Houses brought forth all things for them, each according to his own file structure. And it came to pass that a Data Management System was selected; and the Chief Programmer saw that it was good. And there was a Payday and the Happy Hour; a second month.

And the Chief Programmer said, "Let the system be divided into parts, and let each part be called a 'Module'. And let programming teams be formed and let each be assigned to write a Module." And it was so. And the Chief Programmer created the programming teams with two levels, a greater and a lesser; and he called the greater the "Senior Programmers", and he called the lesser the "Junior Programmers". and he gave the greater dominion over the lesser. And the Chief programmer saw that it was good. And the Junior programmers saw it differently. And there was a Payday and the Happy Hour; a third month.

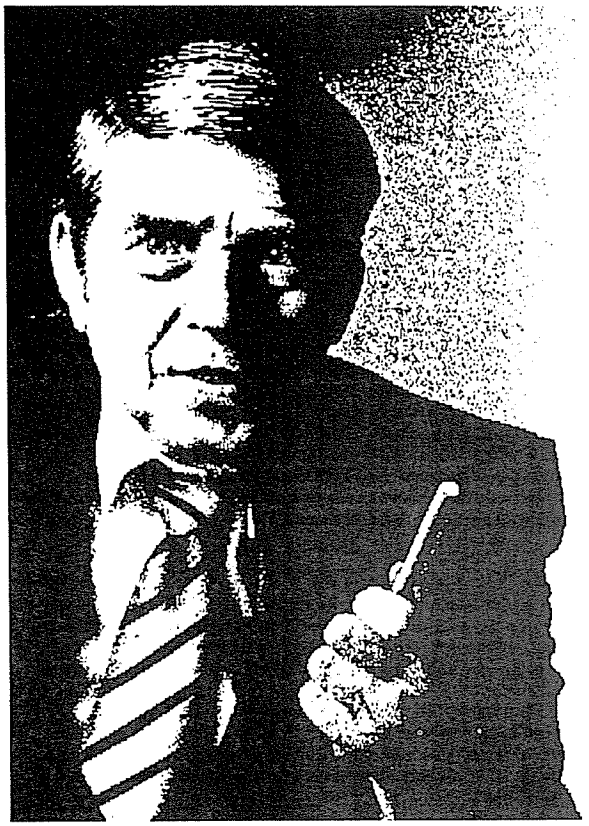
And the Chief Programmer said, "Let the programming be started and let much overtime be consumed, for there is but two months left." And the Programmers, both the Senior and the Junior, were much afraid, they strove to please the Chief Programmer. And they flowcharted, and they coded, each in his own fashion. And the Chief Programmer looked upon the work and liked it not. And the Chief Programmer said, "Let there be a Standard. And the programmers looked upon the Standard and liked it not. And there was a Payday and the Happy Hour; a fourth month.

And the Chief Programmer said, "Let there be Progress Reports, so we can monitor and control", and there were Progress Reports. And the Chief Programmer looked upon the Progress Reports and saw that the Due Date was not to be met. And the Chief Programmer arose, pressed his suit, shaved his beard, and went unto the Project Manager, and groveled. And the Chief Programmer pointed his fingers, and caused Blame to issue forth upon all manner of creatures who sold Hardware and Software. And the chief Programmer asked for an extension.

And the Project Manager was exceedingly angry, and cast doubts upon the Chief Programmer's ancestry; and uttered a multitude of threats. But it came to pass the Extension was granted; and the Chief Programmer took the Extension back to the programming teams, and there was much rejoicing. And there was a Payday and the Happy Hour; a fifth month.

And the Chief Programmer said, "Let the programming Modules be integrated, one with another, so that System Testing may begin." And it was so. Two by two the Modules were integrated. one with another. And great difficulties were experienced, and many hours of overtime were used, and many cups of coffee were consumed. And it came to pass that System Testing was completed. And there was a Payday and the Happy Hour; a sixth month.

Then the Chief Programmer did go to the Project Manager and said unto him, "Behold, I bring you good tidings of joy which will come to the Users; for on this day The System is completed." And suddenly there was with them a multitude of Users praising the Chief Programmer and saying, "Glory be to the System in the highest, but could you make one small change??"



Letters to the Editor

To the Editor,
The latest issue is excellent! While it did cost $5.00, which some of your readers consider expensive, it has more of value to me than BYTE and several other magazines. As for the others, 80% of the storage space is for advertising! Keep up the good work!

I think modems are neat and setting up RS-232 helps insomnia (although it does not cure it). However, we don't intend to get a modem and we think paper still has great value. So don't limit significant information and programs to a BBS. Let us know what is available via "Z". $5 or $10 for a disk full of programs is very reasonable.

Subroutines with Arguments: Isn't this what a LONG FN is? It would be more elegant to distinguish between procedures and functions (as Pascal does) but it's not vital. Maybe you could change the format: X=FN PROCEDURE_X when you have no use for X is tacky. CAPS/lower case for GLOBALS/locals is fine.

By responding to many of the letters, you may move us in an evil direction. Is ZBasic better than QuickBASIC??? It sure is on our Apple and on our TRS-80! Wasn't that one of your goals to make ZBasic transportable? As long as Quick, True, Real, Classic and other compilers only work on one kind of machine, a lot of us don't care how good they are.

William Horton
WDH Resources
RD 1, Box 138
Alfred Station, NY 14803

Dear William,
Thanks for the support! On of the strengths of ZBasic is that code is transportable. We also think we're the best on a one-to-one basis with ANY OTHER BASIC COMPILER ANYWHERE.

To the Editor,
A suggestion on the Tech-support line: some questions/problems are relatively simple. Perhaps a secretary or "level 1" type support person could screen calls and offer help. They could free up the phone for the regular support person.

A newsletter is undoubtably a lot of work and supplies a lot of information. But, problems and solutions can surface faster than four times per year. Perhaps a

monthly "bug notice" with fixes (either supplied, or available if a diskette is sent in).

In the last "Z" newsletter, there were several mentions of the Cauzin Stripper. I think it would be the ideal medium for propagating ZBasic programs and routines. Submissions would be easier also! Someone suggested accompanying newsletters with diskettes. How could you supply so many formats? One of ZBasic's big advantages is portability between machines, but unfortunately diskettes are not. Strips are. And considerably cheaper.

A few months ago, InCider magazine had a deal with Cauzin selling Strippers at a discount. I passed that up as I couldn't quite justify using it only with one magazine. If Zedcor could make such an arrangement, I would seriously consider a Stripper.

William J. Coohon
8235 Welter Rd.
Ovid, MI 48866

Dear William,

Thank you for your feedback. We have considered the CauzinStripper and have debated the issue here considerably. At this point we are going to "Wait and see".

As for a newsletter of more frequency... this also is being debated. Perhaps a smaller newsletter every two months? Perhaps an even smaller one every month? Say 2-3 pages?

To the Editor,

Business consultant Tom Peters has noted a decline of American Business due to the basic diregard most organizations hold for customers.

I'm pleased to not only exempt you from that category, but commend you for the continued service you have provided me.

It has become increasingly difficult to justify using any other BASIC compiler since I began with ZBasic a year and half ago.

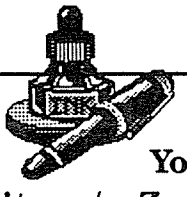
Version 4.0 includes features I specifically asked about. At the time, your staff provided effective answers to enable me to use the features requested (specifically; determining display type and accessing the command line). The latest version provides simple commands for these features.

Again, thank you for an excellent product and user support.

