UpTime

The newsletter for: RS-DOS, OS9, OSK, CoCos, and 68xxx's.

Volume 2, Issue 12 FINAL ISSUE

Editorial

Welcome to the 24th and final issue of *UpTime*! *UpTime* was originally started shortly before the demise of the long-heralded cornerstone of the CoCo Community, the *Rainbow*. When it became obvious that the Rainbow would no longer be able to support its own existance due to the declining advertising and user base, the idea for *UpTime* was born as a way to give the CoCo Community a newsletter about what was new and happening if the *Rainbow* were to cease publication. *UpTime* was never designed as a profit-based publication, but rather an informational newsletter and a base from which to continue advertising CoCo and OSK products. Now that several other newsletters have taken hold, *UpTime* will conclude with this final issue. Any remaining subscription credits will be transferred to *68 'Micros*, an excellent publication that also covers the CoCo and OSK machines. Each two remaining issues of your *UpTime* subscription will be transferred as a one issue credit to *68 'Micros*. If you already have a subscription, your subscription will be extended by the same amount. Enclosed with this issue is a statement of your account— please check to be sure that this is correct and inform us of any problems so that they can be resolved quickly. For more information about **FARNA Systems**, their advertisment is contained elsewhere in this issue.

Here's an excerpt from our Product Hotline listing for various clubs and dealers who offer subscription based products either on printed or magnetic media:

Name: **68Micros**
Description: Newsletter published every six weeks featuring various columns and features, including a hardware corner, C programming, Q&A, and classifieds, starting with the August 1993 issue
Status: In good standing
Cost: $23/yr
Distributor: **FARNA Systems**, 904 2nd Ave., Warner Robins GA 31098-1029, *(912)-328-7859*

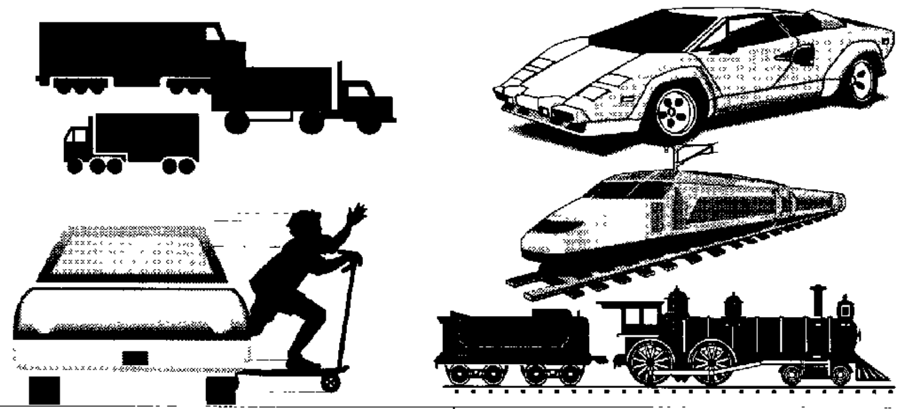
Name: **OS-9 Underground**
Description: Newsletter dedicated to OS-9 products and information, published sporadically, would recommend contacting the dealer for more information
Status: Have had past problems- contact first
Cost: unknown
Distributor: 4650 Kahuenga Blvd. Suite #7, Toluca Lake CA 91602, *(818)-761-4135*

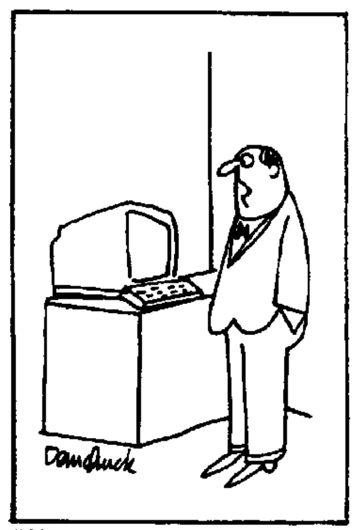
Name: **CoCo Trader**
Description: Newsletter for buy/sell/trading of any items or information. Reasonable ad rates. Write for more information.
Cost: $2.50 / 1/8-pg ad
Distributor: **CoCo Trader**, 2861 Easy St., Sevierville TN 37862

Name: **MI&CC Diskletter**
System Req.: CoCo 3, 128K
Description: Newsletter on disk for Mid-

Editorial cont'd on page 12

On The Move...





'Well, Mac ... What's It Gonna Be Today?... Are You With Me or Against Me?'

In this issue...

Thank you for your continued support throughout the years!

UpTime

Editor: Jordan Tsvetkoff

UpTime is published 12 times a year by **JWT Enterprises**, 5755 Lockwood Blvd., Youngstown, OH, 44512. Yearly subscription rates are $15.00, or $7.50 in two installments. Foreign orders (except Canada and Mexico) add $7.00 postage.