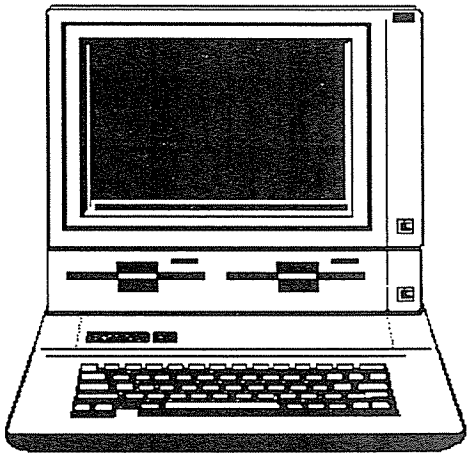
Peter Olivola
2036 N. Kenmore
Chicago, IL 60614

Dear Peter,

Thank you for your feedback and support. We are still working hard to give ZBasic new commands and features that people ask for. While this takes time, we still strive to make the customer happy.



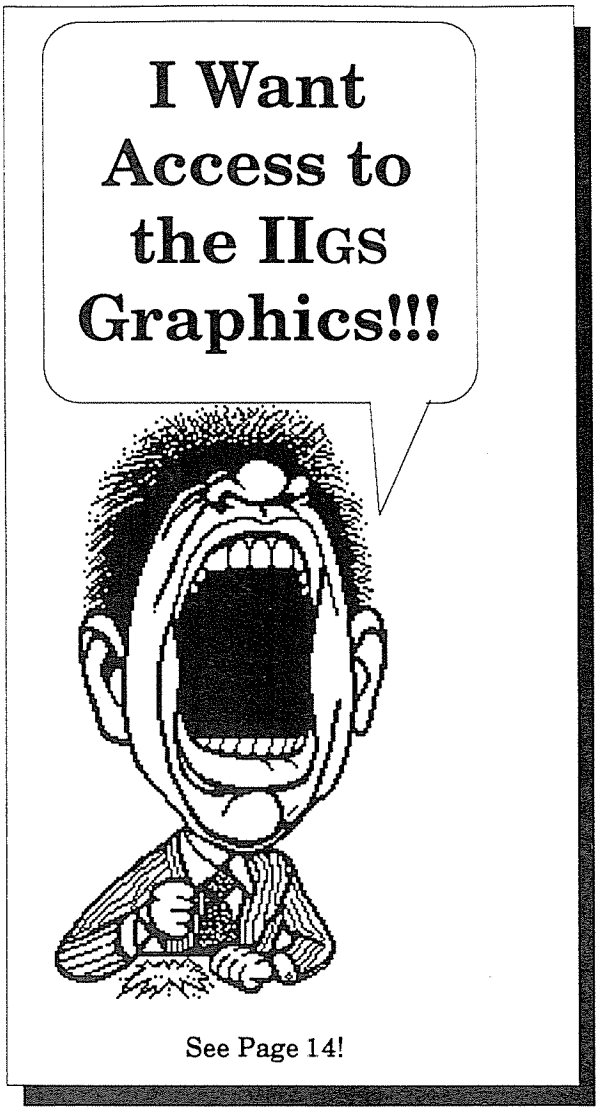
You can write to the Editor c/o Zedcor, 4500 E. Speedway, Suite 22, Tucson, AZ 85712-5305.



APPLE II ProDOS™ and DOS 3.3

The Apple DOS 3.3 version is being reworked now to bring it up to version 4.0 standards. The current version is 3.2. Hopefully version 3.9 will be released before the end of the year. It will be the final release of this version. It will contain bug fixes and a number of other features but will not have a full screen editor like the newer ones.

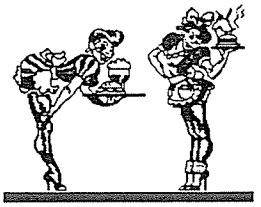
The Apple ProDOS version of ZBasic is selling like hotcakes. Version 4.02 is fully debugged (at least so far) and is getting great reviews. The following program lets you access pathnames:



See Page 14!

```
LONG FN GETPATH$(SLOT,DRIVE)
  REM This function will allow you to use a slot and drive specification
  REM to access a ProDOS device.  It will return the volume name of the
  REM device.
  :
  REM Convert slot and drive into ProDOS unit_num
  UNIT_NUM = (SLOT << 4) OR ((DRIVE - 1) << 7)
  POKE &1F01, UNIT_NUM : REM Place it in the parmlist
  POKE &1F00, 2 : REM 2 parms for the online call
  POKE WORD &1F02, &1F12 : REM use file name buffer
  REM Change the following &803 to &865 for 128K version
  MACHLG &A9, &C5, &20, &803 : REM Perform the ProDOS call (64K version)
  A% = VARPTR(PATH$) : REM Point to ZBasic string
  LNGTH% = PEEK(&1F12) AND &0F : REM Get length of volume name
  POKE A%, LNGTH% : REM Init ZBasic string length
  LONG IF LNGTH% <> 0 : REM Only store string if there's a string to store
    FOR I = 1 TO LNGTH%
      POKE A% + I, PEEK(&1F12 + I)
    NEXT I
  END IF
END FN = PATH$
```

MAIN BANK machine language subroutines in 128K ProDOS



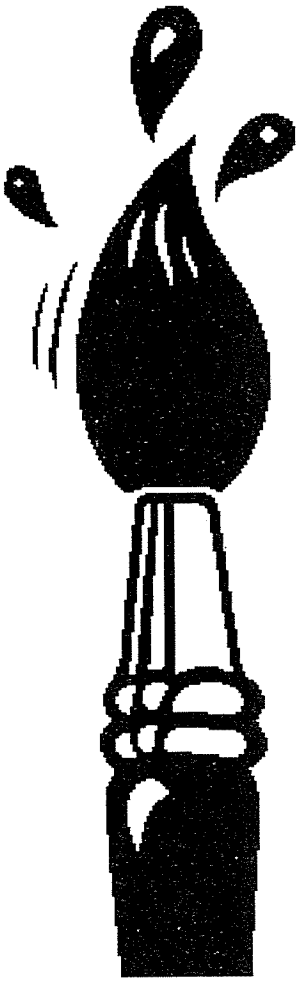
By Greg Branche

This handy little utility allows you to POKE or BLOAD machine language programs into variables and then execute them. Since the main program area of ZBasic is in the Aux bank this was a problem. This program fixes that:

```
REM Long function simply pokes address into caller
REM subroutine, and then calls it to switch banks
REM and call the main subroutine. This functions
REM should be used for cross-bank subroutine calls
REM in the 128K ProDOS version.
:
LONG FN CallMain(Address)
:
  REM First set up the actual address to call
:
  POKE WORD &F7,Address
:
  REM And call the subroutine
:
  CALL &F0
:
END FN
:
REM This sets up a subroutine on zero page to call
REM the real subroutine in the main bank of memory.
REM It MUST be done before the function is called.
:
REM Assembly language subroutine
REM
REM         STA   $C004
REM         STA   $C002  ;Turns on the main bank
REM         JSR   $0     ;subrtne addr goes here
REM SUBLOC EQU    *-2
REM         STA   $C005
REM         STA   $C003   ;This turns the aux bank on
REM         RTS
:
FOR I = &F0 TO &FF
  READ A
  POKE I,A
NEXT I
DATA &8D,4,&C0,&8D,2,&C0,&20,0
DATA 3,&8D,5,&C0,&8D,3,&C0,&60
:
```

Sample Program

```
00010 REM Long function simply pokes address
00011 REM into caller subrcutine, and
00020 REM then calls it to switch banks and
00021 REM call the main subroutine.
00030 :
00040 LONG FN CallMain(Address)
00050 :
00060   REM First set up the actual address to call
00070 :
00080   POKE WORD &F7,Address
00090 :
00100   REM And call the subroutine
00110 :
00120   CALL &F0
00130 :
00140 END FN
00150 :
00160 REM This sets up a subroutine on zero
00161 REM page to call the real subroutine
00170 REM in the main bank of memory.
00171 REM It MUST be done before the function
00180 REM is called.
00190 :
00200 REM The subroutine looks like this
00201 REM in assembly language
00210 REM
00220 REM         STA   $C004
00230 REM         STA   $C002  ;Turns on main bank
00240 REM         JSR   $0     ;addr of sub goes here
00250 REM SUBLOC EQU    *-2
00260 REM         STA   $C005
00270 REM         STA   $C003 ;This turns aux bank on
00280 REM         RTS
00290 :
00300 FOR I = &F0 TO &FF
00310   READ A
00320   POKE I,A
00330 NEXT I
00340 DATA &8D,4,&C0,&8D,2,&C0,&20,0,
00341 DATA 3,&8D,5,&C0,&8D,3,&C0,&60
00350 :
00360 REM This is just a sample subroutine
00361 REM that is poked into page 3 of
00370 REM the main bank.  It simply prints a CHR$(7)
00380 REM It would be just as easy to CALL
00381 REM a BLOADed subroutine
00390 :
00400 FOR I = &300 TO &304
00410   READ A
00420   POKE I,A
00430 NEXT I
00440 DATA &A9,&87,&4C,&ED,&FD
00450 :
00460 REM Call the function to call
00461 REM the subroutine in the main bank
00470 :
00480 FN CallMain(&300)
00490 :
00500 REM And now to prove that
00501 REM we're back where we started...
00510 :
00520 PRINT "Program terminated successfully!"
00530 END
```



Paint Program Utilizes IIGS Super-Hi-RES Graphics

Now you can see how programs can access the features of the IIGS graphics.

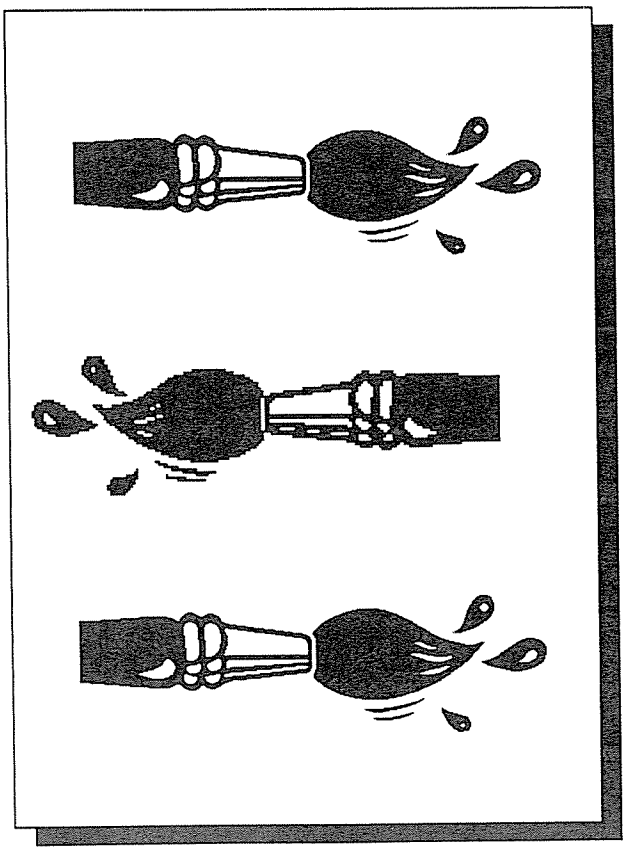
This program is fun and will give you some important insights.