Postmaster: Send address changes to *UpTime*, **JWT Enterprises**, 5755 Lockwood Blvd., Youngstown, OH, 44512. *Readers:* Direct all subscription inquiries, payments, and changes of address to *UpTime Subscriptions*, **JWT Enterprises**, 5755 Lockwood Blvd., Youngstown, OH, 44512.

Purpose: To provide information about products, services, and activities relating to the **Tandy Color Computer** and **OSK/68xxx-based machines**.

Basic09 in ?? Easy Steps

As promised in the last installment, this time we will look at the commands embedded in the Disk ROM of DECB and how to deal with them under Basic09. The disk ROM lives up to its name in more ways than one. Not only is it (physically) situated inside your floppy disk controller so you won't be able to use it until you have (upgraded to) a disk based system, it also is almost entirely devoted to executing commands dealing with access to disk drives.

Under OS-9, this whole concept becomes obsolete because OS-9 doesn't use any of the ROMs inside your system. All code dealing with disk access and functions is spread out over a variety of managers, device drivers, device descriptors, and utilities. This isn't done to make the system hard to understand for people, but to make it easily adaptable to a variety of hardware setups.

For Basic09, this means that all commands you give it that need access to a disk (or other input or output device) are not handled by Basic09 itself but passed on to OS-9. There are basically three ways in which you can tell your application to do a certain job, although sometimes one of them is certainly preferred over the others.

The first way is to use commands embedded in Basic09. For example OPEN, CREATE, DELETE, etc. In this case Basic09 will gather and check the information neccessary to execute an OS-9 system call and pass the information on. OS-9 becomes completely transparant since the user has no way of knowing whether Basic09 executed its own code or something else.

The second way is to use the SysCall utility. As described in part three of the series, you can use this utility by issuing a **RUN syscall** statement. Your program must supply and check the information passed to OS-9 for executing a certain system call. Although this is a little more work, it also allows you access to functions not accessible through corresponding Basic09 statements, thus complementing and enhancing Basic09.

The third way is through the use of Basic09's SHELL command. In this case, OS-9 starts a new shell to execute whatever command Basic09 passes on to it. This way is usually preferred for launching other processes, calling and/or loading programs and utilities, etc.

And now for the conversions. Following is a list of commands that, under OS-9, are included in the system as separate programs called utilities: BACKUP, COPY, DIR, KILL, LOAD, MERGE, RENAME and SAVE.

Generally speaking, you will use these commands most often from the OS9: prompt. However, there may be cases in which you will want to run them from within a program, such as when your application uses temporary files during execution and has to clean up afterwards. Under DECB you would rename a file as follows: **RENAME oldfile TO newfile**.

In Basic09 this becomes: **SHELL "rename oldfile newfile"**. In this example "oldfile" and "newfile" are presumed to be literal names. If they represent string variables, the command would look like this: **SHELL "rename "+oldfile+" "+newfile**. Of course, your program must take care of two things here: A) the variables must contain valid names, and B) your current working directory is the directory where these files are located.

This second problem doesn't exist with DECB because it keeps all of its files in one directory, which is always both your working and execution directory. Under OS-9, generally speaking, your working directory holds your data files, while your execution directory holds your programs, system utilities, etc. When you boot up, OS-9 will set the execution directory to CMDS and your working directory to the root directory of your bootup disk. On a floppy system this is typically called /d0.

Although DECB has no equivalents, I do want to mention the CHX and CHD commands here. One uses these commands under OS-9 (as well as from within Basic09) to reset the system's directory pointers. Suppose you have a collection of programs in a separate directory called "programs". For OS-9 to be able to run them, it must be able to find them. You can tell OS-9 where it can find the programs by typing (in this case): **CHX /d0/programs**. CHD works in the same way, but resets the pointer to the working directory. For instance: to access a directory called "textdata" on drive /d1 you would type **CHD /d1/textdata**. Once you have set the directory pointer in this way, you can access all files within that specific directory by just typing their name rather than having to type an entire pathlist to one of those files.

In a way one could say that CHX/CHD are the OS-9 equivalents of DECB's DRIVE command. They are just a little more complex to use because OS-9's directory system is more complicated.

With regard to the commands mentioned above, I want to point out two more things. OS-9 and Basic09 do not support the LOADM and SAVEM commands. The reason for this is that OS-9 can load programs from any language as long as the programs' module header is recognized by OS-9. This module header contains, among other things, a code for the language in which the program is written. This code is checked by OS-9 before it tries to run the program, but not when it loads a program. So a simple LOAD command will do for all applications.

The second point is that Basic09 does recognize the KILL command but not for deleting files. KILL is used by Basic09 to unlink modules that it no longer needs from its workspace. This is very useful when you write programs too large to fit into the CoCo's 64K address space. If you want to delete a file from within Basic09, you must use the DELETE command.

There are a number of commands that have been dealt with in previous articles so I won't do a repeat of them: CLOSE, EOF, GET, INPUT, OPEN, PRINT, PUT, RUN and WRITE.