Enjoy.

```
ON ERROR GOSUB "ERROR HANDLER"
MODE 0 : CLS : POKE $C035,$16 : VT = 1 : HT = 1
Color = 1
LINE INPUT ! &1, "Load picture? (Y/N) "; P$
P$ = UCASE$(P$)
LONG IF P$ = "Y"
  GOSUB "LOAD PIC"
  REM Either load a picture,
XELSE
  GOSUB "INIT SCREEN"
  REM or clear the screen
END IF
:
X = MOUSE(0) : REM Init mouse
:
REM - - - - - - - CHECK KEYBOARD - - - - - - - -
:
"MAIN LOOP"
KEY$ = INKEY$
LONG IF KEY$ = CHR$(27)
  POKE $C029, 1 : REM Switch back to text screen
  GOSUB "MENU"
  POKE $C029, $E1
  REM Switch to graphics screen again
END IF
IF KEY$ = " " THEN Color = (Color + 1) AND 15
REM  CHANGE COLOR
:
REM  - - - - - - GET X,Y FROM MOUSE - - - - - -
:
X = MOUSE(1) : Y = MOUSE(2) : S = MOUSE(3)
X = X \ 3.2
REM SCALE TO 320 X 200 COORDINATES FROM 1024 X 767
Y = Y \ 3.85
REM FORCE FLOATING POINT MATH
:
REM  - - - DRAW COLOR C AT X,Y OR MOVE CURSOR - - -
:
Mem = 8192 + Y * 160 + X / 2
REM Calculate pixel address
REM Get original byte into PIXEL
MACHLG &18
MACHLG &FB
MACHLG &08
MACHLG &C2, &10
MACHLG &AE, Mem
MACHLG &BF, 0, 0, 1
MACHLG &8D, PIXEL
MACHLG &28
MACHLG &FB
REM Calculate high or low nybble of byte
NYBBLE = X AND 1
LONG IF S = 1
  CHANGED = 1 : REM Set changed flag
  LONG IF NYBBLE = 0
    PIXEL = PIXEL AND $0F : REM Clear pixel value
    PIXEL = PIXEL OR (Color << 4) : REM set value
  XELSE
    PIXEL = PIXEL AND $F0
    REM pixel is in lower nybble
    PIXEL = PIXEL OR (Color AND $0F)
    REM set new pixel
  END IF
  MACHLG &18
  MACHLG &FB
  MACHLG &08
  MACHLG &C2, &10
  MACHLG &AD, PIXEL
  MACHLG &AE, Mem
  MACHLG &9F, 0, 0, 1
  MACHLG &28
  MACHLG &FB
END IF : REM Then flash cursor
MV = PIXEL : REM Save original pixel value
LONG IF NYBBLE = 0
  PIXEL = PIXEL XOR $F0
  REM complement color to flash cursor
XELSE
  PIXEL = PIXEL XOR $0F
END IF
REM set altered pixel value
MACHLG &18            : REM CLC
MACHLG &FB            : REM XCE
MACHLG &08            : REM PHP
MACHLG $C2, &10       : REM REP #$10
MACHLG &AD, PIXEL     : REM LDA PIXEL
MACHLG &AE, Mem       : REM LDX Mem
MACHLG &9F, 0, 0, 1   : REM STA $010000,X
MACHLG &28            : REM PLP
MACHLG &FB            : REM XCE
DELAY 50 : REM MAKE SURE IT FLASHES!
REM then restore original value
MACHLG &18            : REM CLC
MACHLG &FB            : REM XCE
MACHLG &08            : REM PHP
MACHLG &C2, &10       : REM REP #$10
MACHLG &AD, MV        : REM LDA MV
MACHLG &AE, Mem       : REM LDX Mem
MACHLG &9F, 0, 0, 1   : REM STA $010000,X
MACHLG &28            : REM PLP
```

```
MACHLG &FB            : REM XCE
GOTO "MAIN LOOP"
:
REM  - - - - - - - - - - - - - - - - - - - - - - -
REM                  LOAD PICTURE
REM  - - - - - - - - - - - - - - - - - - - - - - -
"LOAD PIC"
POKE $C029,1 : REM Display text screen
LOCATE HT-1, VT-1
CLS LINE
INPUT "Enter picture to load -> "; PIC$
POKE $C029, $E1 : REM Display graphics screen
CHANGED = 0 : Loading = -1
POKE WORD $1F01, VARPTR(PIC$)
POKE WORD $1F03, $8800 : REM I/O BUFFER
POKE $1F00, 3 : REM 3 PARMS FOR OPEN
MACHLG &A9, &C8, &20, &803, &90, 3, &20, &8C51
REM OPEN THE FILE
POKE $1F01, PEEK($1F05) : REM GET REF_NUM
POKE $1F00, 4 : REM 4 PARMS FOR READ
POKE WORD $1F02, $2000 : REM READ INTO $2000
POKE WORD $1F04, $2000 : REM READ $2000 BYTES
MACHLG &A9, &CA, &20, &803, &90, 3, &20, &8C51
REM DO THE READ
POKE WORD &3C, &2000 : POKE WORD &3E, &3FFF
POKE WORD &42, &2000
MACHLG &38, &20, &C311
REM MOVE PART 1 OF 4 TO AUX MEM
MACHLG &A9, &CA, &20, &803, &90, 3, &20, &8C51
REM READ PART 2
POKE WORD $3C, $2000 : POKE WORD $3E, $3FFF
POKE WORD $42, $4000
MACHLG &38, $20, $C311 : REM MOVE PART 2 TO AUX MEM
MACHLG &A9, &CA, &20, &803, &90, 3, &20, &8C51
REM READ PART 3
POKE WORD $3C, $2000 : POKE WORD $3E, $3FFF
POKE WORD $42, $6000
MACHLG &38, $20, $C311 : REM MOVE PART 3 TO AUX MEM
MACHLG &A9, $CA, $20, $803, $90, 3, &20, &8C51
REM READ PART 4
POKE WORD $3C, $2000 : POKE WORD $3E, $3FFF
POKE WORD $42, $8000
MACHLG &38, $20, $C311 : REM MOVE PART 4
POKE $1F00, 1 : REM 1 PARM FOR CLOSE
MACHLG &A9, &CC, $20, $803, $90, 3, $20, $8C51
REM CLOSE THE FILE
RETURN
:
REM  - - - - - - - - - - - - - - - - - - - - - - -
REM                  SAVE PICTURE
REM  - - - - - - - - - - - - - - - - - - - - - - -
"SAVE PIC"
IF LEN(PIC$) = 0 THEN "SAVE AS"
Loading = 0
POKE WORD $3C, $2000 : POKE WORD $3E, $3FFF
POKE WORD $42, $2000
MACHLG &18, $20, $C311
REM MOVE PART 1 OF 4 TO MAIN MEMORY
POKE WORD $1F00, 3 : REM 3 PARMS FOR OPEN
POKE WORD $1F01, VARPTR(PIC$) :
REM POINT TO FILENAME
POKE WORD $1F03, $8800 : REM USE I/O BUFFER AT $8800
MACHLG &A9, $C8, $20, $803, $90, 3, $20, $8C51
REM OPEN THE FILE
POKE $1F01, PEEK($1F05) : REM GET REF_NUM
POKE $1F00, 4 : REM 4 PARMS FOR WRITE
POKE WORD $1F02, $2000 : REM WRITE FROM $2000
POKE WORD $1F04, $2000 : REM WRITE $2000 BYTES
MACHLG $A9, $CB, $20, $803, $90, 3, $20, $8C51
REM WRITE PART 1
POKE WORD $3C, $4000 : POKE WORD $3E, $5FFF
POKE WORD $42, $2000
MACHLG &18, $20, $C311 : REM MOVE PART 2
MACHLG $A9, $CB, $20, $803, $90, 3, $20, $8C51
REM WRITE PART 2
POKE WORD $3C, $6000 : POKE WORD $3E, $7FFF
POKE WORD $42, $2000
MACHLG &18, $20, $C311 : REM MOVE PART 3
MACHLG &A9, $CB, $20, $803, $90, 3, $20, $8C51
REM WRITE PART 3
POKE WORD $3C, $8000 : POKE WORD $3E, $9FFF
POKE WORD $42, $2000
MACHLG &18, $20, $C311 : REM MOVE PART 4
MACHLG $A9, $CB, $20, $803, $90, 3, $20, $8C51
REM WRITE PART 4
POKE $1F00, 1 : REM 1 PARM FOR CLOSE
MACHLG $A9, $CC, $20, $803, $90, 3, $20, $8C51
REM CLOSE THE FILE
POKE $C029, $E1 : CHANGED = 0 : P$ = "Y"
RETURN
:
REM  - - - - - - - - - - - - - - - - - - - - - - -
REM                  INIT SCREEN
REM  - - - - - - - - - - - - - - - - - - - - - - -
"INIT SCREEN"
POKE &C029, &E1
MACHLG &18
REM CLC                  ;put 65816 into native mode
MACHLG &FB REM XCE
MACHLG &C2, &30          : REM REP #$30
;16-bit registers and accumulator
MACHLG &A9, &0, &0
REM LDA #0               ;clear accumulator
MACHLG &A2, &7E00
REM LDX #$7E00     ;clear this many bytes
MACHLG &9F, &1FFE, &01
REM STA $011FFE,X  ;clear the graphics buffer
MACHLG &CA               : REM DEX
MACHLG &CA
REM DEX           ;decrement twice for words!
MACHLG &D0, &F8
REM BNE -8         ;loop through entire buffer
MACHLG &E2, &30          : REM SEP #$30
;8-bit registers/accumulator
MACHLG &38
REM SEC            ;back to emulation mode
MACHLG &FB               : REM XCE
RESTORE
FOR I = 0 TO 31 STEP 2
  READ T
  POKE WORD $2000 + I,T
NEXT : REM  SET PALETTE #0
POKE WORD $3C, $2000 : POKE WORD $3E, $201F
POKE WORD &42, $9E00
MACHLG &38, $20, $C311
REM Move Palette #0 into aux memory
RETURN
DATA  0,&777,&841,&72C,&F,&80
DATA  &F70,&D00,&FA9,&FF0,&E0
DATA  &4DF,&DAF,&78F,&CCC,&FFF
:
REM  - - - - - - - - - - - - - - - - - - - - - - -
REM                  PRODOS MENU
REM  - - - - - - - - - - - - - - - - - - - - - - -
"MENU"
CLS
LONG IF LEN(PIC$) > 0
  FOR I = LEN(PIC$) TO 1 STEP -1
    A$ = MID$(PIC$,I,1)
    IF A$ = "/" THEN PSN = I : I = 1
```

```
  NEXT I
  A$ = RIGHT$(PIC$,LEN(PIC$)-PSN)
XELSE
  A$ = ""
END IF
PRINT "FILE : "; A$
PRINT "COLOR:"; Color
PRINT @(11, 6) "1) Load picture"
PRINT @(11, 8) "2) Save picture"
PRINT @(11,10) "3) Save picture as..."
PRINT @(11,12) "4) Go back to picture"
PRINT @(11,14) "5) Clear screen"
PRINT @(11,16) "6) Quit": POKE $C029, 1
VT = 19 : HT = 6
B$ = "123456" + CHR$(27)
DO
  A$ = INKEY$
UNTIL INSTR(1, B$, A$) > 0
IF A$ = CHR$(27) THEN "GO BACK"
ON VAL(A$) GOTO "LOAD IT", "SAVE PIC", "SAVE AS",
"GO BACK", "CLEAR IT"
REM Quit here!
GOSUB "CHANGED?"
CLS : MODE 2 : END
:
"CHANGED?"
LONG IF CHANGED = 1
  LOCATE 6, 19 : CLS LINE
  PRINT "Save picture? (Y/N)";
  DO
    P1$ = UCASE$(INKEY$)
  UNTIL INSTR(1, "YN", P1$) > 0
  IF P1$ = "Y" THEN GOSUB "SAVE PIC"
END IF
RETURN
:
"LOAD IT"
GOSUB "CHANGED?"
VT = 20 : HT = 1
GOTO "LOAD PIC"
:
"SAVE AS"
LOCATE 0, 19 : CLS LINE
INPUT "Type name of picture -> "; NEWPIC$
LONG IF LEN(NEWPIC$) > 0
  PIC$ = NEWPIC$
  P$ = "Y"
  GOSUB "CREATE FILE"
  GOTO "SAVE PIC"
END IF
LOCATE 0,19 : CLS PAGE
RETURN
:
"CLEAR IT",
GOSUB "CHANGED?"
GOSUB "INIT SCREEN"
"GO BACK"
RETURN
:
REM  - - - - - - - - - - - - - - - - - - - - - -
REM               ERROR TRAPPER
REM  - - - - - - - - - - - - - - - - - - - - - -
"ERROR HANDLER"
POKE $C029, 1 : PRINT "ERROR = "ERROR
ERR = ERROR AND $FF : ERROR = 0
LONG IF ERR = 12 : REM DUPLICATE FILENAME ERROR
  PRINT @(4,22) "File exists.  Overwrite? (Y/N)"
  DO
    P1$ = UCASE$(INKEY$)
  UNTIL INSTR(1, "YN", P1$) > 0
  IF P1$ = "Y" THEN GOSUB "SAVE PIC"
  POKE $C029,$E1 : REM DISPLAY GRAPHICS
  RETURN "MAIN LOOP"
END IF
LONG IF ERR = 3 AND NOT Loading
REM FILE NOT FOUND ERROR
  GOSUB "CREATE FILE"
  GOSUB "SAVE PIC"
  POKE $C029, $E1
  RETURN "MAIN LOOP"
END IF
PRINT ERRMSG$(ERR)
PRINT "Press any key to continue..."
DO : UNTIL LEN(INKEY$)
RETURN "MENU"
:
REM  - - - - - - - - - - - - - - - - - - - - - -
REM                FILE CREATOR
REM  - - - - - - - - - - - - - - - - - - - - - -
"CREATE FILE"
POKE $1F00, 7 : REM 7 Parms for the create call
POKE WORD $1F01, VARPTR(PIC$) : REM Point to file-
name
POKE $1F03, $C3 : REM Allow full access
POKE $1F04, $C1 : REM Filetype = $C1
POKE WORD $1F05, 0 : REM Clear aux_type field
POKE $1F07, 1 : REM Seedling storage type
POKE WORD $1F08, 0 : REM Clear out creation date &
time
POKE WORD $1F0A, 0
MACHLG &A9, &C0, $20, $803, $90, 3, $20, $8C51
REM Call ProDOS
RETURN
```

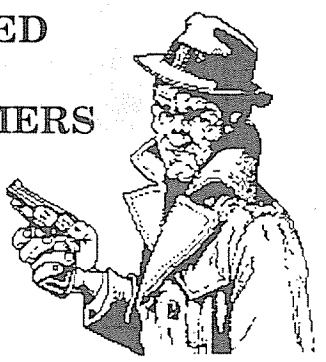


TO: ZBASIC Newsletter

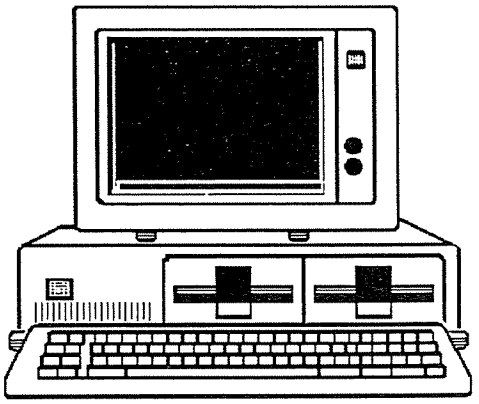
Attached are four nasty little "functions" from someone who thinks structured programming is about as much fun as an extra helping of boiled TOFU. They work on the Apple ProDOS versions of ZBasic; I don't know if they work on the DOS 3.3 version.

Bryon Bowe
P.O. BOX 928
West Acton, MA 01720

FOUR NEW WEAPONS FOR NON-STRUCTURED ProDOS PROGRAMMERS



```
REM  FN gosub(address)     Simlar to GOSUB (and CALL) but
REM                        branch to addresses as well as constants:
REM                        (1) FN gosub(100)     --->CALL 100
REM                        (2) FN gosub(LINE 100) --->GOSUB 100
REM                        (3) jmp=LINE "Prompt"
REM                            FN gosub (jmp)    --->GOSUB "Prompt"
REM
REM  FN goto (address)     Simlar to GOTO but
REM                        branch to addresses as well as constants:
REM                        (1) FN goto(LINE 100)--->GOTO 100
REM                        (2) jmp=LINE "Prompt"
REM                            FN goto (jmp)     --->GOTO"Prompt"
REM
REM  FN pop                POPs one level off the stack (just like
REM                        Applesoft's POP command:
REM                        (1) FN pop            --->POP (Applesoft)
REM                        (2) FN pop
REM                            GOTO "Promt"      --->RETURN"Promt" (ZBasic)
REM
REM  FN return (level)     Returns through multiple levels:
REM                        (1) FN return (0)     --->Does nothing
REM                        (2) FN return (1)     --->RETURN
REM                        (3) FN return (2)     --->POP: RETURN
REM                        (4) FN return (3)     --->POP:POP:RETURN
REM                        (5) FN return (4)     --->POP:POP:POP:RETURN
:
LONG FN gosub (XXX_address%)
  POKE WORD$02, XXX_address%: MACHLG $6C, $02, $00
END FN
:
LONG FN goto (XXX_address%)
  POKE WORD $02, XXX_address%: MACHLG $68, $68, $6C, $02, $00
END FN
:
LONG FN pop
  MACHLG $68, $AA, $68, $A8, $68, $68, $98, $48, $8A, $48
END FN
:
LONG FN return (XXX_level%)
  POKE WORD $02, XXX_level%: MACHLG $68, $68, $C6, $02, $D0, $FA
END FN
```



MSDOS™

4.01 Upgrade Notice

As you all probably know, version 4.0 was released with some serious problems. These were corrected in version 4.01 which you should all have by now.

Well, lo-and-behold, we have discovered problems with 4.01. These problems are fixed with the following patch (which pops you up to the current version 4.01p).

If you version already has a "p" after it do not do these patches. SEE PATCH PROGRAM PAGE 20!

We will be releasing version 4.02 in six to eight weeks. You can recieve a FREE UPGRADE by just sending in your master diskette. We will return a new diskette when 4.02 is ready at no charge.

NOTICE TO ALL USERS OF ZBASIC VERSION 4.01

Since releasing V4.01 on August 26th (yes, we started shipping immediately), we have been discovering Basic Unwanted Grievances (in other words, BUGs) and correcting them as quickly as we can. The following is a list of what we've found so far, and instructions on patching your copy of ZBasic to correct the problems.

First of all, a warning! In the following fixes/patches, if the original value of the byte shown by the system does not match the value shown in the "Original Value" column, DO NOT MAKE THE PATCH! What this probably means is that your copy of the software has already been fixed (either by patching or reassembly) and modifying it could have disastrous results. In addition, make these patches only to a COPY of your ZBASIC.COM file. I will not be held responsible for any inadvertant trashings of master disks.