CVN and MKN$ are not supported. OS-9 uses a technique often called streaming to access disks. In simple terms, this means data transfer without format checking, so there is no real need for conversions.

On the point of conversions I would like to point out that if you write a program that needs to share its (numerical) datafiles with programs written in languages other than Basic09, you may want to store numbers as strings. The reason for this is simple: If you store a real

QUALITY SOFTWARE
AT
DOWN TO EARTH PRICES

#GS01 - Pensave Graphic System
Disk Includes:
- Pen and Paint 3-Doodle Program
- Pensave-HS2 Drawing Program
- PN640-HS4 Drawing Program
- Coco Colorbook-HS2 Paint Program
- PPrint-Graphics Print Program
- Plus 6 Graphics

#GF01 - Graphic Disk for Pensave
Disk Includes:
- Penload-HS2 & HS4 Graphic Loader
- SRcolor-Loads and Paints Sunrise
- Plus Graphic Files-J3CUB, JT00, AL, ALT00, TWONTREE, PA1921, DOGSDAY, C7BULLET, and SUNRISE

Req: 128K Coco III & Disk Drive
$3.95 per disk or $7.95 for both
S&H-$1.00 U S. and Canada, $3.00 other
Check or Money Order
MO residents add 4.725% sales tax

Southern Missouri Softworks
HCR7, Box 218
Doniphan, MO 63935
(314) 255-3722

JWT Enterprises

UpTime

A monthly magazine for CoCo's and OSK machines. **In each issue:** Find out about new products and upgrades for your favorite computers. ☞ Learn about what new technical breakthroughs are on the horizon. ☞ Discover what programs really can do and what they cannot. ☞ Published monthly in newsletter format. *One Year Subscription, two installments of $7.50; Canadian Orders, two installments of $9.00; Foreign Orders, 2 X $11.00 Can also be paid in one installment*

UpTime Back-Issues

From September 1992. Limited supplies. *$1.50 each; Foreign Orders, add $1.25 each; Bulk orders of six or more, $1.25 each; Foreign Orders add $1.00 each*

UpTime Volume 1 Pack

All 12 issues of UpTime's first year of publication are contained in a three-ring folder for their protection. If you missed them the first time around, now is your last chance as there are only a limited number available. September 1992 - August 1993. *Volume One Pack, $18.00; Canadian Orders, $9.00; Foreign Orders, $11.00*

JWT Enterprises
5755 Lockwood Blvd.
Youngstown, OH 44512

Foreign postage excludes U.S. Territories and Canada. These products for OS-9 Level 2 on the CoCo 3. Sorry, no C.O.D.'s or credit cards: Foreign & Canadian orders, *please* use U.S. money orders. U.S. checks, allow 4 weeks for receipt of order. Ohio residents, please add 6% sales tax.

Looking For Full and Real CoCo Support?
Look to: Mid Iowa & Country CoCo's
"The UPGRADE" Disk Magazine!

- Is not hard copy, though it will dump to your printer.
- Displays first rate 16 color H2 graphics, with articles.
- Does keep you informed with news from around the country.
- Includes ads and recommendations of better dealers.
- Does have OS-9 articles concerning the CoCo. Has a Level II tutorial series and others. with extensive OS-9 library.
- Does have Basic programing technique article and tutorials.
- Does have a full variety of RS-DOS & OS-9 indepth articles and reviews with tips for all; where to buy printers, drives, and hardware
- Does have a free buy/sell CoCo hardware listing. With CoCos and drives plus the hard to find items.
- Some are double disk issues!

This is a News disk, not a software disk. That we leave to the "Mid Iowa & Country CoCo" library "available wherever you are at". You can select from a variety of the *best* available RS & OS Public Domain, Shareware, and Orphanware for only a filing, backup, and mailing fee. Plus a Christian software sub-chapter. ROM burn and other support to "all" members. Gosh what don't we offer? Bonus disks? Yes, we've had a bonus disk or two per year too!

Now in our ninth year, four as a national Diskmagazine! With "The UPGRADE National Diskmagazine", we've grown to be one of the largest CoCo outreaches! With over 250 satisfied subscribers in over 40 states and 5 provinces of Canada; plus Australia and England. Join with those who write: "*Here's my renewal, I don't want to miss an issue!*"
May we also serve you?

Your UPGRADE subscription includes:
• 1-year membership in MI&CC and • UPGRADE Diskmagazine subscription with
• bonus disks and double issues (Req: 128K CC3, w/1 drive, RGB or TV)
$16.00 US • $21 Canada • $31 Foreign Air

Say, "I saw it in **UpTime**" and receive,,,
an UPGRADE Disk plus added bonus disk via return mail!