PROBLEM: ZBasic does funny things after reconfiguring the sizes of floating point variables. (Exact symptoms are undefined.)

FIX: You CANNOT use the P)atch option from within ZBasic, since by the time you get to it, the damage has already been done. You must use DEBUG, a program that is supplied with every copy of MSDOS. Follow the instructions exactly (the <cr> represents the carriage return key):

```
What you type:                  DEBUG responds:

DEBUG ZBASIC.COM<cr>            - (it's prompt)
E CS:C1A2<cr>                   xxxx:C1A2  30._
2E<cr>                          -
W<cr>                           Writing D090 bytes
                                -
Q                               (this quits DEBUG)
```

For the following patches, you can use the P)atch command available at the ZBasic startup screen. Instructions on using PATCH can be found on page B-15 of the ZBasic manual (the same instructions apply to the MS-DOS version).

PROBLEM: The FIX command will cause the system to hang if a null (i.e. blank) line is followed by a line containing 32 bytes (27 characters/tokens + 5 header bytes).

FIX: Use the PATCH option from the main ZBasic title screen to enter the following patch:

Address	Original Value	New Value
&55ED	&E3	&80
&55EE	&F9	&E9
&55EF	&80	&05
&55F0	&E9	&74
&55F1	&05	&F6

PROBLEM: A compiled program does not perform a signed integer comparison correctly.

FIX: Enter the following patch to change the opcode table that the compiler uses to generate the comparison:

Address	Original Value	New Value
&BD2A	&73	&7D
&BD2E	&77	&7F
&BD30	&76	&7E
&BD34	&72	&7C

PROBLEM: The ZBasic runtime system does not update the default segment value after chaining (for statements such as POKE, PEEK, CALL, etc.).

FIX: Use a DEF SEG statement at the beginning of any chain module that performs direct memory manipulation, such as PEEKs, POKEs, etc. This will reset the default segment back to the program's DATA segment.

PROBLEM: If a non-line-numbered line is deleted while in the screen editor, other non-line-numbered lines could be deleted also.

FIX: Enter the following patch, using the P)atch option on the main startup screen:

Address	Original Value	New Value
&77D9	&E8	&90
&77DA	&E1	&90
&77DB	&DE	&90

PROBLEM: Using the VARPTR function to retrieve the address of an integer array element will return the incorrect address, and may cause the system to crash.

FIX: Due to the complexity of the problem, there is no patch, but there is a work around. This consists of using VARPTR to retrieve the address of a simple integer variable, and then calculating the address of the array variable from this. The following program demonstrates this method.

```
00010 DEFINT A, I, X
00020 DIM A, A(10): ' Use "A" to access 'A(*)'
00030 Z = MEM D    : ' Point to integer var segment
00040 PRINT HEX$(VARPTR(A)) : ' Prints address 'A'
00050 PRINT HEX$(Z),: ' Prints seg of integer vars
00060 A = VARPTR(A) : ' Retrieves address of 'A'
00070 A = A + 2
00071  ' Integers 2 bytes long, 'A' points to A(*)
00080 PRINT HEX$(A)
00081 ' This prints the address of A(0)
00090 FOR I = 1 TO 5
00100   A(I) = I * 100
00101   ' Just some stuff to PEEK at!
00110   X = A + (2 * I): 'Calc address each element
00120   PRINT HEX$(Z); ":"; HEX$(X); "-";
00130   PRINT PEEK WORD (X),
00140   PRINT A(I) : ' Print value (should be same)
00150 NEXT I
00160 END
```

PROBLEM: There is a major bug in chaining. When a program chains to another module, the runtime system fails to update some system variables that are used to manage the variable segments. This problem will most likely show up if you attempt to chain a second time within the same execution cycle (i.e. without exiting the program).

FIX: The following patch requires the use of the 30 byte patch area that is mentioned on page A-21 of the ZBasic manual. If you are currently using that area of memory for something else, you will have to make a choice between this patch and your own. The additional patch area (at &1B9) is not affected by this patch.

Address	Original Value	New Value
&0169	&00	&A1
&016A	&00	&64
&016B	&00	&02
&016C	&00	&2E
&016D	&00	&A3
&016E	&00	&C6
&016F	&00	&02
&0170	&00	&A1
&0171	&00	&68
&0172	&00	&02
&0173	&00	&2E
&0174	&00	&A3
&0175	&00	&C8
&0176	&C0	&02
&0177	&00	&8C
&0178	&00	&D0
&0179	&00	&2B
&017A	&00	&06
&017B	&00	&19
&017C	&00	&03
&017D	&00	&2E
&017E	&00	&A3
&017F	&00	&CA
&0180	&00	&02
&0181	&00	&E9
&0182	&00	&6A
&0183	&00	&07
&43FC	&F0	&6B
&43FD	&C4	&BD

PROBLEM: With CONVERT TO UPPER CASE and SPACES BETWEEN KEYWORDS both configured to YES, the editor will attempt to convert digits within variable names to upper case (in other words, funny characters will be produced).

FIX: If you must have both options configured to YES, then don't use digits within variable names. If you must use digits within variable names, then don't have both options configured to YES at the same time (in other words, "DON'T DO DAT!").

PROBLEM: Communications buffers can not be set larger than 255 bytes.

FIX: Patch the following locations.

Address	Original Value	New Value
&4B86	&BA	&58
&4B87	&47	&EB
&4B88	&46	&16
&4B97	&75	&90
&4B98	&0C	&90

PROBLEM: COM (port) ON and COM (port) OFF communications commands do not work.

FIX: Patch the following locations.

Address	Original Value	New Value
&46C2	&75	&E4
&46C3	&06	&21
&46C4	&BA	&75
&46C5	&FC	&04
&46C6	&02	&24
&46C7	&EB	&F7
&46C8	&04	&EB
&46C9	&90	&11
&46CA	&BA	&24
&46CB	&FC	&EF
&46CC	&03	&EB

&46CD	&B0	&0D
&46CE	&01	&A9
&46CF	&EE	&01
&46D0	&C3	&00
&46D1	&A9	&E4
&46D2	&01	&21
&46D3	&00	&75
&46D4	&75	&04
&46D5	&06	&0C
&46D6	&BA	&08
&46D7	&FC	&EB
&46D8	&02	&02
&46D9	&EB	&0C
&46DA	&04	&10
&46DB	&90	&E6
&46DC	&BA	&21
&46DD	&FC	&B0
&46DE	&03	&20
&46DF	&B0	&E6
&46E0	&09	&20
&46E1	&EE	&C3

PROBLEM: System will not save itself after configuration, nor will it save ASCII source files. This has been tracked down to a bad batch of disks that had twelve apparently random bytes altered during the duplicating process.

FIX: Patch the following locations.

Address	Original Value	New Value
&3BE0	&06	&20
&3BE1	&00	&75
&3BEA	&00	&80
&3EE0	&54	&AE
&3EE1	&00	&FB
&3EEA	&00	&02
&3F00	&0F	&FA
&3F01	&00	&3E
&3F0A	&00	&03
&3F80	&00	&08
&3F81	&00	&F6
&3F8A	&00	&74

These are all of the known problems so far (optimistic, ain't I). If anybody comes up with anything new, please let me know as soon as possible. If I find anything new, I'll post updates here as soon as I have a fix.

I have updated my ASM files, and will produce a version 4.02 sometime within the next 6-8 weeks. Within that time, I'll be on the lookout for any other problems that pop up. In the meantime, send in your ZBasic master disk. When 4.02 becomes available, we'll update your master disk and send it back to you at no charge.