Terry Simons (MI&CC Treas.), 1328 48th St.
Des Moines, IA 50311 (515) 279-2576 after 8:00 PM
Include your Phone & System information

NEW THIS FALL:

- much asked for (long awaited by some): multiple file transfers for OScopy/RScopy. New interface almost eliminates typing and allows you to transfer entire disks with a single command.
- 6309 processor: CoCoTop, Quickletter, and Accounting Level 2 have been updated to support the 6309 processor. New installation routines detect the 6309 and update programs and printer driver if necessary.
- CoCoTop version 2. Includes a programmable calendar, hotkey access of functions, one key repeat function, file marking for batch processing, and help screen. It also uses larger buffers to prevent problems with hard drives. $29.95 (version 2.1 $24.95). CoCoTop version 1 continues to be available. Upgrades to version 2: $10.
- Homepac09. Keep track of all your important things: finances, mailing addresses, telephone numbers, and computer programs. Also allows you to make and print blank forms. One package: $29.95

For more info and prices, call or write to Chris Dekker:
RR #4 Centreville NB E0J 1H0 Canada (506) 276-4841

number on disk, it has the same 5-byte format as in memory. Most other languages use either 4-byte and/or 8-byte formats to represent real numbers. So, even if you get a program to read the correct number of bytes, it may not have a clue about their meaning. If you store a number in string format it looks the same as you would enter it from the keyboard- a format that is universally recognized.

DSKINI is not recognized by OS-9/Basic09. Use FORMAT instead.

DSKI$ and DSKO$ are not supported by OS-9 either. OS-9 does have a low level disk access command but it works differently. You can get low level access to a disk under OS-9 by adding "@" to the drive name when you open a path to it (e.g. **OPEN #path, "/d0@":READ**).

The main difference with the DECB commands is that you still can not access a disk by referencing track/sector numbers. Instead you must specify what is called a Logical Sector Number or LSN. OS-9 converts this number into track/sector numbers using data contained in the device driver.

Using this command, the entire disk is regarded as a single file. On a DS 40 track disk, this file is 1440 sectors long (numbers: 0-1439). This approach makes for easy programming when you want to write a backup utility and it also works fine for reading DECB format disks. A drawback is that you can not access, for instance, MSDOS format disks this way on a standard setup. The reason for this is that addresses get incorrectly decoded by the device driver. To correct the problem, you will have to install an extra driver capable of handling the 512 byte sector format used by MSDOS.

All FIELD statements in your program can be replaced by TYPE statements. This essentially defines your record as a complex datastructure which can be transferred to and from files with the same GET/PUT commands as used in DECB. For example, **FIELD #1, 5 AS A1$, 10 AS A2$, 7 AS B$** would translate into:

```
TYPE record=A1$:STRING[5]; A2$:STRING[10]; B$:STRING[7]
DIM buffer:record
```

Note that the #1 (the buffer number) is not specified here since Basic09 returns a path number when you OPEN a path to the file. "Buffer" as defined in the DIM statement is simply a label to identify the memory space set aside for "record". The buffer that is alluded to by #1 in the FIELD statement does exist under OS-9 but is handled internally by OS-9 and completely transparant to the user.

The same is true for the FILES statement. Every time you OPEN a path to a file, OS-9 creates the necessary buffer(s), but this process is transparant to the user.

LOC and LOF are not supported by Basic09. This leaves basically two ways to deal with the problem of where you are. Your application can take care of it. Either by using an internal counter or by defining a byte (or two) as part of the record to hold that record's number. Another way of dealing with the problem is using OS-9's filepointer. For every open file of every program running OS-9 maintains a filepointer. This pointer always points to the next byte to be read from or written to in that file. Upon OPENing or CREAT(E)ing a file, the pointer is automatically set to 0. This means that your first access to a file always starts at the first byte in the file.

If you want to access a different part of the file you must use the command SEEK #path,bytenumber with 'bytenumber' being the exact location of the start of the next disk access.

However this doesn't help much if you want to know where you are. Basic09 doesn't have any direct commands built-in to tell you, but we can use a system call to read the current value of the filepointer. Assuming you have a data structure set up to mimic the 6809's registers the following code will do:

```
regs.a=path (path associated with file)
regs.b=5 \ RUN syscall($8D, regs)
filepointer=65536*regs.x+regs.u
```

The number of the record last read or written to is calculated as:

```
recordnumber=filepointer/SIZE(buffer)-1
```

Note that this line only works if the abovementioned TYPE and DIM statements are also included in the program.

LSET and RSET are not recognized by Basic09. Basic09 always left justifies a string in the space allocated to it. Note that if you do *not* expressly define a variable as a string of length 'x', but use a variable name ending with $; Basic09 sets this space to 32 characters. If a string is longer than its allocated space, Basic09 truncates it to make it fit and discards the rest. If a string is shorter than its allocated space Basic09 terminates it with a CHR$(255).

If you have to justify a string for output formatting use the PRINT USINGstatement with the following codes:

Sxx> (right justifies a string in a field xx chars wide)
Sxx< (left justifies the same setup)
Sxx^ (will center the text in the field)

The UNLOADcommand is not recognized by Basic09 either. The best way to prevent problems with paths to files being left open, which can happen, is to exit your program with a BYE statement instead of END. As I mentioned earlier in this series, BYE forces OS-9 to execute a *F$Exit* system call. Part of the function of this system call is to close the paths left open by a program, so it implies an UNLOAD command. BYE also exits Basic09 so the best way to implement it, is *after* you have debugged your code. Before you pack your code replace all END statements in the *primary* module with BYE. In this context "primary" refers to the module that runs all the other modules that are part of a program.

If you have problems with a program not closing paths it usually shows up as follows. After you have run and exited a program you want to delete a file the program accessed. After typing the command del filename, OS-9 doesn't delete the file but prints an **ERROR #253**.

VERIFY ON/OFF has no direct counterpart in OS-9. Although it is possible to turn write verification on and off under OS-9, you will have to get a utility from a third party source to do it. Rainbow (2/90) contained an article by Stephen Goldberg on the subject, which includes C source code for a similar program.

There is a more radical way of dealing with the problem: altering the drive's device descriptor and saving the changes in a new bootfile. This will boot up your system with write verification disabled which results in much speedier programs if they have to write to the disk fairly often. Note that this has no impact on the speed with which data is read from a disk.

Most hardware seems to be reliable enough that this is asafe way of speeding up programs without creating unpleasant surprises when its time to read the data back from the disk. If you want to go this route change, the byte at offset $1A (#26) in the floppy drive descriptors from goes 0 to 1. If you know how: fine, but it is somewhat beyond the scope of this article to explain the process in detail.

If your program was written under ADOS, JDOS, or similar implentations, you may run into commands like RATE, which OS-9 handles in the same way as VERIFY (reading a code byte from the device descriptor), and the FREE command whose duties are taken over by a utility with the same name. However, since I'm not familiar with these ROM's, I won't deal with them in this article. The one last thing I want to mention is the BAUD command which can be replaced by OS-9's TMODE (or XMODE) utility.

- Chris Dekker

Pythagoras

The similarity of K-Windows to CoCo 3 OS-9 windows and the appeal of the Pythagoras tree made Chris Dekker's gracious invitation to experiment with the program in a recent *UpTime* irresistable. The results of some minor experimentation follow.

Conditions:

I did this and timed it on my MM/1a, which, unlike the CoCo 3, has no special text windows. I just used standard output, cleared the screen, and before exiting turned the cursor back on. The original program (with those changes, and using **bgfx** instead of **gfx2**) came right up and ran in 21.3 seconds on the MM/1a.

(Just for the heck of it, I made the program select random palette registers instead of reading the same ones each time. Initializing the palette array takes a negligible amount of time compared with drawing circles, so it makes no significant difference in run time.)

If you want to use the code from the accompanying listings on a CoCo, you'll either have to prearrange an appropriate window or add the code from the original listing that opens a window, performs a **DWSet** on it, selects it, and closes it at the end, change **bgfx** to **gfx2**, and add the path parameter to the graphics calls to specify the selected window.

Easy Changes:

Some variables in the original program do not appear in DIM statements. BASIC09 gives them the type REAL, because their names don't end with "$". Declaring those variables that contain only integral values to have the type INTEGER cuts the program's run time to 15.3 seconds.

Unlike tokenized BASICs, or BASICs using suffixes to indicate the types of constants, BASIC09 considers numerical constants without scientific notation or decimal points to have type INTEGER. Calculating pixel coordinates entirely in REAL arithmetic without gratuitous conversion cuts the run time to 12.6 seconds.

Examining the pixel coordinate calculations shows we can simplify them to do more in integer arithmetic:

- instead of x3=320.+x1*640./4.8 we can use x3=320+FIX(x1*400.)/3, and

- instead of y3=116.-y1*192./3.6 we can use y3=116-FIX(y1*160.)/3

This cuts the run time to 10.8 seconds, almost doubling the speed with minimal effort.

Slightly Less Easy Changes:

If you've studied recursion and data structures and know some old programming languages, you will recognize the parallel arrays in the original program. They comprise a stack of structures, a common dodge to avoid recursion in languages lacking it or to eliminate recursion whether the language supports it or not.

Eliminating recursion can have advantages, even if you use a language supporting recursion, like BASIC09— but apart from some easy cases, eliminating recursion makes a program more obscure. (To give recursion elimination its due, not using BASIC09 TYPEs causes much of the obscurity here.)

I started these experiments by making the program recursive, to reassure myself that I understood it.

Doing so, and changing the pixel coordinate calculation as we described earlier, reduced the run time to 5.8 seconds. I expected it to slow down, but I think the I-code interpreter can implicitly push and pop values via procedure calls faster than it can do so explicitly via multiple statements and multiple array references (each with subscript range checking). I do not know whether using BASIC09 TYPEs to reduce the number of array references per push/pop would help.

More Interesting Changes:

To go further requires study of the Pythagoras tree itself. Starting with an initial point in the plane, (1, 0), the program generates a sequence of points and draws circles around them. Because each point in turn determines two other points, a binary tree arrangement seems natural. We will number the terms in a way you'll recognize if you have ever looked at heapsort:

```
      1
     / \
    2   3
   / \ / \
  4  5 6  7
```

We label the root of the tree term 1, and then in general, we label term n's children 2n and 2n+1.

The speedups we derive in the following depend on the formula for the points. In the original program, you will notice a variable C that the code sets to .5 and leaves unchanged. I can think of a couple of reasons for the variable:

1. Tokenized BASICs like Color BASIC can take longer to interpret numeric constants than to look up a variable's value, and programmers sometimes try to speed up their BASIC programs by assigning constants to variables and then referring to the variables.

2. The Pythagoras tree may have other interesting variants generated with different values of C, making C a "parameter" in the mathematical sense rather than the programming sense.

In the following, we will use the value .5 for C. In the original program, X and Y correspond to the root of the current subtree, X1 and Y1 to the left child, and X2 and Y2 to the right child. Using .5 for C, the tree-oriented numbering, and the distributive law, the smoke clears to reveal:

x(2n) = (x(n) - (y(n) + 1)) / 2
y(2n) = (x(n) + (y(n) + 1)) / 2

and

x(2n+1) = (x(n) + (y(n) + 1)) / 2 = y(2n)
y(2n+1) = (-x(n) + (y(n) + 1)) / 2 = -x(2n)

This simplification cuts the runtime to 5.5 seconds. (I used excess parentheses above to clarify the relationships involved; the source does not contain them.)

The next change will occur to you if you look at the program output. Notice its symmetry about a vertical line. That will let us draw two circles at a time, halving the calculation, *if* we can generate points in the right order, skipping the ones we can derive.

"But wait," you say, "the original program already draws two circles at a time!"

Yes, it does— and since you may find the technique useful in other tree- walking code, it merits study. The typical "inorder" binary tree traversal code reads something like:

```
"visit" the node
IF it has a left child THEN
  traverse the left subtree
ENDIF
IF it has a right child THEN
  traverse the right subtree
ENDIF
```

(People who write about tree traversal use "visit" to mean doing what you actually want to do with each node. In the case of the Pythagoras tree, when we "visit" a node, we draw a circle around the matching point.)

In the Pythagoras tree, the sequence goes on forever, so every node has both children and we decide where to quit (in this program, at the tenth level), changing the standard traversal to

```
"visit" the node
IF we haven't gone as deep as we want THEN
  traverse the left subtree
  traverse the right subtree
ENDIF
```

In addition, we don't draw a circle around the root, making it easy to "hoist" the visitation of the children and write:

```
"visit" the left child
"visit" the right child
IF we haven't gone as deep as we want THEN
  traverse the left subtree
  traverse the right subtree
ENDIF
```

The original programmer did so, and thereby saved one level of recursion for a given tree depth, or rather, since the original program didn't use recursion, the explicit stack used one less entry. (I had to do a small example by hand to convince myself. If you don't see it at first either, emulate it by hand with a small maximum depth.) So, remember this method. It may come in handy for your next inorder tree traversal.

Returning to the quest for symmetry, if we take advantage of symmetry *and* use the above traversal method, then we will draw *four* circles per invocation: the two we drew before and two corresponding to them.

But *can* we take advantage of symmetry? The output gives us a big hint— we notice matching points because their circles have the same radius. What determines the radius of the circles? Depth in the tree. Symmetric points in the tree must thus have equal depth. The first few levels reveal the pattern:

```
                  — (0,1) —
              /               \
        (-1,1)                  (1,1)
       /      \               /       \
(-1.5,.5)  (.5,1.5)   (-.5,1.5)  (1.5,.5)
```

(Fortune smiles upon us. Not only does symmetry hold, but symmetry about the y axis— the simplest kind, as Mrs. Olsen would say.)

Math majors who haven't already figured it out will now say

"Of course! For all n >= 0, for all k in {0, ..., 2^n - 1}, the (2^n + k)th point corresponds to the (2^n + (2^n - k - 1))th point!"

For non-math majors, a bit more slowly: 0 = -0, so the root corresponds to itself. It and the second level can only work one way, and the third gives away the pattern: corresponding points run from each end towards the middle. You can easily prove the more formal statement of the property by mathematical induction on n, which corresponds to tree depth.

The relationship between symmetric points lets us just traverse either the left or the right subtree of the root, drawing the other subtree by taking advantage of symmetry. Combining this with the hoisted visitation hack takes some finesse. We have to skip one child, but only at the top level. Also, unless we check for the top level, we draw the two largest circles twice— but nobody will notice two circles of 1,022 drawn twice. We didn't bother.

Taking advantage of symmetry cuts the program's run time to 4.0 seconds.

One More Change:

If we can calculate the coordinates in INTEGER arithmetic with sufficient accuracy, we can speed up the program yet again.

Because the calculations mix X and Y coordinates, we must use the same scaling factor for both of them. We must also keep to the 6809 BASIC09 INTEGER value limits, -32768 to 32767, to make sure the program will run on a 68xxx computer or the CoCo.

We multiplied the X coordinate by 400 and then divided by 3, and multiplied the Y coordinate by 160 and then divided by 3. That suggests the least common multiple of 400 and 160, 800, as our scaling factor, which allows a coordinate as large as 32767 / 800 = 40.96 without overflowing. The recurrence relation adds two coordinates, constraining their absolute values to about 20 for safety— but 20 easily exceeds the maximum coordinate in the tree.

Then the conversion to pixels changes as follows:

```
x3=320+FIX(x1*400.)/3  ->
     x3=320+x1/6  (because 400 / 3 = 800 / 6)
y3=116-FIX(y1*160.)/3  ->
     y3=116-y1/15  (because 160 / 3 = 800 / 15)
```

and 1 in the calculation of the successive terms, of course, becomes 800.

To move REAL calculations entirely out of the main part of the program, we can also precalculate the scaled radii and save them in an INTEGER array analogous to the array of palette numbers. (It makes the relation between radius and tree depth more obvious, too.)

These changes don't visibly affect the output, and cut the run time to 3.0 seconds.

Summary:

I have run out of ideas for speeding up the Pythagoras tree program in BASIC09, though with some compiled languages I might try recursion elimination.

On the other hand, we may have criteria other than speed. The integer version depends very much on our choice of resolution and region to display, and any extensions will probably force us back to floating-point. Symmetry doesn't combine neatly with the hoisted visitation trick, so you could decide to use the standard tree traversal and take advantage of symmetry by just traversing the left subtree of the root. (It still beats the non-symmetric program by almost a second.) The clearer code may run fast enough. A mathematician may just want to look at the output to gain intuition, as we did by noticing the tree's symmetry, and may not even care if the program takes all night to generate results.

On the third hand, as Lt. Arex on the animated Star Trek would say, most of the speed came from simple, obvious changes. Keep them in mind when you write your next BASIC09 program or convert a program from an unstructured BASIC to BASIC09. Why pass up easy speedups?

On the fourth hand, we haven't covered all the improvements you can make when converting a Color BASIC program to BASIC09. Cleaning up control structures will have to wait for another article.

- James Jones

(see next page for program listings)

Unmodified PYTH listing:

```
PROCEDURE pyth
DIM i,palette(10):INTEGER
FOR i:=1 TO 10
palette(i):=1+RND(14)
NEXT i
RUN bgfx("clear")
RUN bgfx("curoff")
RUN pythrec(0.,1.,.5,palette,1)
RUN bgfx("curon")
PROCEDURE pythrec
PARAM x,y,r:REAL
PARAM palette(10),depth:INTEGER
DIM x1,y1,r1:REAL
DIM rs:INTEGER

RUN bgfx("color",palette(depth))
x1:=.5*(x-y-1.)
y1:=.5*(x+y+1.)
rs:=130.*r
RUN bgfx("circle",320+FIX(x1*40
   0.)/3,116-FIX(y1*160.)/3,rs)
RUN bgfx("circle",320+FIX(y1*40
   0.)/3,116-FIX(-(x1)*160.)/3,
   rs)
IF depth<9 THEN
r1:=r*.707107
RUN
pythrec(x1,y1,r1,palette,depth+1)
RUN pythrec(y1,-(x1),r1,palette
   ,depth+1)
ENDIF
```

New PYTH listing:

```
PROCEDURE pyth
DIM i,palette(10):INTEGER
FOR i:=1 TO 10
palette(i):=1+RND(14)
NEXT i
RUN bgfx("clear")
RUN bgfx("curoff")
RUN pythrec(0.,1.,.5,palette,1)
RUN bgfx("curon")
PROCEDURE pythrec
PARAM x,y,r:REAL
PARAM palette(10),depth:INTEGER
DIM x1,y1,r1:REAL
DIM xs1,xs2,ys1,ys2,rs:INTEGER

RUN bgfx("color",palette(depth))
x1:=.5*(x-y-1.)
y1:=.5*(x+y+1.)
xs1:=FIX(x1*400.)/3
xs2:=FIX(x1*160.)/3
ys1:=FIX(y1*400.)/3
ys2:=FIX(y1*160.)/3
rs:=130.*r
IF depth=1 THEN
RUN bgfx("circle",320+xs1,116-ys
   2,rs)
RUN bgfx("circle",320+ys1,116+xs
   2,rs)
r1:=r*.707107
RUN pythrec(x1,y1,r1,palette,dep
   th+1)
ELSE
RUN bgfx("circle",320+xs1,116-ys
   2,rs)
RUN bgfx("circle",320+ys1,116+xs
   2,rs)
RUN bgfx("circle",320-xs1,116-ys
   2,rs)
RUN bgfx("circle",320-ys1,116+xs
   2,rs)
IF depth<9 THEN
r1:=r*.707107
RUN pythrec(x1,y1,r1,palette,dep
   th+1)
RUN pythrec(y1,-(x1),r1,palette,
   depth+1)
ENDIF
ENDIF
```

Quality OS-9 Software from

ColorSystems

NEW! K-Windows Chess for MM/1 ....$24.95

NEW! X10 Master Control Program for MM/1....................$29.95

Variations of Solitaire $29.95 (MM/1)
Pyramid, Klondike, Spider, Poker & Canfield
$19.95 (CoCo3)

OS-9 Game Pack $29.95 (MM/1)
Othello, Yahtzee, KnightsBrudge, Minefield and Battleship
$19.95 (CoCo3)

WPShel $14.95 (CoCo3)
An OS-9 Word Processing Point and Click User Interface

Using AWK with OS-9 ....$14.95 (MM/1)
Includes V2.1.14 of GNU AWK for OS9/68000

To Order Send Check or Money Order to
ColorSystems
PO Box 540
Castle Hayne, NC 28429
(910) 675 1706

Call or write for a free catalog!
Demo Disks also available!

Owned and Operated by Zack C Sessions

NC Residents please add 6% Sales Tax

Come see us at Chicago!
Mention this advertisement and
get a special discount!

FARNA Systems

New from FARNA Systems: **The CoCo Family Recorder/OS-9!** Record your family's history on your CoCo! Very easy to use genealogy program features all necessary modules to make your research easy. The OS-9 version requires OS-9 Level 2, an 80 column monitor, and at least one double sided 40 track drive (larger is fine). Can be shipped on a 3.5" 720K drive if requested (no extra cost). $28.50

If you own the DECB version, you can upgrade for $20.00 by sending your ORIGINAL DECB disk (it will be returned). The OS-9 version has a data file conversion utility (RS-DOS command and CC3Disk patch required... see "Patch OS-9" below). The DECB version is still available for $18.50

Patch OS-9: A complete set of the most used and desired patches and utilities for OS-9 Level 2! We have sold nearly 150 sets of these disks! Many OS-9 users have stated that this is the closest thing to a Level 2 Upgrade yet! An auto-install utility is included (requires two drives, 40T/DS or larger, and 51K) $7.50

Want to get those bulky OS-9 Level 2 or OSK manuals off your desk? Order one of our Quick Reference and Progammers Guides! The Level 2 version is $7.50, OSK version is $8.50. Order the Level 2 QRG and Patch OS-9 together for only $12.50.

We also sell Ken-Ton SCSI hard drive interfaces and complete systems. Please inquire for a catalog.

PLEASE INCLUDE $2.50 S&H PER ORDER

FARNA Systems, Box 321, Warner Robins, GA 31099-0321

"So this is a slide-rule. How did they program it?"

Iowa and Country CoCo Club, features articles and graphics from various authors
Cost: Free with membership ($16/yr)
Distributor: **Mid Iowa and Country CoCo, Terry Simons,** 1328 48th, Des Moines IA 50311

Name: **CoCo~1 2 3**
Description: Newsletter of the Glenside CoCo Club
Cost: Club Membership, $15
Distributor: **Glenside Coco Club of Illinois, Tony Podraza,** 119 Adobe Cir., Carpentersville IL 60110, *(708) 428-3576*

Name: **CoCo Friends Disk Magazine**
System Req.: CoCo 3, 128K, RGB monitor
Description: Bi-monthly magazine on disk dedicated to RS-DOS, 8-15 new programs per issue, articles, and graphics
Cost: $30/yr
Distributor: **Rick's Computer Enterprise,** P.O. Box 276, Liberty KY 42539, *(606)-787-5783*

This issue closes with another article in the Chris Dekker Basic09 series, along with a followup to one of the previous Dekker articles from James Jones of Microware. He presents an interesting look into optimizing the Pythagoras Tree algorithm which Chris introduced previously.

JWT Enterprises plans to keep back issues of *UpTime* for sale (as our supply permits), as well as back issues of our *Nine-Times* publication and our Optimize Utility Set Pack. I hope you have enjoyed *UpTime* in its relatively brief existance, and I wish everyone the best! Thank you for supporting *UpTime*!

Jordan Tsvetkoff

Advertiser's Index

UpTime
JWT Enterprises
5755 Lockwood Blvd.
Youngstown, OH 44512

Address correction requested

JWT Enterprises
5755 Lockwood Blvd. Youngstown, OH 44512 Tel. (216) 758-7694

Dear UpTime subscriber,

Thank you for your continuing support of *UpTime*. As explained in the newsletter, this is the final issue of publication for UpTime. All of your remaining subscription credit, if any, will be transferred to **FARNA System's** *68 Micros*. For every two issues of *UpTime* credit, you will receive a one-issue credit for *68 Micros*. Listed below is the total issues that will be credited to your *68 Micros* account. If you feel that this information is incorrect, please get back with us as soon as possible so that any errors may be corrected. Thank you again for your support over the past two years!

You have ______ credits of *68 Micros* issues.

For more information about our products, JWT Enterprises advertisement in on the reverse