I have uploaded the source code for a ZBasic program that will run through your ZBASIC.COM file and make all the patches for you. Simply download the program from the library, load it into ZBasic, then RUN it. Once you are done with the patches, make sure you keep the program around somewhere. We'll 'patched'). This is so that we will be able to differentiate from patched and non-patched versions.

I'm sorry about the inconvenience that we all have gone though. I dearly hope that the end result is well worth it to everybody.

Greg Branche
Product Manager
Apple/IBM Division

4.01 TO 4.01p Patch Program

If you don't like Patching programs, this program is available on GEnie and CompuServe and may be downloaded to patch your version 4.01 until 4.02 comes out. It's in the IBM library and is named ZBUGS.ZBS.

```
00010 MODE 2 : CLS
00020 PRINT "PLEASE ENTER COMPLETE PATHNAME "
00021 PRINT "TO FILE TO BE PATCHED:"
00030 INPUT "-> "; FILE$
00040 INPUT "OUTPUT TO PRINTER? "; A$
00050 A$ = LEFT$(A$,1) : IF A$ = "Y" THEN ROUTE 128
00052 A$=MID$(FILE$, 2,1)
00054 LONG IF A$=":"
00060   DR$ = LEFT$(FILE$,2)
00065 END IF
00061  FILE$ = RIGHT$(FILE$, LEN(FILE$)-2)
00070 FOR I = LEN(FILE$) TO 1 STEP -1
00080   A$ = MID$(FILE$,I,1)
00090  IF A$ = "\" THEN PSN = I : I = 1
00100 NEXT I
00110 LONG IF PSN <> 0
00120   PTH$ = LEFT$(FILE$,PSN)
00130   FILE$ = RIGHT$(FILE$,LEN(FILE$) - PSN)
00140 XELSE
00150   PTH$ = PATH$(0)
00160 END IF
00170 OLDPTH$ = PATH$(0) : CHDIR PTH$
00180 PRINT "THE FOLLOWING BYTES WERE"
00181 PRINT "NOT PATCHED BECAUSE:"
00190 PRINT TAB(5) "1>  ORIGINAL VALUE <> CURRENT"
00191 PRINT " VALUE" : PRINT "AND"
00200 PRINT TAB(5) "2>  CURRENT VALUE  <> NEW
00201 PRINT "VALUE" : PRINT
00210 FILE$ = DR$ + FILE$
00220 OPEN "R", 1, FILE$, 1
00230 PRINT "ADDRESS","OLD","CURRENT","NEW"
00240 READ BYTE, OLD, NEW
00250 WHILE BYTE <> 0
00260   RECORD #1, BYTE-&100
00270   READ #1, A$;1
00280   LONG IF (ASC(A$)<>OLD) AND (ASC(A$)<>NEW)
00290     PRINT HEX$(BYTE), HEX$(OLD),
00291     PRINT HEX$(ASC(A$)), HEX$(NEW)
00300   XELSE
00310     LONG IF ASC(A$) <> NEW
00320       A$ = CHR$(NEW)
00330       RECORD #1, BYTE - &100
00340       WRITE #1, A$;1
00350     END IF
00360   END IF
00370   READ BYTE, OLD, NEW
00380 WEND
00390 CLOSE #1
00400 CHDIR OLDPTH$
```

```
00410 ROUTE 0
00420 END
00430 :
00440 ' PATCH TO FIX RECONFIGURATION OF FLOATING
POINT VARIABLES
00450 :
00460 DATA &C1A2, &30, &2E
00470 :
00480 ' PATCH TO FIX 'FIX' COMMAND
00490 :
00500 DATA &55ED, &E3, &80
00510 DATA &55EE, &F9, &E9
00520 DATA &55EF, &80, &05
00530 DATA &55F0, &E9, &74
00540 DATA &55F1, &05, &F6
00550 :
00560 ' PATCH TO FIX SIGNED INTEGER COMPARISON
00570 :
00580 DATA &BD2A, &73, &7D
00590 DATA &BD2E, &77, &7F
00600 DATA &BD30, &76, &7E
00610 DATA &BD34, &72, &7C
00620 :
00630 ' PATCH TO FIX DISAPPEARING EDITOR LINES
00640 :
00650 DATA &77D9, &E8, &90
00660 DATA &77DA, &E1, &90
00670 DATA &77DB, &DE, &90
00680 :
00690 ' PATCH TO FIX CHAINING
00700 :
00710 DATA &0169, &00, &A1
00720 DATA &016A, &00, &64
00730 DATA &016B, &00, &02
00740 DATA &016C, &00, &2E
00750 DATA &016D, &00, &A3
00760 DATA &016E, &00, &C6
00770 DATA &016F, &00, &02
00780 DATA &0170, &00, &A1
00790 DATA &0171, &00, &68
00800 DATA &0172, &00, &02
00810 DATA &0173, &00, &2E
00820 DATA &0174, &00, &A3
00830 DATA &0175, &00, &C8
00840 DATA &0176, &00, &02
00850 DATA &0177, &00, &8C
00860 DATA &0178, &00, &D0
00870 DATA &0179, &00, &2B
00880 DATA &017A, &00, &06
00890 DATA &017B, &00, &19
00900 DATA &017C, &00, &03
00910 DATA &017D, &00, &2E
00920 DATA &017E, &00, &A3
00930 DATA &017F, &00, &CA
00940 DATA &0180, &00, &02
00950 DATA &0181, &00, &E9
00960 DATA &0182, &00, &6A
00970 DATA &0183, &00, &07
00980 DATA &43FC, &F0, &6B
00990 DATA &43FD, &C4, &BD
01000 :
01010 ' PATCH TO FIX COM ON / COM OFF BUG
01020 :
01030 DATA &46C2, &75, &E4
01040 DATA &46C3, &06, &21
01050 DATA &46C4, &BA, &75
01060 DATA &46C5, &FC, &04
01070 DATA &46C6, &02, &24
01080 DATA &46C7, &EB, &F7
01090 DATA &46C8, &04, &EB
01100 DATA &46C9, &90, &11
01110 DATA &46CA, &BA, &24
01120 DATA &46CB, &FC, &EF
01130 DATA &46CC, &03, &EB
01140 DATA &46CD, &B0, &0D
01150 DATA &46CE, &01, &A9
01160 DATA &46CF, &EE, &01
01170 DATA &46D0, &C3, &00
01180 DATA &46D1, &A9, &E4
01190 DATA &46D2, &01, &21
01200 DATA &46D3, &00, &75
01210 DATA &46D4, &75, &04
01220 DATA &46D5, &06, &0C
01230 DATA &46D6, &BA, &08
01240 DATA &46D7, &FC, &EB
01250 DATA &46D9, &EB, &0C
01260 DATA &46DA, &04, &10
01270 DATA &46DB, &90, &E6
01280 DATA &46DC, &BA, &21
01290 DATA &46DD, &FC, &B0
01300 DATA &46DE, &03, &20
01310 DATA &46DF, &B0, &E6
01320 DATA &46E0, &09, &20
01330 DATA &46E1, &EE, &C3
01340 :
01350 ' PATCH TO FIX COM BUFFER SIZE BUG
01360 :
01370 DATA &4B86, &BA, &58
01380 DATA &4B87, &47, &EB
01390 DATA &4B88, &46, &16
01400 DATA &4B97, &75, &90
01410 DATA &4B98, &0C, &90
01420 :
01430 ' PATCH TO UPDATE VERSION NUMBER
01440 :
01450 DATA &6AE4, &20, &70
01460 DATA &C768, &20, &70
01470 DATA &C769, &42, &20
01480 DATA &C76A, &61, &42
01490 DATA &C76B, &73, &61
01500 DATA &C76C, &69, &73
01510 DATA &C76D, &63, &69
01520 DATA &C76E, &20, &63
01530 DATA &C76F, &49, &20
01540 DATA &C770, &6E, &49
01550 DATA &C771, &74, &6E
01560 DATA &C772, &65, &74
01570 DATA &C773, &72, &65
01580 DATA &C774, &70, &72
01590 DATA &C775, &69, &70
01600 DATA &C776, &6C, &69
01610 DATA &C777, &65, &6C
01620 DATA &C778, &72, &65
01630 DATA &C779, &01, &72
01640 DATA &C77A, &20, &01
01650 :
01660 ' PATCHES TO FIX BIT SHIFTS DURING DUPLICATION
01670 :
01680 DATA &3BE0, &06, &20
01690 DATA &3BE1, &00, &75
01700 DATA &3BEA, &00, &80
01710 DATA &3EE0, &54, &AE
01720 DATA &3EE1, &00, &FB
01730 DATA &3EEA, &00, &02
01740 DATA &3F00, &0F, &FA
01750 DATA &3F01, &00, &3E
01760 DATA &3F0A, &00, &03
01770 DATA &3F80, &00, &08
01780 DATA &3F81, &00, &F6
01790 DATA &3F8A, &00, &74
01800 :
01810 DATA 0,0,0
```

PALLETTE Demonstration in EGA Mode

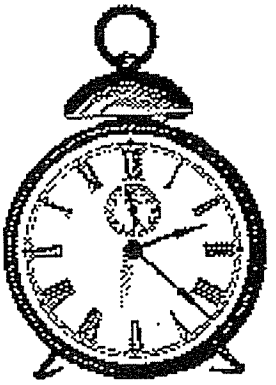
The following little program demonstrates the pallettes in the mode 19 use of the EGA board. It displays each pallette as a rainbow of color, and shows how different selections change the pallette colors.

The program presented here shows 4 pallettes, the first 16 of 64, second 16, third 16 and fourth 16 groupings. To have more fun, change the variable D by another increment by replacing the 1 in line 60 with some other e.g. 5 or 7 and run it again.

The background can also be changed by substituting another number for the 9 in line 80.

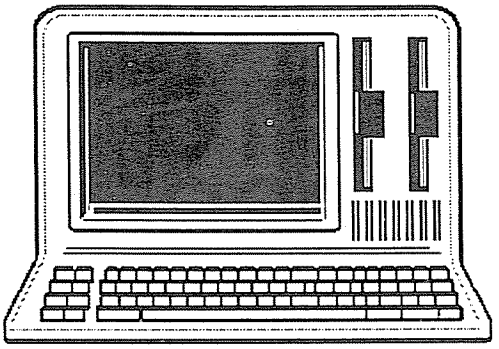
This program was submitted by Randall J. McClelland, M.D.

CHANGE IN TECHNICAL SUPPORT HOURS



The New Hours are
11:00 AM to 4:00 PM
Mountain Standard Time

```
00010 MODE 19                              'EGA Mode
00020 D=0
00030 FOR A=1 TO 4
00040   FOR B=0 TO 15
00050     PALLETTE B,D
00060     D=D+1                  'Change the 1 for variations
00070     IF D>64 THEN D=D-64
00080     COLOR B,9              'Change the 9 for other backgrounds
00090     C=B*80
00100 '   CIRCLE FILL C+250, C+150, C+50
00110     PRINT% (700,700)"# ";B+1;" OF PALLETTE ";A
00120     PRINT%(700,720) "COLOR";D
00130     TRON X
00140   NEXT B
00150  DELAY 2000
00160  CLS
00170 NEXT A
```

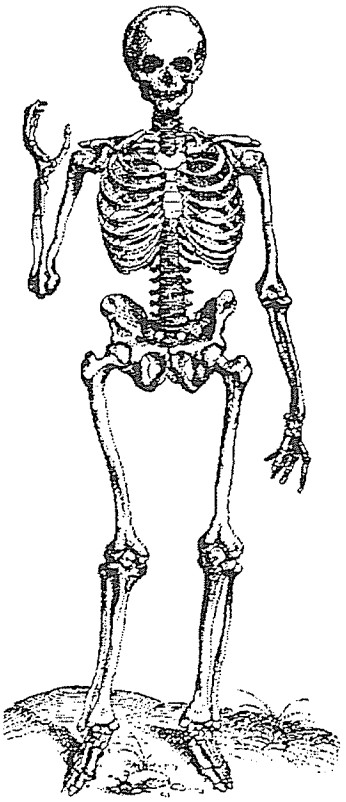


Z80 Computers

CP/M™ and TRS-80™

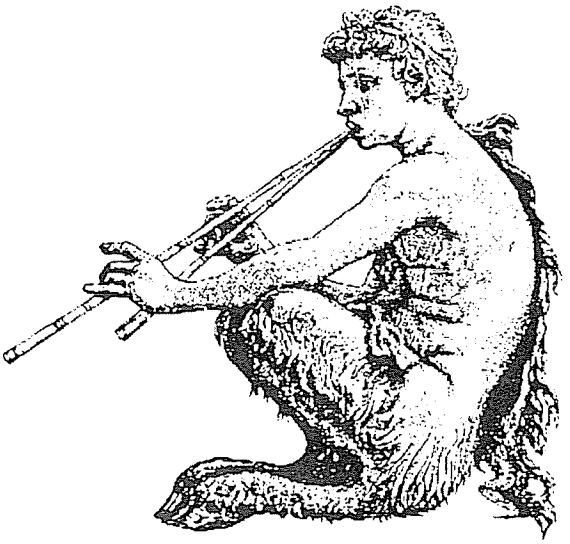
We don't have a lot of new stuff for the Z80 folks this issue.

As usual we request that anyone with special routines send them in to us for printing.



Are the Z80 computers fading away?
Tell us why not?! *We certainly hope not (ed).*

Sound Generation Program for the TRS-80 Model 4



Gentlemen:

I am offering the following subroutine to readers of "Z" to enhance the sound capabilities of the TRS-80 Model 4:.

```
"TONE"
  FOR X=1 TO DR
    OUT 144,1
    FOR XX=1 TO TN
    NEXT XX
  OUT 144,0
  NEXT X
RETURN
```

Variable DR controls the duration of the tone, while variable TN controls the pitch of the tone. I use this in many of my game programs and have used it to generate some simple music. Have fun with it.

I enjoy the newsletter but of course I would like to see some more applications for the model 4.

Sincerely,

Douglas Stone
1135 E. 24th Ave.
Albany, OR 97321

Thanks Doug! We'd like to provide more examples for the TRS-80 and CP/M systems. If you and others keep sending them in, we'll keep printing them. (ed)

WE WANT YOU TO SUBSCRIBE TODAY!

Name_____

Address_____

City_____ State_____ ZIP_____

Daytime Phone_____

Credit card number _____ Exp.____

Signature_____

$19.95 One year___ $37.00 Two years____ (quarterly publication)
(please add $5.00 for Canadian and Overseas subscriptions)

Mail Order to:

"Z" Newsletter
4500 E. SPEEDWAY, SUITE 22
TUCSON, AZ 85712-5305

The "Z" newsletter provides you with important information about your language of choice. Just one or two of the programs and ideas provided can save you many hours of work and frustration.

Only $19.95 per year (save $5 off the newstand price) and only $37 for two years (save $10). Send in the coupon or call our toll free order line to subscribe with your credit card.

1-800-482-4567

"Z" Newsletter
ZEDCOR, INC.
4500 E. SPEEDWAY, SUITE 22
TUCSON, AZ 85712-5305

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

SEND TO: