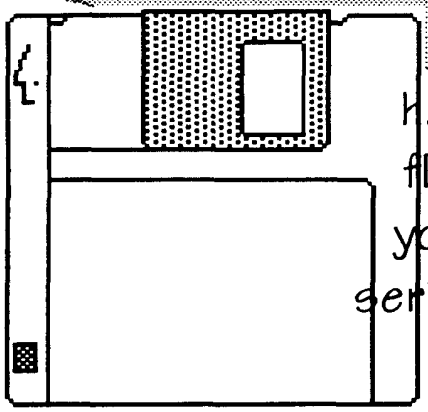
March/April 1996 Vol 3 Number 4 $4.50 Canada, $4.00 US

the world of 68' micros

Supporting Tandy Color Computer Disk BASIC, CoCo OS-9, and OS-9/68000



Now.. you can have hi-density floppydrives on your CoCo... for serious hardware hackers only!!

James Jones shows us how to write a check in Basic09!

CONTENTS

GLENSIDE COLOR COMPUTER CLUB PRESENTS:

THE FIFTH ANNUAL "LAST" CHICAGO COCOFEST!

Held at the Elgin Holiday Inn

(for more info see ad on page xx. HURRY! Dates are 13-14 March!)

POSTMASTER:

If undeliverable return to:
FARNA *Systems* PB
Box 321
Warner Robins, GA 31099

Address Correction Requested

the world of 68' micros

Publisher:
FARNA Systems
P.O. Box 321
Warner Robins, GA 31099-0321

Editor:
Francis (Frank) G. Swygert

Subscriptions:
US./Mexico: $24 per year
Canada: $30 per year
Overseas: $50 Per year (air mail)
Back and single issues are $4.50 U.S. And Mexico, $5.50 Canada, $9.00 overseas air. Bulk/newsstand orders are available.

microdisk:
Companion to magazine with program listings and share/fre ware mentioned in some articles. Published three times per year.
U.S./Mexico: $20 per year ($8 single)
Canada: $25 per year ($10 single)
Overseas: $40 per year ($15 single)

Advertising Rates:
Contact publisher. We have scales to suit every type of business. Special rates for entrepreneurs and new businesses.

Contributions:
Any and all contributions are welcome. Submission constitutes warranty on part of the author that work is original unless otherwise stated. Publisher reserves the right to edit or reject material without explanation. Editing will, however, be kept to a minimum.

General Information:
Current publication frequency is bi-monthly. Frequency and prices subject to change without notice. All opinion expressed herein are those of the individual authors, not necessarily of the publisher or editor. No warranty as to the suitability or operation of any hard or soft ware is given nor implied under any circumstances. Use is entirely up to the discretion and responsibility of the reader

All trademarks/names property of their respective owners.

The publisher is available for comment via e-mail at dsrtfox@delphi.com. The Delphi CoCo and OS-9 SIGs on Delphi and related Internet news groups are also frequented

A message from the editor...

Where do I start this time? I believe the best is with an apology. I still haven't sent out, or even put together, the last issue of "microdisk". I'll be getting it together soon, maybe even have it done before you read this.

This is due to several reasons: 1) I just moved again because I recently got married, and I seem to have misplaced some disks. 2) I had not one but THREE hard drive crashes on the PC (which ultimately led to some lost files, including the entire last issue an 1/3 of this one I had already put together). This led to a new hard drive and rebuilding the entire PC system. Luckily, I had copies of all the files I needed for this issue, just hadn't copied the last issue to floppy yet. And finally, 3) I have to find time to do all this between the new wife, work, moving, and rebuilding the PC. Luckily, my wife understands that I have an obligation to you and that I have to sit down and fulfill that obligation from time to time.

Those who've subscribed a while will realize that I just recently (last May) got divorced. All I can say is that it was something that had been waiting to happen for years, and finally did.

I'm sure many of you thinkg I must be crazy, getting married less than a year later (December of this year), but I didn't mean for it to happen, really! And no, I didn't know my wonderful wife until AFTER I was separated. In fact, I separated from my "ex" a couple weeks before last year's Chicago 'fest, and met my bride two weeks (exactly!) after. Neither of us intended to get serious, but guess what happened? We just hit it off so well that anything less just didn't make any sense at all.

For those who've congratulated me on my new posting, there wasn't one. It seems that there would be no openings in the area I wanted to be in for two more years, so I had to decline the recruiting position. To much work to go just anywhere!

Speaking of the Chicago 'fest, I hopr to see many of you there. Stop by the FARNA Systems booth and say hello! I may not have much time to talk, but I do like to at least meet the people on the receiving end of this thing.

The fests have mainly become an annual get together for the real CoCo/OS-9 "diehards". I generally see the same people every year, with a few new faces. And I miss seeing a few also. But I'll be there again this year, and Tiffany (my new bride!) will be there with me. She attended the Atlanta fest with me last year, and I can't wait to take her downtown Chicago to the Hard Rock Cafe or Planet Hollywood... but without last year's excitement! The Rambler has pretty much recovered from the wreck, but is still in bad need of a paint job and some other cosmetic and just a little mechanical work. Don't know if I'll drive it to Chicago this year... might be in Tiff's Jeep Comanche instead. Decisions, decisions... you just decide to be there yourself if you possibly can! I rushed to get this issue out to you before the fest...

How do you like the "new look"? I wanted to make a noticeable change, but still maintain an easy to work with layout and all the things readers have said they liked. The folding in half for mailing idea didn't work... to much trouble... so most received their's flat. Only about 20 or so of you got the folded issues I mentioned in the editorial. This is the way you'll get them from now on.

As always, if you think of something that will make the magazine look better, or be more practical, let me know! And keep writing! I always need articles and "shorts" to fill in here and there. And though you may not think you're much of a writer, you still may have an idea that will help others. So don't be afraid to put in your "two cents worth". It may prove to be invaluable to someone else...

Letters from our readers...

I am trying to track down a used MM/1 to buy, so I am sending this letter to you (and a few others who may have some information). Even if you don't have an MM/1 to sell, maybe you know of someone who does. This is a shot in the dark, but I thought I would try. I am interested in having an OSK machine and the MM/1 seems to be the most reasonable way to get started, since I already have a CM-8 monitor I could use with it. Any feedback about the MM/1 or other OSK machines would be appreciated.

Ted Willi
Box 447
Athens, GA 30603

Nice to hear from you Ted! And your thoughts on the MM/1 -- that it would be inexpensive for a CoCo user to get into -- was both what compelled the designers to make it the way it is AND what crippled it. The CM-8 is a liability on another computer, not an asset. While you are correct, you can get an MM/1 at a reasonable price, I would NOT recommend it unless you get an extremely good deal.

You see, the very factors that the designers considered literally crippled the machine. It is very limited in the graphics department. Had they put more serious thought into it and had a higher resolution mode, it would have been more practical. Instead, you are limited to CoCo Level II type windows and graphics not much better than Multi-Vue. The idea was to allow CoCo users to transfer as much of their knowledge and equipment to the new machine as possible. Unfortunately, there wasn't enough thought put into future expansion, and now the machine is almost as obsolete as the CoCo 3.

The most efficient thing to consider (moneywise) is the new MM/1b board from BlackHawk or Wittman Computer Products. For $400 you get the board and enough software to get started. You can still use your CM-8 by connecting your CoCo 3 as a terminal. You will still have to buy 30 pin SIMMs, a case and power supply, and a 1.4M drive, but the overall value and future expansion of the machine will make the extra cost worthwhile.

Browsing through a current Computer Shopper, I came up with the following prices:

Memory, $28/megabyte
Case and Power Supply, $40
1.4M floppy drive, $30
130MB IDE hard drive, $80

So the complete system would cost just under $600 with 2MB of RAM, which is plenty to start with. The good thing is you can easily add more memory (up to 16MB... the MM/1 is limited to a max of 9MB, I believe), the hard drives are cheap (and you can add another easily), a keyboard and video card can be added easily, and 1024x768 color graphics are attainable. A full blown system can actually be purchased for $800-$900, including a color VGA monitor. You could save a lot by using a standard VGA (640x480) Color monitor or a black and white hi-resolution monitor. Were it not for the fact that the drivers were real particular about the VGA card, one could buy a used 286 or 386 system with an IDE hard drive and throw away the motherboard. If the deal is right (and there ARE some out there), it may be worth persuing that idea anyway. I'll look into that later, maybe...

Oh yes, don't get the idea that I'm telling you this just to support my advertisers. The truth is, I really do think you'd be better served by trying to build a system than get an MM/1. It would be more cost effective in the long run. So you might just want to keep your CoCo for a while longer, at least until that tax check comes in!

Thanks for the info on replacement hard drives. I have been thingking I have a cold solder joint on mine in the motor circuit. Would I be better off to replace it or buy a rebuilt one? I know my unit has few hours on it, perhaps it would have a longer life than a rebuilt.

I noticed you sent your last issue first class. If you are short a few subscriptions, you can send multiple copies to the same address. A little figuring on your part would determine if this is feasible. I'm a long time Postal Service employee and checked this with our bulk mail clerk.

Thanks for the tip on the mailing! I did check this, but it wouldn't save me very much. And I have a responsibility to you, the reader. I could save about $20 each mailing by doing just that, but that's not enough to keep you waiting the extra week. So it isn't worth your money for me to do that.

And is probably isn't worth your money to have that drive repaired. Repair prices for drives are outrageous when compared to what the big wholesalers want for remanufactured or reclaimed (pulled from unsold but still new machines, or from known good machines that a large corp. sold when new computers came in) units. If you want to try your hand go ahead though! If you'll have to replace it anyway, you won't be hurting anything!

CoCo's For Sale...

I am enclosing an advertisement for some CoCo equipment that I no longer need since I have moved to a System IV. Please run in your next available issue.

CoCo3, 6809/512K,
CoCo3, 6309/2MB
B&B System w/ 2 10MB HDs
Disto FD Controller
1 5.25" 360K drive
1 5.25" 720K drive
1 3.5" 720K drive
Upgraded Multi-Pak
Telepak RS-232 pak
Tandy LPVII
Magnovox RGB monitor
Mouse and trackball
Disto 512K RAMDisk
w/clock & parallel printer port

Much software, including OS-9 Levels I and II, Multi-Vue, and many Rainbow on tape cassettes. Some other misc. hardware.

The first $400 takes everything. I would rather sell all at once, but may break it up.

Ed Haas
3448 Marcella Ave.
Stow, OH 44224
(216) 688-4265

WANTED

Copy of the manual and disk for the "VDX Grafx" video digitizer. This was made for the CoCo circa 1984 and used the I/O port for connection. If you have one, please let me know -- I'd like to try using the one I have!

Edmund C. Bassick
75 Romanock Place
Fairfield, CT 06432
Day (work) 800-283-5411
Evening (home) 203-254-0571

Checks 09!

James Jones

We not only get a conversion, but a lesson in programming as well!

Editor: *This article was written in response to the call for a Basic09 programmer to convert the DECB Checks3 program, presented in teh last issue, to Basic09. Mr. Jones went well above and beyond that, presenting instead a very good lesson in programming in general, and pointing out major differences between DECB and Basic09. You may want to keep this one fro reference later.*

I fear that you and Mr. Heath have set people a difficult task. Color BASIC is one of the family of BASIC interpreters that Kemeny and Kurtz, the inventors of BASIC, coined the phrase "gutter BASIC" to refer to. It is nearly bereft of legible control structures. It has only one data structure, the array. There's no way to limit the scope of variables. By the time one has written a Color BASIC program of any significance, whatever structure may have been in one's head is barely visible in the source code. Moreover, the inner workings of the interpreter encourage the programmer to do his best to positively obliterate any clues to the function of the code. Comments? They consume valuable space, leave them out! Line numbers use up space, too, so cram as many statements onto each line as possible!

Trying to convert the result to BASIC09 calls as much for cryptanalytic as for programming skills. One has to reconstruct the structure in the original programmer's head, and with that, one might as well start over. A line-by-line conversion will result in the same unmaintainable mess one started out with, confirming the famous claim that one can write an unreadable program in any language. (Please note: I would not denigrate the skills, much less raw courage, of anyone who actually *does* such a conversion.)

NOTE: a useful tool for anyone trying this sort of conversion on any significant scale would be a filter to delete all unused line numbers. Line numbers are your enemy. Wherever a line number occurs, you can no longer tell what's going on when you reach the numbered statement without reading "the whole program". Alas, BASIC09 lacks structured equivalents of ON...GOTO/GOSUB and ON ERROR GOTO, so you can't avoid line numbers completely; all you can do is try to use them in a disciplined fashion. (The same goes for the RESTORE statement, but I digress.)

In the case at hand, the important structure is the check file, and to determine its layout, the place to look is at the FIELD statement. Aha! From evidence elsewhere in the program, we infer the purposes of the fields:

da$: 9 characters (check date)
ch$: 5 characters (check number)
amt$: 5 characters (raw bits of floating point amount) (Note that as converters, we can't let that third character lull us into a false sense of security...)
pa$: 20 characters (payee)
bn$: 3 characters
(bank? And if so, which one?)
ec$: 2 characters (expense code)
cl$: 1 character (has the check cleared?)

NOTE: it turns out that the program in the magazine is somewhat stripped down from the original on Delphi (editor: this is my fault... I printed the copy I modified for my own use!). One of the changes is that one could attribute the checks to either of two banks, and changing the bank was a choice in the main menu. Since I've never seen an arrangement in which a single account can have checks drawn on it from multiple banks, and because it was stripped from the program in the magazine, I will remove the "bank" field from the records. (editor: The original program would track two individual bank accounts. I changed this to keep from getting my two separate accounts confused)

Corresponding to that, and with the "bank" field removed, we would write the following BASIC09 TYPE statement:

```
TYPE check = date:STRING[9];
number:INTEGER; amount:REAL;
payee:STRING[20]; exp_code:
STRING[2]; cleared:BOOLEAN
```

INTEGER on the 6809 is a signed two-byte quantity, but you'd have to write a check a day for not quite ninety years to write 32767 checks, so I won't worry about check number overflow even on the 6809. Note how the fields have types that reflect their purpose, rather than just being undifferentiated strings. In Disk BASIC, if one were to mistakenly type something like:

```
amt$ = "$%#"!"
```

the BASIC interpreter wouldn't lift a finger in protest, until the vict,er, user, trips over it at run time when the CVN function blows up.

From looking at the source, it doesn't appear that the data file is anything more than a simple sequential file. To avoid the need to keep multiple copies of the program, I'd eventually want to put the bank name and account number in the file rather than hard-wiring them into the program; Disk BASIC makes this hard to do, so I can understand why it's that way in the original program. In BASIC09, it's so simple that I'll leave it as an exercise to the reader (It would affect various SEEK statements in the following code).

Now, for the control flow. Fortunately, from the beginning, it flows straight into what looks like a top-level menu:

(1) checkbook entries
(2) display entries
(3) review expenses
(4) balance checkbook
(5) correct/delete entry
(6) quit

followed by an ON...GOTO that presumably branches to code to perform the selected function.

Speaking of which, note that the first option is the only one not of the form <verb> <direct object>, and it's also the only option that doesn't give one an idea of what it means, what *about* checkbook entries? Looking at the code shows that the associated function is *adding* checkbook entries, so I'd propose changing the displayed message to say "add entries."

Once we go to the code to do the function, do we ever get back? Do we have to run the program again to do something else? With ON...GOTO rather than ON...GOSUB, there's no way to tell without following through the maze. We'll pray that we don't overlook some obscure path through the code and try just looking at the final pieces of each section...OK. Some loop back to the start of the section, but all eventually go to 9010, which does get back to near the beginning, assuming that "EXEC 44539", whatever "that" does, doesn't affect the control flow (editor: it tells DECB to wait for any key press). Then we have top-level logic like do any needed setup, open files, etc.

```
LOOP
    display menu of actions
     select action
  EXITIF user chose to quit THEN
  ENDEXIT
      do action
  ENDLOOP
  do any needed cleanup, close
   files, etc.
   END
```

NOTE: the above is "pseudocode"; it glosses over language details but indicates the general method and flow of control. Because BASIC09 provides reasonable control flow constructs, I can write "BASIC09-like" pseudocode and still preserve the purpose of pseudocode, making it that much quicker to get to running code.

If we want to commit to representing the choice as it's done in the Color BASIC program, i.e. as a number corresponding to the desired action, we'd make the code more specific, like do any needed setup, open files, etc.

```
LOOP
    display menu of actions
     select action
```

```
    EXITIF action = number for "quit"
  THEN ENDEXIT
       ON action GOSUB
        1000,2000,...
   ENDLOOP
   do any needed cleanup, close
    files, etc.
    END
```

(If the number for quit is at the end, as it is in the original, then we can take advantage of that and the behavior of ON...GOSUB to write the loop as a REPEAT...UNTIL.)

The GOSUB, like Arnold Schwarzenegger in "The Terminator", promises, "I'll be back", so that for understanding the control flow at a high level, you "don't" have to trace through a maze of twisty little passages, all different.

Actually, in this particular program, they're not "all" different. Look at those hunks of code starting at 2080 and at 3070! Very similar. "display entries" displays all entries or those for a particular month; "review expenses" displays all entries with a particular expense code, or those for a particular month. In either case, one is selectively displaying entries, so why make it two different functions? I propose a single "display entries" action, with the user entering values for the date, payee, and expense code to match, where we say that a string S1 "matches" the string S2 if:

SUBSTR(S1,S2) = 1

i.e. S2 begins with S1. Since every string "begins with" the empty string, anything left empty doesn't affect the match.

Not only does this cut out a bunch of replicated work, it also makes the program more flexible and powerful. It lets us select by any conjunction of {date, payee, expense code}; if we want to see all the checks paid to Joe Blow's Pizza in March for rent, we can.

NOTE: one difference between the two functions in the original code is that "display entries" prints the balance, while "review expenses" only prints a total of the selected checks. What options should we give the user in our unified function? We're not sure yet; stay tuned.

Of course, things other than checks affect the balance: deposits, interest (if your bank or credit union pays it on checking or share draft accounts), ATM withdrawals, and charges. These really shouldn't have check records associated with them, or rather, we should say the file isn't a sequence of checks, but a sequence of items that can be of any of several kinds (editor: I get around this by putting "ATM" in the check number field and entering the withdrawal as a check in date sequential order). In some other languages one can represent this as a discriminated union (aka union with variant tag); in more recent "object oriented" languages, these would be subclasses of the larger "checking transaction" class, which share the common attributes of "date" and "amount".

BASIC09 is lacking in this regard compared with more modern languages. We could give TYPE statements for the various kinds of transactions and isolate the tag, but for this simple case, we're better off doing what the original program did: sticking with a single record type and overloading the "expense code" to function as a variant tag.

Back to the specific menu items: the first one, adding entries to the file, turns out once the smoke clears as

```
read file and accumulate balance
  display last entry and print balance
   old_balance := balance
    LOOP
        select action
   EXITIF user chose to return to
   main menu THEN ENDEXIT
        IF user chose deposit THEN
            get deposit entry
         ELSE
            get check entry
         ENDIF
        display new entry, balance
        IF user approves THEN
            write new entry to file
            old_balance := balance
         ENDIF
    ENDLOOP
```

The "read file and accumulate balance" would expand into something like this, presuming that we have #cpath open to the file and a variable named "record" of type check. (We will want the file open for update, since we will be writing as well as reading it.)

```
      balance := 0
      SEEK #cpath,0
    WHILE NOT EOF(#cpath) DO
        GET #cpath, record
        balance := balance +
        record.amount
    ENDWHILE
```

Well...at least I "think" that's the intent of lines 1030-1080 in the original code. (I have a Disk BASIC manual somewhere, but I can't find it.)

NOTE: Here's yet another thing we'll leave as an exercise for the reader. We know what all the actions in the program do to the balance, so there's no need to read the whole file solely to determine the balance more than once. Indeed, we could just write the current balance at the beginning of the file (along with the bank name), and then we'd "never" have to read the whole file just to determine the balance.

By the way, there's a slight problem with the logic here. The very first time the user runs the program, there are no entries in the file, so the "display last entry" will print garbage. We should either initialize record with some dummy values or check whether we actually read anything before display.

On to the balancing act. Alas, a mistake caused some crucial lines from this portion of the code to escape printing, as evidenced by the lack of any code to change the variables B3 and B4 from the zero assigned to them at the start of this section of the code. From the original, we reconstructed this pseudocode for balancing:

```
    for each uncleared entry do
        display it
        IF the user says it cleared
        THEN
            mark it in the file as
             cleared
            IF it's a deposit THEN
                add the amount to
                 a total of newly
                 cleared credits
             ELSE
                add the amount to
                 a total of newly
                 cleared debits
             ENDIF
        ENDIF
    endfor

    IF user says there was interest
    paid THEN
        add entry for interest
        add the amount to a total of
        newly cleared credits
    ENDIF
    IF user says there were bank
    charges THEN
        add entry for bank charges
        add the amount to a total of
         newly cleared debits
    ENDIF

    for each entry do
        IF it's cleared THEN
            add the amount to a total
             of cleared entries
        ELSE
            add the amount to a total
             of non-cleared entries
        ENDIF
    endfor
    display total of newly cleared credits
    display total of newly cleared debits
    display total of cleared entries
   (should match bank statement balance)
    display total of cleared entries +
     total of non-cleared entries
     (should match checkbook balance)
```

The documentation file that comes with the program say that as your file gets larger you will notice the listing and balancing functions take more time. From the above pseudocode one of the reasons is clear; the balancing code makes two complete passes over the file, one to look for entries that haven't yet cleared, and one to add up overall totals. Let's cut the run time in half and rewrite this as:

```
for each entry do
    IF it's cleared THEN
        add the amount to a total
        of cleared entries
    ELSE
        display it
        IF the user says it hasn't
        cleared yet THEN
            add the amount to
            a total of non-
            cleared entries

        ELSE
        mark it as cleared in the file
        add the amount to a total of
        cleared entries
            IF it's a deposit THEN
            add the amount to a
            total of newly cleared
            credits
            ELSE
            add the amount to a
            total of newly
            cleared debits
            ENDIF
        ENDIF
    ENDIF
endfor
IF the user says there was interest
paid THEN
    add an entry for interest
    add the amount to a total of
    cleared entries
    add the amount to a total of
    newly-cleared credits
ENDIF

IF the user says there were bank
charges THEN
    add an entry for bank charges
    add the amount to a total of
    cleared entries
    add the amount to a total of
    newly-cleared debits
ENDIF
display total of newly cleared credits
display total of newly cleared debits
display total of cleared entries
(should match bank statement balance)
display total of cleared entries +
total of non-cleared entries
(should match checkbook balance)
```

That should help things out.

NOTE: Here is yet another chance to improve performance that I will leave to the reader as an exercise. If we write the balance at the beginning of the file, then we don't need what was in the original code the second pass over the file. If we also keep a low-water mark indicating a spot below which we're certain that all entries are cleared, we can start there and need only read the current month's entries rather than the whole file, keeping the balancing function relatively quick.

Here's the above loop moved a little closer to BASIC09. Overwriting an entry is different from the sequential reads we've done hitherto, it means some explicit seeks and remembering our position in the file. On the 6809, the record position needs to be REAL to avoid overflow, but on the 68000, INTEGER will suffice:

```
tot_clear := 0.0 \ tot_unclear := 0.0
tot_newcredit := 0.0 \
tot_newdebit := 0.0
SEEK #cpath, 0
rec_pos := 0.0
WHILE NOT EOF(#cpath) DO
    GET #cpath, record
    IF record.cleared THEN
        tot_clear := tot_clear +
        record.amount
    ELSE
        display it
        IF the user says it hasn't
        cleared yet THEN
            tot_unclear :=
            tot_unclear +
            record.amount
        ELSE
            record.cleared :=
            TRUE
            SEEK #cpath,
            rec_pos
            PUT #cpath,
            record
            tot_clear :=
            tot_clear +
            record.amount
        IF record.exp_code
        = deposit_code THEN
            tot_newcredit
            := tot_newcredit
            + record.amount
        ELSE
            tot_newdebit
            = tot_newdebit
            + record.amount
        ENDIF
    ENDIF
ENDIF
rec_pos := rec_pos +
SIZE(record)
ENDWHILE
```

By the way, the original balancing code had a line reading:

```
IF CL$="C" THEN 4430 ELSE IF CL$="N"
THEN 4440 ELSE 4450
```

Huh? An entry is either cleared or not, but since the "correct/delete" code as written doesn't validate V$, which gets assigned to CL$, a careless mistake could cause an entry to slip through the cracks in the output for the "balance checkbook" function.

Speaking of deletion, you'll note that the original program copies the file (minus the entry to be deleted), deletes the original file, and renames the new file to give it the old name. (That's why all the menu items reopen the file.) Since chances are an entry to be deleted will be near the end, and to avoid having to generate random names for temporary files (this is OS-9; two processes might be running this code on different files in the same directory), it makes more sense to update the file in place. Presuming that rec_no has the record number of the entry to be deleted, something like:

```
OPEN #spath, file: READ
SEEK #spath, (rec_no + 1) *
SIZE(record)
SEEK #cpath, (rec_no) *
SIZE(record)
WHILE NOT EOF(#spath) DO
    GET #spath, record
    PUT #cpath, record
    rec_no := rec_no + 1
ENDWHILE
CLOSE #spath
set length of file to rec_no *
SIZE(record)
```

That should do the job. (Remember that cpath has the file open for update.) OS-9's ability to have two paths open to the same file makes it easy; since we're reading ahead of where we write, record locking will never trip us up. It will take a system call to set the file length.

NOTE: Once again, one could choose a different approach here. We could just "mark" "deleted" entries rather than actually deleting them, or perhaps mark them and then make one final deletion pass when the user chooses to leave the program.

The documentation for the original program says that deletion is rare, and we'd have to add checks for the mark everywhere else we scan the file—we'll leave this alone for now.

Speaking of paths open to the file, and of disciplined use of ON ERROR GOTO, we should deal with opening the file. BASIC09's OPEN, like the underlying system call, opens an existing file; it won't create one if it's not already there. We will write something like the following:

```
ONERRORGOTO100
OPEN#cpath, file: UPDATE
IFFALSETHEN
    100     errno := ERR
            IF errno <> 216 THEN
            PRINT "can't open
            "; file; ": error "; errno
            END
            ENDIF
            CREATE #cpath, file:
            UPDATE
        ENDIF
        ON ERROR
```

I must admit that the above code cheats a little—the error conditions for the CREATE are sufficiently similar to those for OPEN (with the exception of 216, of course!) that the same ON ERROR GOTO will work for both. Nevertheless,

the code is inspired by languages like CLU that let you associate error actions with specific statements. Rather than having a catchall ON ERROR GOTO, we tie it to the specific statement we anticipate errors from as clearly as we can; note the explicit deactivation with ON ERROR after the error handler.

By the way, using a variable for the file name is intentional, not pseudocode. We will make the file name a parameter of the main procedure, so that it needn't be changed to work on a different file. The shell will pass it along from the command line, so we needn't prompt the user for it as we would have to in Color BASIC.

Back to that "display entries" action: let's say that we'll print a balance if we're selecting at most by month; otherwise, we will only display a total of the selected entries. Just to anticipate possible later options, we'll also declare a template of type check to hold what we're matching against. Then the pseudocode would run something like this:

```
get match constraints in template
ask whether user wants hard copy
do_balance := template.payee = ""
AND template.exp_code = ""
 accum := 0.0
 SEEK #cpath, 0
WHILE NOT EOF(#cpath) DO
     GET #cpath, record
  match := SUBSTR(template.date,
   record.date) = 1 AND
         SUBSTR(template.payee,
          record.payee) = 1 AND
        SUBSTR(template.exp_code,
        record.exp_code) = 1
       IF match THEN
           display record
       ENDIF
    IF match OR do_balance THEN
           accum := accum +
           record.amount
       ENDIF
 ENDWHILE
 IF do_balance THEN
     print accum labeled as balance
 ELSE
     print accum labeled as total for
     selected entries
 ENDIF
```

The above is presuming that "display record" will go both to the screen and to the printer if the user wants hard copy; we may not want to implement it that way. (Also, we've not dealt with pagination issues.)

We've been very vague about how the user inputs data and chooses one action or another and about how we display a particular checkbook entry either to the screen or to the printer. There's something to be said for that kind of vagueness. We could just prompt the user for a character, or we might decide to go for something flashier and mouse-driven, but whatever we do, it should not affect the logic of the program. We'd also like to keep things in one place where it makes sense; it saves code and ensures that if we change our mind, the code changes in only one place—i.e. we're talking separate PROCEDUREs for some of these.

As an example, let's write something to deal with input of an integer, and in passing add some error checking that the original program doesn't do. BASIC or BASIC09 will endlessly reprompt the user for input when the reply to an INPUT for a numeric variable is invalid, and reprompt in a way out of the programmer's control. For that reason, the original programmer uses LINE INPUT to get a string, even for numeric fields, and doesn't try to catch errors. Let's see if we can do better. (Using GET instead of INPUT avoids the prompt, and the anal retentive condition on the EXITIF insists that nothing but the number be entered.)

```
    PROCEDURE input_integer
    PARAM row, col:INTEGER
    PARAM result:INTEGER
     DIM   line:STRING[80]

    ON ERROR GOTO 100
     LOOP
        RUN gfx("CURXY", row, col)
         GET #0, line
        result := VAL(line)
   EXITIF STR$(result)+CHR$(13)=line
    THEN ENDEXIT
100      (* We could beep here or
           something... *)
    ENDLOOP
     END
```

The REAL version will need a somewhat more complicated exit condition because one can enter more significant digits than a REAL can handle and because STR$ in BASIC09 insists on returning, for example, "12." when handed the REAL representation of 12.

NOTE: the above only checks for a valid integer. An actual invocation might require, for example, a check number, which must be positive, or an entry number which, unless we decide to bias it, could be either zero or positive. To allow for more stringent constraints, we could add a parameter that is the name of a validation procedure to run, or we could put a loop with the individual constraint on the caller's side, though to keep the error handling consistent and in one place, we'd have to add a parameter to say "we're back here because we rejected the number; beep at the user even the first time around. Stay tuned to see what we decide.

The polymorphic nature of checkbook entries shows up in the code for their input and output—the obvious example is that check numbers only make sense for checks. Another example that we hide from the user on input and output is that amounts deposited or earned as interest are positive, but amounts for everything else are taken to be negative. That last part is easy to take care of for display, just print the absolute value of the amount (ABS() could have simplified the original code, by the way), and on input, negate the amount for everything but deposits and interest before accumulating or writing the entry to disk.

For input, we can selectively skip the input of irrelevant fields either by passing extra parameters to the input routine or by presetting the expense code field in the structure to be filled. If it's the empty string, then we're asking for a check, and we should prompt the user for a check number. Otherwise, we're presumably asking for a deposit, bank charge, or interest, in which case we will skip the input of the check number (and for the expense code, which we already have set). Since we are going to fill in the expense code for these anyway, I choose the preset expense code method.

(The original bypasses the "payee" field for deposits. I think I'd want to be able to record where the money came from, so I can ask for a selective display and total of all the deposits from those big ninety-nine cent dividend checks I get from my three shares of AT&T stock. So...I say we'll always ask for the "payee" field, though it doesn't always represent a payee.)

Come to think of it, we could set aside a special "expense code" for templates, which will bypass entering the amount as well as the check number, though it will ask for an expense code, since we give the user the option of matching against it.

But wait, what about the "correct entry" action? We will want to let the user correct the expense code, so rather than asking whether the incoming string is empty, we will have to ask whether it has the specific values for deposit, interest, or bank charge. We'd rather have a single way to recognize a family of expense codes, though, anticipating possible future changes. Perhaps our special "expense codes" could share a property, e.g. starting with "*" or some other marker, though with enough of this kind of hassle, we may decide to split the variant tag from the expense code.

Due to the length of this "conversion", we'll have to continue in the next issue. I hope that the preceeding discusion pointed out the differences between Basic09 and DECB without offending anyone. See you next issue!

James Jones is an experienced programmer in Disk BASIC, Basic09, Microware Basic, and C. He currently works for Microware. He may be reached for comment via this publication or e-mail as jejones@delphi.com.

Hi-Density Drives for CoCo

Robert E. Brose II

Yes, you can have 1.2 & 1.4MB drives, but only under OS-9.

This mod ONLY works in LEVEL 2 because it requires the higher clock speed of a CoCo 3. The controller MUST be the one with the full sized board, a 1793 controller chip and three adjusting potentiometers (RS Catalog Number 26-3022). Note that a 5 volt only long controller with a 1793 was made. I have only seen a few, and do not know if they have the adjustable pots in them. If so, they may be modifiable also. According to the Western Digital manual, the 1773 (used in the newer controllers) CANNOT do high density.

The controller must work perfectly in LEVEL 2 BEFORE the modification, to be able to be used. Some of these older controllers don't work consistantly at 2 mhz. The problem is most likely to be slow gates on the clock lines, but I was able to solve the problem on one controller by using a better FDC. The original was a FD1793, the FD1793CL-02 works fine as does the MB8877A.

Some facts about hi-density drives

There is a difference in using a high density on a PC as opposed to using it on a CoCo, the PC uses 512 byte sectors, 15 per track on the 5.25" drive and 18 per track on the 3.5" drive. Since OS-9 uses 256 byte sectors it is not always possible to get quite as much storage on the drive. The reason is this, each sector has a header of information preceeding it on the disk (this is the meaning of SOFT SECTORING) so when 256 byte sectors are used there will be twice as many headers as when 512 byte sectors are used (overhead). These extra headers use up space that would be used to store other sectors on a PC. In the case of the 5.25" drive, there was a little extra space left over to begin with so you can usually get 30 spt.

The 3.5" drives are another matter. There seems to be some variance between manufacturers of 3.5" drives as to exactly how much can be squeezed on a track. I have gotten 34 spt for some drives and 32 spt for others. The way to find out is to try 34 spt (the default in the included descriptor) and if the verify after format doesn't work, drop the spt down by 1 (both spt and spt track 0, offsets $1C and $1E in the descriptor) and try again (you need to patch the descriptor and reboot).

The above inconsistancy in 3.5" High Density drives ALSO exists in the 720k variety. Some 3.5" 720k drives, especially older ones (Hitachi for example), will NOT fit 18 256 byte sectors on a track. Looking in the manual for the Hitachi, it shows only 16 spt. By minimizing the size of the headers, 17 can be had, but not 18.

The modification...

This modification is NOT for the faint of heart or those unexperienced with hardware modifications. If you don't know what "piggyback" means when refering to chips, forget it! This modification requires 32 soldering connections, 18 jumper wires and a lot of patience. Do this on a spare controller if you can. The old controller needs 12 volts therefore you MUST have a multipak or equivalent, or have made other provisions to supply 12 volts to the controller. This modification will allow the controller to use either 250 kbs or 500 kbs data transfer rate. This is the difference between the standard 5.25" 360k or 3.5" 720k drives and a 5.25" 1.2 meg or 3.5" 1.4 meg drive.

What you need:

1 74LS74
1 74LS158
1 3.9k 1/4 watt resistor
1 mini DPDT toggle switch (optional)
Wire for the jumpers. I recommend standard wire wrap wire as RS carries. This is very important, DO NOT use thick wire. Wire wrap wire is 30 gauge. Just right for these kind of projects.

The mod will be done so if a mistake is made and you want to abandon it, you can just remove all of the jumpers plug in replacement chips for the ones piggybacked to and you'll be back to where you started. If you want this option, buy an extra 74LS74 and a 74LS221. There are NO trace cuts in this mod. IC pins are left out of the socket to get the equivalent of a trace cut.

If you need to reverse the mod, those pins MUST be reinserted into their respective sockets. There is ABSOLUTLY NO GUARANTEE OR WARANTEE EXPRESSED or IMPLIED FOR THIS MODIFICATION. Now, on to the fun part!!

We will be piggybacking a 74LS74 onto the existing 74LS74 at IC1. We will also be piggybacking a 74LS158 onto the 74LS221 at IC7. Some other chips will be soldered to and some pins will be removed from the sockets for some IC's. These instructions will be entirely verbal, no illustrations.

First, remove U1 (74LS74) from it's socket. Position a new 74LS74 on top of it with the pins EXACTLY overlapping (this is called piggybacking). Be sure both pin 1's are lined up or it'll be poof time when you apply the power. On the upper 74LS74, bend up pins 2,3,5,6,8,9,10,11,12 and 13 so they point directly away from the body of the IC. Pins 1,4,7 and 14 should still be overlapping the lower 74LS74. Carefully solder these pairs of pins together being careful not to blob the solder onto the legs of the lower 74LS74 as you will be plugging the pair (stack) of chips back into the U1 socket when done.

On the lower 74LS74, bend pin 11 out away from the body of the chip as you did for some of the pins on the upper IC. Pin 11 will NOT be going back into the socket. Prepare six 3" jumper wires (prepare means strip back the insulation on each end of the wire, no more than 1/16". Then tin the exposed wire on each end of the jumper). Solder the wires to the stacked IC's as follows. One end of each wire will be unconnected. Solder:

1 jumper to pin 11 on the lower IC (the pin sticking out)

1 jumper to the lower IC pin 3 (must still be able to go into the socket)

1 jumper to the lower IC pin 6 (must also be able to go back into the socket)

2 jumpers the the upper IC pin 3

1 jumper to the upper IC pin 6

Also, prepare a 1.5" wire and solder it from the upper IC pin 2 to the upper IC pin 6 taking care not to disconnect the wire already on the upper IC pin 6. You may now carefully plug the IC stack back into the IC1 socket making sure all pins get seated into the socket with the exception of pin 11.

Second, we'll be doing a similar piggyback mod to the 74LS221 in the U7 socket. Remove the 74LS221 from the socket. Position the 74LS158 on top of the 74LS221. Make sure that the two IC's are properly aligned and that the two pin 1's are aligned together. Bend up all of the pins on the upper IC EXCEPT pins 8 and 16. solder the two pin 16's together and also solder the two pin 8's together. As before, make sure not to blob solder on the legs as the stack will be plugged back into the U7 socket.

Bend pin 13 on the lower IC away from the body of the IC so it cannot be reinserted into the socket. Prepare four 1.5" jumpers, one 2" jumper and one 3" jumper. Solder them in as follows:

1 2" jumper to the lower IC pin 2

1 1.5" jumper from the joined pin 8's to the upper IC pin 15

1 1.5" jumper from the upper IC pin 15 to the upper IC pin 10 (taking care to not disconnect the wire already at pin 15)

1 1.5" jumper from the tied together pin 16's to the upper IC pin 11

1 1.5" jumper to the upper IC pin 7

1 3" jumper to the upper IC pin 1

Plug the stack back ino the U7 socket making sure all of the pins are seated firmly EXCEPT pin 13 which should be sticking out. Solder a 3.9k resistor from the upper IC pin 9 to the side of R18 (3.9k) which is the closest

to the U7 socket.

Final Assembly

Remove U11 (the 74LS629). Solder one of the 3" jumper wires from U1, the upper 74LS74 pin 3 to the top of the 74LS629 pin 7 making sure not to blob solder.

Plug U11 back in making sure ALL of the legs seat firmly into the socket. Unplug U3 (7406 or 7416). Connect the 2" wire from the lower IC pin 2 of the stack at U7 to the top of pin 1 of the IC that was in U3 (making sure not to blob solder on the leg). Plug U3 back into it's socket making sure all of the legs seat firmly into the socket.

Solder the open end of the jumper connected to U1 lower IC pin 11 to U7 upper IC pin 4

Solder the open end of the jumper connected to U1 lower IC pin 3 to U7 upper IC pin 3

Solder the open end of the jumper connected to U1 lower IC pin 6 to U7 upper IC pin 2

Solder the open end of the jumper connected to U1 upper IC pin 6 to U7 upper IC pin 5

Solder the open end of the jumper connected to U1 upper IC pin 3 to U7 upper IC pin 6.

Choose 1 of the following select methods.

Select option 1

Using WRITE PRECOMP bit and a SWITCH

I use the WRITE PRECOMP bit for controlling the HIGH/NORMAL density. Any other bit could be used instead (for instance, if you were always going to use drive 1 as a high density drive, you could use DS1 as the density select). I used WRITE PRECOMP because I don't use DECB, I boot directly to OS-9 on a hard drive without a floppy and OS-9 Level II doesn't use write precomp.

When using the WRITE PRECOMP BIT as the density select, you will need to set the density enable switch to the normal position while booting OS-9 from a floppy. This is because DECB sets write precomp for any track greater than 22 regardless of whether the controller will be doing a read or a write. You can move the switch to the high density select position after booting. If this seems like too much work, use another bit. (like DS1).

Mount the dpdt mini switch somewhere handy. I mounted mine in the hole near C1 and the piggybacked 74LS74's. Make sure that the switch DOESN'T SHORT OUT any traces! I'll refer to the switch pins as follows:

1 2 3 pin 2 toggles between pins 1 & 3

4 5 6 pin 5 toggles between pins 4 & 6

Carefully remove U12 (the 1691) from its socket. Bend up pins 9 and 16 away from the body. Put the 1691 back into the U12 socket making sure that all pins firmly seat with the exceptions of pins 9 & 16.

Prepare and solder a 4" jumper from U12 (1691) pin 9 to switch pin 5.

Solder the open end of the jumper connected to U7 upper IC pin 7 to U12 pin 16.

Solder the open end of the jumper connected to U7 upper IC pin 1 to switch pin 2

Prepare and solder a short jumper from the DPDT switch pin 3 to the DPDT switch pin 4

Prepare and solder a short jumper from the DPDT switch pin 4 (taking care to not disconnect the wire already there) to a convenient ground (for example, IC1 pin 7 on the SOLDER side of the board).

Prepare and solder a short jumper from the DPDT switch pin 1 to the DPDT switch pin 6

Remove U8 (the MC14174) and prepare a 3.5" jumper. Solder a wire to the top of pin 12 without blobing solder on the leg. Plug U8 back in making sure all of the pins seat firmly into the socket.

Solder the open end of the jumper connected to U8 pin 12 to the DPDT switch pin 1 taking care not to disconnect the wire already there.

Skip to check procedure below.

Select option 2

Using a DRIVE SELECT BIT.

Carefully remove U12 (the 1691) from its socket. Bend up pin 16 away from the body. Put the 1691 back into the U12 socket making sure that all pins firmly seat with the exception of pin 16.

Solder the open end of the jumper connected to U7 upper IC pin 7 to U12 pin 16. Remove U2 (7406) from the socket.

Choose a drive select line to use, either DS1 or DS2 (DS0 should not be used or you will not be able to boot, DS3 is usually used to access the back side of double sided drives so that cannot be used either). Solder a 2" jumper to pin 3 (DS1) OR pin 5 (DS2) without blobing solder on the leg. Plug U2 back into it's socket making sure all pins seat firmly. Solder the open end of the jumper just attached at U2 to U7 Upper IC pin 1.

Check Procedure

Now recheck the entire procedure to make sure no mistakes were made. Check all soldering joints for good connections. Check for shorts, especially by the DPDT switch. There should be NO unconnected jumper wires! If there are, go through the entire sequence to see what you missed. Now, we need to calibrate and test the controller.

It is advised that you use a multipak which will protect the CPU (you need +12 anyway) in case you made a fatal wiring mistake. Plug the controller into slot 4 as usual. Power on the multipak, then the computer. If the DISK BASIC message doesn't come up quickly then shut the computer off immediatly and power everything off. Unplug the controller and check for shorts and recheck all connections against the modification procedure. If all else fails, you can always remove the piggybacked stacks at U1 and U7, carefully pull off all of the jumpers, insert a new 74LS74 into U1 and a new 74LS221 into U7, pull out U12, carefully bend pins 9 and 16 back down and reinsert it into it's socket, remove the switch and you'll be back to where you started. Presuming you made it past the smoke test, you will need to figure out your switch position and calibrate the controller.

Switch position determination

(skip if drive select method used)

When the switch is in the position such that pins 1 & 2 are connected together (also pins 4 & 5) the controller is in the HIGH DENSITY enabled position (use a meter to test the connection between pins 1 & 2). When the switch is the otherway, the normal configuration is active, which means write precomp is available. Put the switch into normal position for calibration.

Calibration of the VCO

The controller can be calibrated either with a scope or by trial and error. Either way, mark the original position of R8 so you can reverse the modification if you can not get it to work right.

If using a scope, connect the scope to the VCO output of the 74LS629 (U11) pin 7 and adjust R8 for 4 mhz. If doing the adjustment by trial and error, put a formatted RSDOS disk into drive 0 and do a DIR from RSDOS. Turn R8 until you can get a directory. You may have to do lots of DIR commands. Try to find the extreme settings of R8 that will still produce a directory, then set R8 between the two extreme settings. The range in which the DIR will work will be quite small and your final setting for R8 should be as close as possible to the middle of the range. THAT'S IT FOR THE HARDWARE.

The Software...

The cc3diskhigh.ipc file will make a new cc3disk from the ORIGINAL Radio Shack cc3disk edition 9 CRC $759161. The new cc3disk detects the 8" drive bit in IT.TYP in the drive descriptor and uses it to switch the data transfer rate on the modified controller. The new cc3disk will access both normal and high density drives.

Now, apply the patch to CC3DISK, pick your descriptor (there are 2 descriptors included, one for the 5" 1.2 meg drives and one for the 3.5" 1.4 meg drives). make a new OS9 boot disk. After booting put the switch (if you have one) into the HIGH DENSITY ENABLED position and you're ready to go. cc3diskhigh.ipc and the drive descriptors will be on this issue's "microdisk". They are also available in the DelphiOS-9 SIG database under the "general" topic.

In the next issue, we will present another hi-density modification. This one will use the hi/low density switch built into 3.5" drives to allow using either size (the mod above only uses hi-density disks).

Internet Connections

Some tips on using Delphi's Internet

Using Internet FTP from Delphi

FTP stands for "File Transfer Protocol". This feature allows a person to transfer files from a remote Internet site anywhere in the world to their own computer, much as one would download from Delphi or a local BBS. I asked someone on Delphi to give me a step by step set of instructions for using FTP, since many people I've talked with seem to have trouble with it (including myself!). Gene Heskett replied with the following, which is based on access from Delphi. John Baer also added a few tips. Other services with Internet access should be similar.

1. First you must have Internet access.

2. Then, from the root directory of delphi, where you can see the "internet services" entry in the menu, enter "internet" w/o the quotes (type "MAIN" from any SIG to get to the root directory).

3. When the internet menu comes up, type "ftp" without the quotes.

4. The prompt will change to "FTP>" and it will ask you for your destination. Enter it as you see it listed, no quotes. The old cabralas site, for instance, is now chestnut.cs.wisc.edu. Enter the site name in lowercase as here. (Editor: The chestnut site may no longer be up. I'm not sure about this and didn't have time to check it out before printing... sorry!)

5. When it connects, FTP will assume a default name of anonymous and will accept the default of your delphi handle with @delphi.com appended as a password (i.e. – dsrt@delphi.com). All you have to do is hit the enter key to have it accept those defaults.

6. Once into the remote site, I do a 'dir', no quotes, in lowercase. This will show you the present directory's contents and perms, almost as if you were at the keyboard of the remote system and accessing the drives/ directories directly.

7. I was able to 'cd pub' ok, but found from there on that any deeper cd's would have to be in the form of cd "COCO3" (quotes included).

8. Once there, or anyplace else you might want to go up the directory tree, do another 'dir' to see whats there. Then either cd "NEWDIR" to go up one, or cd .. to back out, just as if it were an OS-9 system. In this present case, I did a cd "MISC", then another dir to find the file I was interested in, ansifront010.lzh.

9. to move this file to your delphi workspace, first type (in uppercase) BINARY, then press ENTER, of course. Wait until the prompt comes back, allthough it may be that the site's operating system can stack the commands (I did see it do that once). BINARY can also be typed in lower case without the quotes on Delphi.

10. Now type GET filename(enter). FTP will report as it has before, but I've not noted here, that a path has been opened. There will, however, be no more response from FTP until the file has been moved to your workspace on Delphi. Remember that Delphi is a VAX system. A single 'dot' is needed in a filename. So, if you see a filename like: big.utility.UPDATE. lzh and want to 'get' this, try it like this..

get "big.utility.UPDATE.lzh" stuff.lzh

Just quote the filename as you see it, and shorten it to something you will remember with just one 'dot' between the filename and extension. Delphi will send the string you quoted, and use the second part as the filename for your workspace. Rename it to what you want after downloading to your home computer.

11. Once that has been done, you can either resume your explorations of the remote system or sign off with the usual "bye"(enter)... no quotes.

12. That will return you to the internet prompt on delphi. Hit the usual control z to get out of that, go to your workspace and type zdow (or Xdow, or ydow... whichever protocol you normally use).

13. That's all folks! Hopefully, you all know the rest of how to download a file from Delphi.

Using a Lynx Browser

Chris Hyde leaves us this tid-bit of information on using a Lynx text browser from anothe computer site:

You can travel on the Web from Delphi if you have an Internet account. There are 2 browsers available: a native one running on Delphi (not very good) and a couple of lynx browsers running on various host systems that you telnet to. These are much better and offer some form of support (not as nice as Mosaic or something running on a SLPP line but not too bad).

To use the lynx browsers from the Internet main menu type:

gopher
19
2
9

(This is the browser I use, there are a couple of others)

Using SIG Links...

Susan Drudings, who manages the Hobby SIG area, tells us how to use "linked" areas from Delphi. A linked area is one that automatically takes one to an Internet site.

It is really easy to read WWW stuff on Delphi. You cannot "see" the images as you can on SLPP connections with Mosaic and Netscape but you can read all the text, and you can download many images.

Go to the INTERNET Menu in the Hobby SIG (or for a place that has a LOT of sites hooked up go visit my Travel Talk CF96 where I have a couple dozen linked up). To do this, type INT at the Main Hobby SIG menu.

You will see a list of places numbered. There are two ways:

#1– You can choose a WWW site I have already hooked up (there are several in the Railroading topic, a couple of nice ones in gardening, a few in Models) and elsewhere). Set your screen length first to keep it from scrolling — type /length 24 that will give you one screen of data at a time to read.

You will see things with numbers in brackets like this: [3] [16] etc. When you choose one of those numbers and type it you will be "hypertexted" to that item. You can keep browsing by typing these numbers and reading.

On most places you will have some numbers given at the bottom of the screen to return to the Home Page or to back up. BA will usually back you up one screen and PREV will usually take you back one menu level. Ctrl-Z will also back you out of most places.

#2– You can experiment with a place you have read about in a computer magazine by choosing the number on the first screen that says something like ACCESS ANY URL Here you can type in those weird addresses you see that look something like this: http:/ /www.marvel.loc.gov You have to type it EXACTLY the way it looks (including some occasional tilde signs: ~ which are used.)

Have fun! This is addictive, so you are warned.....

(Seen in a Doonesbury comic strip - Doonesbury has stayed up all night again "Surfing the Net" and his kid is yelling "Mom. Dad stayed up all night again!" IT HAPPENS!!!)

I received the following from the Atlanta Computer Society. I edited the original letter some, but the main point is that they are asking our help in planning future fests. Please take the time to answer them, even if you don't plan on or can't attend the fest. Your ideas will surely help! To make it easy for the guys in Atlanta, please write the questions and answers in the same order. I've also included my own answers.

Dear Computer Enthusiast,
We at the Atlanta Computer Society, Inc., in an effort to promote future computer festivals, are asking for your help. We are sending this questionnaire to vendors and past Atlanta CoCoFest attendees and ask you to send us a letter answering the following questions. Just a few minutes of your time will aid us in keeping your best interests in mind as we make decisions concerning future fests in Atlanta. Thank you for your cooperation.

1. If you did NOT attend the 6th Annual Atlanta CoCoFest, tell us why.
Not applicable... I was THERE!
2. If you DID attend the 6th Annual Atlanta CoCoFest, tell us why.
I'm a vendor, and Atlanta is only a two hour drive from my home.
3. What did you not like about the fest?
Well, there could always be more vendors, but that is a market problem, not yours.
4. How can we improve the fest?
The organization of the fests have always been pretty good. ACS has been doing an excellent job... keep up the good work!
5. Should we open our fests to other "orphan" computer systems?
This may be a good idea, it is certainly worth exploring. Why not contact some of the local clubs (to Atlanta) and see what they think? Just to tie it in, maybe keep it to the less popular 68K machines, namely the Atari ST/Jaguar and Amiga systems. A Mac crowd may crowd us out later.
6. What would increase interest and attendance at our next fest?
Inviting other systems in would definately increase attendance. Promotion as a "CoCo and OS-9" fest should help also, might get Peripheral Technology and some other industrial companies in the area interested in having booths.
7. What would induce you to attend our next fest?
If I'm still in the area, I'll be there!
8. What is the best method of advertising future fests to reach more people?
This magazine and the Internet 6809, 68000, CoCo, and OS-9 groups are the best advertisement.
9. Do you have any other comments or ideas?
I would possibly be able to come to Atlanta for a meeting on some of these subjects. If not, maybe a conference on Delphi or some Internet sight might be possible. Both ideas of promoting as CoCo and OS-9 and inviting some other groups have great possibilities and should be seriously considered.

Operating System Nine

Installing Software

Rick Ulland

The Color Computer is a strange beast indeed. Born a contempory to other 'home' computers like the C-64 and 99/4a, it provided many a young hacker an uncommonly powerful box on which to develope their skills- not only outliving it's peers, but finally it's own BASIC. A CoCo 3 is the only machine I'm familiar with that has to update its very ROMs every time it's turned on!

But the CoCo has another side. Unlike it's former counterparts, CoCo wasn't limited to the semi-proprietary ROM BASIC + assembler, abd had a completely separate identity as an OS9/6809 platform. If CoCo haad stopped there, that would have been noteworthy- but with later hardware enhancements it can run software never intended for small home computers.

This stuff is OLD, and naturally, such software has it's problems. Much of it was expected to run on multi-user systems, under the watchful eye of a supervisor- much like the multiuser Unix or Vax software of today. User friendly it ain't. A second problem stems from the age of the software- stuff written for 'personal' computers expects floppy drives at best, and even the hi-end wares could hope for little more than character based terminals.

Lest we forget, Tandy also released software modified specifically for the CoCo- at least, some CoCo. Even this stuff is not properly set up for the average CoCo of today. This month, we'll set it up.

Display:

If you are lucky, the program you're loking at has some sort of configuaration file, like a termset or termcap file. Unlike MSDOS stuff, there usually isn't a handy menu in the program to change this file, but the best of them let you get in with a text editor. DynaStar is a good example- with separate entries for practically everthing (resolution, colors, etc) for each possible window, and the serial ports! Even the control codes are editable, so they supplied entries for popular termnials of the day. Note that VT100 is a superset of 'ANSI' so the ANSI entry will drive a VT100 terminal usably well.

The next level is a cryptic binary config, and unless you can find it's format someplace you'll need to look for a patch. Tandy had a lot of software like this- among the apps, Dynacalc and tsedit are good choices to patch for 80 columns. (The tsedit patch is renamed to 'vi', and makes a perfectly usable line editor).

The least friendly displays are hard coded right inside the program. Even if you could recognise the part you want to change, the modules CRC has to be changed before your efforts will work. Which calls for a disk editor or a canned patch file. Luckily, the are now patches for just about anything you'd care to patch, and if the program is text based there is a good chance it can be tweaked to 80x24.

In the most extreme cases, the program is so intricately tied to the old 'vdg' style display that it can't be modified to run on a Windint style window at all. Many of Tandie's games fall into this catagory. These can still be run under Windint/ MultiVue, by creating the vdg stile window it craves.

To do this, you'll have to have vdgint in your boot. This will use up some system RAM, so you might find the 'game' boot has problems, like the inability to format a disk. With stock OS9, about the only effective solution is to maintain two boots. You'll also have to use Tandy's xmode utility- the type command changes a normal window into a vdg window. Type=1 gives a 'ti vdg' thing with real lowercase, while type=0 gives a true blue CoCo Two inverse character desenderless wonder.

This can cause friction between the xmode used by SACIA or Fast232. This script assumes the newer xmode is usually in RAM:

```
unlink xmode
load xmode.tan
xmode /w14 type=1
iniz /w14
shell i=/w14&
unlink xmode
load xmode
```

Replace the word 'shell' with any Tandy game, and make the AIF/ icon call your script. Otherwise- some programs (Deskmate comes to mind) are further restricted and will only run from the console (term) window itself. In the case of the DeskMate patch, it's worked out by dropping the menu, and running the apps individually under MultiVue.

Mass Media:

In the days of level one, programs had to specify a particular hard drive, so many programs intended or modified for the CoCo will spec /d0 and /d1 as primary storage. This is, at best, undesirable- a modern 3.5 inch drive can hold more information than all four drives of a CoCo1, and there is the added complication of hard drives.

Luckily, OS9 modules refer to disk drives by the logical names you are used to (/d0, /d1, /h0, etc.) which makes them fairly easy to spot when stepping through the program with a disk editor. Nowadays, the preferred mass storage device is /dd. Remember, the last character is often marked by setting the hit bit- so dd becomes 44 C4. Check the code as well as the ascii represntation before changing anything.

Political Humor:

While you can point any program to a hard disk, the easiest ones simply use the current data path for storage. Programs that assume small floppies sometimes insist on specifing something, and don't leave

continued on page 16

Roadmap to the Internet

Patrick D. Crispen

Spamming Urban Legends

"Well, there's egg and bacon; egg, sausage and bacon; egg and spam; bacon and spam; egg, bacon, sausage and spam; spam, bacon, sausage and spam; spam, egg, spam, spam, bacon and spam; spam, spam, spam, egg and spam; spam, spam, spam, spam, spam, spam, baked beans, spam, spam, spam and spam; or lobster thermidor aux crevettes with a mornay sauce garnished with truffle pate', brandy and a fried egg on top of spam."

--- Monty Python's Flying Circus

It's possible, even easy, to get a list of every Usenet newsgroup and publicly accessible LISTSERV list. With very little thought, you can convert the list into a program that will mail the same message to every single one of these groups. Doing this is called "spamming", after the Monty Python sketch quoted above.

During the past year, there have been three such mailings that have "succeeded": One poster said that the end of the world was nigh; another advertised the services of their law firm in the so-called "Green Card Lottery" message; and a third, labeled "MAKE.MONEY.FAST" was the Usenet equivalent of the old chain letter.

Of the three, the one that got the most attention was the Green Card Lottery spam (1). According to the Washington Post, the law firm in question considered the Internet to be "an ideal, low-cost and perfectly legitimate way to target people likely to be potential clients."

Many people felt differently, though. They felt that, first, the Internet is the wrong place to conduct commercial business. Many of the charters of the Usenet newsgroups and LISTSERVS specifically prohibit offers to do business. The few that do accept offers restrict the buyers and sellers to individuals, not businesses. The net has had a long tradition of non-commercialism, ever since its founding days as ARPAnet.

Second, the net isn't free. One popular newsreader, "trn", displays the following message before it lets you post:

"This program posts news articles to thousands of machines throughout the enter{sic} civilized world. Your message will cost the net hundreds if not thousands of dollars to send everywhere. Please be sure you know what you are doing.

Are you absolutely sure you want to do this? y/n"

Since the spammers are alleged to have posted to over 6,000 groups, they surely spent quite a bit of somebody's money.

Finally, people who gather together to discuss a topic get annoyed when someone discusses something outside the group's charter. They often complain to the newsgroup itself, thereby increasing the traffic even further.

Note that spams generally aren't crossposted. That means that every news host will receive, process, and make available to its readers a separate copy of the spam for every newsgroup. Of course, "courteous" spammers who use crossposting can make things even worse. In one recent spam, not only was the spam sent to all sorts of unrelated newsgroups, but so were the angry replies! (The people replying were guilty of not reading their "To:" and "CC:" lines before they posted).

WHAT TO DO WHEN YOU SEE A SPAM:

First, NEVER reply to the group. The spammer won't read it. He's interested in talking, not listening, and he isn't a list member or a regular reader. Your angry posting will only annoy the other members of the group, and won't affect the spammer in the slightest.

Second, if you have a lot of time on your hands, you may read the responses of members who ignored my first bit of advice. On comp.os.vxworks, for example, one (moderately clueless) member posted (in response to the end of the world spam) "This isn't a religious newsgroup!" An old-timer responded "I think that very much depends on the topic. ;)." (that's a winking smiley)

Third, if you have even more time on your hands, reply to the poster at his own mailbox. But you may not get satisfaction. Quite often spammers hit and run, and by the time you get back to yell at them, they've closed out their accounts (or if their site administrator is on her toes, they'll have had their accounts closed by the administrator).

Fourth, if you're even angrier at the spammer, you can write to the administrator of his site. If the spammer is clown@circus.com, his administrator is postmaster@circus.com.

Fifth, and this is net abuse that can get you removed by your site administrator, you may want to mailbomb the offender. That consists of sending him lots and lots of email until his site or his account crashes. And, yes, it is perfectly possible to make a machine crash, taking down all its users, by sending too much mail to a person on that machine. The same thing can happen to gateways processing the mail.

What I do is *think* about mailing offenders the Manhattan telephone directory. In PostScript. I enjoy the thought without abusing the net myself. Yes, you have it within your power to spam the world, or to mailbomb (mostly innocent) people. You also have it within your power to buy a gun and start shooting at people. That doesn't mean you have to do it.

(continued on page 17)

Basic09 In Easy Steps

Chris Dekker

Advanced Programming: RunB Issues

Operators, Strings and Control Structures

Operators like =, <and> are supported by different types of subroutines inside RunB. These routines deal with byte/ integer, real number or string type variables. The subroutine for bytes/ integers looks like this:

```
LDD 7,Y
SUBD 1,Y
BLT true
BRA false
```

As you can see, there isn't much to it. You substract the two variables and compare the result to zero. This code is executed when your program reads like this: IF x<y THEN ... "true" and "false" are labels. They point to other subroutines that set the value of a boolean variable to true or false. This variable is returned to RunB, which then continues program execution based on the variable's value.

BLT is a 6809 mnemonic that means "branch if lower than". To see if it should branch to "true" here, the 6809 checks 2 flags in the condition code register. (Yes, the same regs.cc we check for errors after executing a system call.) If the 2 flags differ, the processor starts executing "true". If they are the same (both set or cleared), it executes the "BRA false" instruction.

RunB contains an entire battery of such routines: one for each operator or combination of them (e.g. <> or <=). The only difference between these subroutines is that the "BLT" instruction is replaced by another instruction. This causes the microprocessor to check other (sets of) flags in the condition code register. [If you're not familiar with ML programming, the SUBD instruction automatically sets/clears these flags depending on the result of the substraction.]

There are similar sets of subroutines for strings and real numbers. The difference being that you can not substract strings or real numbers in the same way as integers: the routine for real numbers compares two values (which is quicker then substraction), while strings are compared on a character for character basis.

Strings are actually compared by comparing the ASCII values of the individual characters. If they are the same, the next set of characters will be compared. If not the condition code flags are set according to which of the two characters is the smallest and processed in the same way as described above.

This works great except that it makes the comparison case sensitive. If you want to compare strings without distinguishing between upper and lower case, you're out of luck. You will have to do a conversion in Basic (which is slow), do the comparison in Basic (even worse), use the Compare system call (takes some setting up to stay out of trouble) or write your own ML subroutine.

Now that we are on the subject of strings: What's faster: LEFT$ or RIGHT$? Or is there no difference? Actually there is and LEFT$ is faster, although you will need a very long string to actually notice the difference.

The reason is quite simple: under Basic09 all strings start in the leftmost position of their alloted space. Although this seems logical enough to us, it means that RIGHT$ (and MID$) have to do an extra copying operation to shift the portion of the string they have to return to the left. By the way, MID$ will jump to LEFT$ if you specify 0 as the starting column.

LEFT$ works as follows: it calculates the new end position of the string. If this is beyond the current end, it leaves the string alone and exits. Otherwise it will copy $FF (Basic09's string delimiter) to the position right after the new string end. It also updates some pointers. All of this takes anywhere from 15 to 40 MPU cycles. Of course you have to add to that the overhead of copying strings to and from the expression stack. This adds at least a few hundred cycles.

RIGHT$ and MID$ have (besides abovementioned copying) some extra checks to make. RIGHT$ should generally take 50 to 100 MPU cycles and MID$ 100 to 150 cycles. TRIM$ is also a very short routine that can be compared with LEFT$ in speed (unless you have lots and lots of spaces to trim).

SUBSTR can run in a 100 or so cycles, but may also stretch into the thousands. This entirely depends on the strings you pass to it. That leaves us with VAL and STR$. I won't even try to guess how long it takes for these functions to execute. This is due to the variety of variables these routines have to deal with.

VAL and STR$ both enter large subroutine packages (approx. 200/160 lines respectively). Both also call other subroutines inside loop structures. So, if you want to write a speedy program, don't use them more often then absolutely neccessary.

The same goes for the innocent looking print USING statement. Believe it or not but the subroutines associated with this command make up close to 10% of the RunB module. Now, you won't need all that code to print a single space; but the overhead associated with PRINT USING compared to a simple PRINT is substantial.

Personally, I use PRINT USING only to create decent looking tables. For that application you need complete control over positioning or the results may not be something to brag about. Other than that:

```
PRINT " ";var  and
PRINT USING "x1,i2";var
```

have the same result but the latter is quite a bit slower.

Now, last but not least: control structures.

These includes statements like GOTO, GOSUB, IF/THEN, LOOP/ENDLOOP, etc. To start with the easiest of all: GOTO. RunB takes this quite literal. The code implementing GOTO looks like this:

```
LDD ,X
ADDD modstart
TFR D,X
RTS
```

A little explanation might be in order. To make this work Basic09's assembler converts the line number in a GOTO statement into an offset relative to the

starting address of the module. This address is loaded into the variable "modstart" when RunB starts executing a module.

All RunB has to do when it encounters a GOTO statement is: add the offset to the starting address and transfer the result to the X register. The X register is used as pointer. It always points into a packed Basic09 module, so RunB knows where the next instruction or variable can be found.

Why all this fuss about GOTO? Well, first of all it shows how fast these jumps get executed. The above code takes 23 MPU cycles (17 cycles for a 6309) and has very little overhead: just looking up the postion of the GOTO subroutine in a jumptable and entering it which takes 25-30 cycles.

The second reason is that RunB uses this code a lot. This must be a hair raising experience for people who insist that any program with a GOTO statement is inferior, but it's true. For instance ENDLOOP is literally translated into: GOTO LOOP. END WHILE, ENDEXIT and ELSE also jump to this subroutine. Control statements like EXITIF, UNTIL and NEXT branch to this routine if a certain condition is not met.

GOSUB (and RETURN which almost executes GOSUB backwards) are a little more complicated then GOTO because they have to preserve the contents of the X register (so your program knows where to return back to) and check the stack for over/under flow errors. Nevertheless the job gets done in about 50 MPU cycles. This beats starting a new module by a mile (or two). So don't get in the habit of converting each 5 line subroutine into a separate module.

Creating your own jumptable with ON x GOTO/GOSUB adds at least a few hundred MPU cycles, but is still a lot faster than going through an entire series of IF/THEN statements. This is due to the fact that every IF is followed by a jump to the routines I described at the start of this article. The THEN (and ELSE) statements have been replaced by offsets that are processed by the GOTO routine.

There is one set of program flow controls whose code looks entirely different than that of the others. I am talking about FOR/NEXT/STEP. Although easy for us: what could be simpler then FOR i=1 TO 10 ... NEXT i?; it is by far the most complicated control structure for Basic09. The code even has it's own jumptables because it comes in 4 variations: integer (step 1 & step x) and real (step 1 & step x). Having said that, Basic09's assembler does an excellent job setting things up so there is probably little difference in execution speed with other loop structures. My best guess is that WHILE/END WHILE is somewhat faster if you can use an implicit counter (one that is already used for another purpose). However if you have to define and increment a separate counter, FOR/NEXT seems somewhat faster. Anyway the difference is too small to make much difference for your program's performance.

So,, there you have it!! Now that we all have some idea why certain programs zip along while others crawl, you, too, can be on your way to writing that impressive program you always dreamed about!

Chris can be reached in care of this magazine or directly at:
Chris Dekker
RR #4
Centreville, NB E05 1H0
CANADA

Operating System Nine

continued from page 12

enough room to insert much of a pathlist. One workaround is to change the internal spec from /D0 or /D1 to dot- (the period character) which is transmorgified to the current data pathlist. Under Multivue, also consider a path of 'dimame' which will hide the data files above the AIF directory you display.

Nicer programs will add a string of 'pad' characters after the internal spec, usually ascii'ed as periods or commas. Sometimes there will be a modifed period ($0D vs $0A) marking the end of a text phase- if you overwrite this make sure to replace it at the end of your $, without overrunning the pad space available.

As an example, Computerware released quite a few business programs written in their own BASIC. The ones I've seen all used a separate set of parameter files, which stored everything from the company name to which drive the 347th data file resides on. The main change is simple, just search out all instances of /D0/CMDS and change them to /dd/cmds (after, of course, moving all the execs to /dd/cmds). Also search out the data locations (often /d1/something). Since these programs were written with 158K floppies in mind, it's usually ok to change everything to /dd.

RAM constraints:

Since Level Two insulates programs from the hardware's RAM configuration, as long as you have some left this usually isn't a problem. Especially with game boots (vdg added) there is the chance of running out of space in the system's 64K (237 error), otherwise most programs use 64K at most, with perhaps 32K more per each additional copy running. Many use much less.

There are three games I know of (Sierra's King's Quest 3, Lesiure Suit Larry, and SubLogic's Flight Simulator) that won't run under normal OS9, They violate the 64K per program rule and need a custom memory manager. Once again, you have to add things to the boot. If you only run the Sierra games, or only run the sublogic simulator, simply add the 'strange' modules from each games boot disk to your normal boot (***aciavidqr? or FT and FTDD)- along with vdgint, of course.

An alternate plan is to obtain Bruce Isted's 'VRN' package, which replaces all the special purpose modules with one set of code. This is the only way to set up everythingthing in one boot,mand at that, it's a little tight. But it does work! Note that all Sierria games use the same executable name 'sierria' so it's not a bad idea to create an independant cmds dir in your data dir, and chd and chx from the proc file.

Rick can be found in care of this magazine or via e-mail: pulland@omnifest.uwm.edu

Roadmap...

(continued from page 13)

URBAN LEGENDS (ULs):

Another example of spamming on a much smaller scale, at least in my mind, are the urban legends that simply refuse to die. There is no better example of an urban legend than the story surrounding Craig Shergold (this as a TRUE urban legend, btw).

"There once was a seven-year-old boy named Craig Shergold who was diagnosed with a seemingly incurable brain tumor. As he lay dying, he wished only to have friends send him postcards. The local newspapers got a hold of the tear-jerking story. Soon, the boy's wish had changed: he now wanted to get into the Guinness Book of World Records for the largest postcard collection. Word spread around the world. People by the millions sent him postcards.

Miraculously, the boy lived. An American billionaire even flew him to the U.S. for surgery to remove what remained of the tumor. And his wish succeeded beyond his wildest dreams; he made the Guinness Book of world records.

But with Craig now well into his teens, his dream has turned into a nightmare for the post office in the small town outside London where he lives. Like Craig himself, his request for cards just refuses to die, inundating the post office with millions of cards every year. Just when it seems like the flow is slowing, along comes somebody else who starts up a whole new slew of requests for people to send Craig post cards (or greeting cards or business cards -- Craig letters have truly taken on a life of their own and had begun to mutate). Even Dear Abby has been powerless to make it stop."

The current variation on the Craig story that is floating around the Internet is that you should send your cards to the Make A Wish foundation in Atlanta, Georgia. Please do not do this. Make A Wish (a foundation that grants the dying wish of children with terminal illnesses) has enough to worry about.

Other urban legends currently making their way around the Internet include a story that gangs are driving around at night with their headlights out and then shooting anyone who "flashes" them with their high beam headlights, and that there is a "virus" called CD-IT that is eating the hard-drives of stupid people. The "lights out" story may be true, but the police departments in Chicago, New York City, and Los Angeles all told me over the phone that the story was false (I called). The CD-IT story it true, but it is FOUR YEARS OLD!!!

I am going to share with you the number one rule for Internet discussion group survival: only post things that are relevant to the topic that the discussion group was created to discuss. The Craig Shergold story would have died a peaceful death years ago if people had only remembered the "relevant posting" rule.

HOMEWORK:

1) If you are really interested in urban legends, there is a Usenet newsgroup (alt.folklore.urban) that you should check out.

2) If you want to see a cute example of what a flame war really looks like, my dad recently recorded a flame war on a relatively calm Usenet group. That file is now on the LISTSERV file server at the University of Alabama under the name FLAME WAR. Please feel free to GET this file.

Linux 68000?

Okay, I promised some information reagarding the Linux (promounced "LI-nucks") operating system and the current generation of popular OS-9/68000 machines. First some bacground on Linux.

Linux started as a "free" alternative to the Unix operating system. It is multi-user and multi-tasking, and totally modular. It is easily customized to fir any users needs, and most Unix utilities and programs can be ported to it. Sound familiar? It should... that is why many o f you have chosen to play with OS-9! Unlike OS-9, Linux source code is free and can be ported to almost any platform, including 680x0 based machines. The most popular form, of course, is that which runs on 80386 or higher Intel compatible machines.

I did mention 80386 or higher, didn't I? Well, one reason is that these are the first Intel chips to really be capable of smooth multi-tasking. Secondly, they have built in memory management, if not in the chips then in the chipsets used for the motherboards.

If you are thinking you'd like to see Linux on your MM/1 or Delmar System IV or V, forget it. There ARE ports to 68K based systems, but the minimum requirement is a 68020 with a standard Motorola memory management unit (MMU). This even knocks some older Sun systems out of the picture, as they used a custom MMU.

There are ports available on Internet sites for some Atari and Amiga models that meet or exceed the minimum specifications, and some Sun and VME card based systems. If you have one of these (An Amiga 2000 or higher, and the Atari Jaguar... maybe a Mega ST with some brands of accelerator boards) you may want to use the various search engines and find some of the Internet sites. A good place to start is the Delphi newsreader from the Internet menu. Search for Linux.68K.

Frank Swygert

for all your CoCo hardware needs, connect with

CoNect

449 South 90th Street
Milwaukee, WI 53214
(pulland @omnifest.uwm.edu)

The main problem with OS9 under a CoCo is the serial port. With a one character buffer, it's hard to do much before the serial port needs service. Our Fast232 port uses a 16 byte buffer to alleviate missed characters at any speed! Not only that, but one can add a second port via a daughterboard at a very reasonable price. And to top it all off, the Fast232 is easily configured to match any system via easily placed jumpers.

Tandy with Sacia		**Fast232**	
bps	load ____ thruput	load ____	thruput
2400	28.3 sec 237 cps	25.3 sec	235 cps
9600	73.6 sec 938 cps	31.4 sec	950 cps
57600	not availabie	32.6 sec	5373cps

Local machines withfaster modems can now be connected to properly (or improperly at 115K!). OS9 drivers by Randy Wilson. Free bonus software! The pd release of 'SuperComm' (Dave Phillipsen and Randy Wilson). All software includes 6809 and 6309 versions.

Fast232 $79.95
Second port $45.00

FINAL SALES FROM DISTO!

I have VERY FEW items left in stock. If you have wanted any Disto products, NOW is the time to call! FARNA Systems will be selling some items at this year's Chicago CoCoFest (and Atlanta, if anything is left!). After the remaining stock is depleted, THERE WILL BE NO MORE DISTO ITEMS MADE! Some items left include:

1. **"Inside 2-Meg"**: A technical booklet that fully describes how the DISTO 2-Meg Upgrade kit works. Includes schematic, PAL listing, theory and chip by chip circuit explanations. $20 + $2.50 S/H.
2. **"Blank Board Kit"**: Includes blank virgin boards (no components) of the SCII, SCI, 4IN1, MEBII, MPROM and Mini Controller. Collect all the components and make your own! $29.90 + $4.50 S/H.

DISTO
1710 DePatie
St. Laurent, QC H4L 4A8
CANADA
Phone 418-747-4851

FARNA Systems

Software, Books, and Hardware for all OS-9/OSK Systems!
ADD $3.00 S&H
($4.00 Canada, $10.00 Overseas)

Box 321
Warner Robins, GA 31099
Phone 912-328-7859
Internet: dsrtfox@ delphi.com

Software:

CoCo Family Recorder (DECB) - $15.00
Genealogy program for CoCo 3. Requires 2 drives, 80 col. monitor.
OS-9 Version - **$20.00**

DigiTech Pro (DECB) - $10.00
Record any sound for easy play-back in your BASIC or M/L programs. CoCo 3 512K.

ADOS (DECB)
Support for double sided drives, 40/80 tracks, faster formatting, much more!
Original (CoCo 1/2) - $10.00
ADOS 3 (CoCo 3) - $20.00
Extended ADOS 3 - $30.00 (ADOS 3 req., RAM drives, support for 512K-2MB)
ADOS 3/Ext. Combo - $40.00

Pixel Blaster (OS-9)- $12.50: High speed graphics tools for OS-9 Level II. Easily speed up your game programming with this tool kit and subroutines!

Patch OS-9 - $7.50
Automated program installs most popular/ needed patches for OS-9 Level II. 512K and two 40T/DS (or larger) drives required. (*128K /35T users can install manually- state 35T.*)

Books:

NEW!! **Mastering OS-9 - $30.00**: This is the long awaited update of Paul Ward's "Start OS-9". New format is easier to read, has easier to follow tutorials,and updated information files. Comes complete with disk, which has a few added utilities.

Tandy's Little Wonder - $20.00
History,technical info, schematics, peripherals, upgrades, modifications, repairs, much more- all described in detail for all CoCo models! Vendors, clubs, BBSs also listed.

Quick Reference Guides
OS-9 Level II - $5.00 OS-9/68K - $8.00
These handy QRGs have the most needed info in a 5.5"x 8.5" desk-top size. Command syntax, error codes, special keys, etc.

CoCo Hardware:

DigiScan Video Digitizer - $125: Capture images from VCR, camcorder, or TV camera. No MPI required- uses joystick ports. CoCo Max3, Max 10, Color Max 3 compatible. Special order- allow 90 days for delivery. Send $75 deposit, remainder due before delivery.

Puppo Keyboard Adapters - $65: Use IBM PC/XT keyboards with your CoCo! Mounts in CoCo case with no soldering.

GLENSIDE COLOR COMPUTER CLUB PRESENTS:

THE FIFTH ANNUAL "LAST" CHICAGO COCOFEST!

Held at the Elgin Holiday Inn

(A Holidome Family Recreation Center)

Room Information:
For reservations call the Holiday Inn directly at 708-695-5000. The special 'fest rate is only $57 per night. A limited number of rooms will be held for the 'fest! These rooms will not be held after March 28.

Tickets:
Glenside Members: $5; all others $10. Now would be a good time to consider a membership! Receive the club newsletter and other benefits! Send ticket and/or membership fees to:

George Schneweiss, Treasurer
Glenside Color Computer Club
RR#2 Box 67
Forrest, IL 61741-9629

For More Info Contact:
Eddie Kuns: 708-820-3943
eddiekuns@delphi.com (e-mail)
Tony Podraza: 708-428-3576
708-428-0436 (club BBS; 8-N-1, 2400 baud)

Vendors: Please inquire! It is still not to late for us to squeeze you in!

The OS-9 User's Group, Inc.

Working to support OS-9 Users

Membership includes the Users Group newsletter, MOTD, with regular columns from the President, News and Rumors, and "Straight from the Horse's Mouth", about the use of OS-9 in Industrial, Scientific and Educational institutions.

Annual Membership Dues:

United States and Canada	Other Countries
25.00 US	30.00 US

The OS-9 Users Group, Inc.
6158 W. 63d St. Suite 109
Chicago, IL 60638
USA

Northern Xposure

'Quality Products from North of the Border'

OS-9 Level II Color Computer 3 Software

NitrOS-9 v1.20 Call or write for upgrade info or new purchase procedure. Requires Hitachi 6309 CPU	$29.95
Shanghai:OS-9 Introductory price Send manual or RomPak to prove ownership	$25.00
Thexder:OS-9 Send manual or RomPak to prove ownership	$29.95
Smash! Breakout-style arcade game	$29.95
Rusty Launch DECB/ECB programs from OS-9!	$20.00
Matt Thompsons SCSI System v2.2 'It flies!' 256/512 byte sectors, multipak support	$25.00

Disk Basic Software

Color Schematic Designer v3.0 New lower price	$30.00
Oblique Triad Software	*Write for catalogue*

Color Computer 3 Hardware

Hitachi 6309 CPU (normally 'C' model, may be 'B')	$15.00
SIMM Memory Upgrade Runs Cooler! 512k $44.95	0K $39.95
Sound Digitizing cable	$15.00

OS-9/68000 Software

OSTerm 68K v2.2 External transfer protocol support TTY/ANSI/VT100/K-Windows/Binary Emulation	$50.00
Upgrade from TasCOM (Send TasCOM manual please)	$30.00

7 Greenboro Cres
Ottawa, ON K1T 1W6
CANADA
(613)736-0329

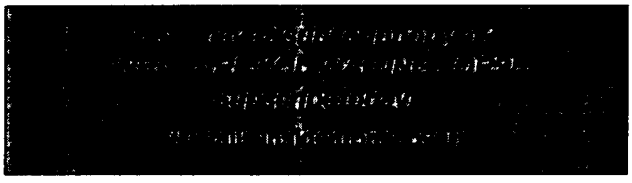
All prices in U.S. funds.
Check or MO only.
Prices include S&H

Internet mail: cmckay@northx.isis.org

Don't have a subscription to "microdisk"? Don't want to pay the high price for back issues? You can now get the complete Volume 1 of microdisk for $30 (plus $2.50 S&H)! That's an $18 savings over back issues and a $10 savings over the subscription price! Just write and tell us you want the entire volume 1.

All files will be on as few disks as possible, not separate disks for each issue. "microdisk" is not a stand-alone product, but a companion to this magazine. Subscriptions are $40 per year. Single issues are $6 each in US, $45/$6.50 in Canada. Overseas add $10 per year, $1 each for airmail.

ADVERTISER'S INDEX:



Wittman Computer Products

HARDWARE * SOFTWARE * CONSULTING

We Offer Gift Certificates

Software

Game Pcak - Sea Battle, Minefield, KnightsBridge, Othello & Yahtzee	$30	MM/1
	$25	CoCo
Variations of Solitaire - Pyramid, Klondike, Spider, Poker & Canfield	$30	MM/1
	$25	CoCo
Gold Runner 2000 - KWindows game	$35	MM/1
KChess - GNU Chess with KWindows GUI	$25	MM/1
GNU Chess Sources Disk	$ 5	OS-9/68000
CirCad - Circuitry CAD program with EPS output	$80	MM/1
DeskTamer v2.0 - a premier personal information manager	$65	MM/1
LaTerm / LaDial - GUI terminal package with script auto-dialing	$65	MM/1
LaPhone / LaFax - GUI voice, fax & address database manager	$65	MM/1
LaFax - stand-alone fax sending utility	$40	OS-9/68000
InfoXpress - automates the message gathering process to save the user on-line time & money		
	$75	OS-9/68000
	$55	OS-9/6809
X-10 Home Controller - program and monuitor your X-10 system	$40	MM/1
M6809 - utility to emulate the OS-9 Level II operating system	$65	OS-9/68000
TVP Point-Of-Sale - fully integrated multi-user accounting, inventory, ordering,		
⇒ and check-out (cash drawer) POS software all in one package	$CALL	OS-9/68000

Hardware

WCP306 single board computer with 16 bit PC/AT I/O Bus — 3 slot board $400

⇒ MC68306 CPU at 16.67 MHz, code compatible with 68000. — 5 slot board $425

⇒ Running OS-9 V3.0 including many utilities.

"The WCP306 is a good product to use. There are certainly opportunities in using it in the manufacturing environment."
quote by: "Sam T. C. Kwok of Motorola Semiconductors H. K. Ltd."

Point-of-Sale Equipment
Hard Drives & Floppy Disk Drives
Cases & Power Supplies
Data / Fax Modems
Keyboards, Memory & Mice

For a more detailed description, write or call us for a catalog.

We have moved to a larger location!

Wittman Computer Products
William L. Wittman, Jr.
39 South Lake Avaenue
Bergen, NY 14416
(716) 494-1506
e-mail: ww2150 @ acspr1.acs.brockport.edu
(acspr "number 1", "lowercase l" won't go through!)

For superior OS-9 performance, the

SYSTEM V

Provides a 68020 running at 25 MHz, up to 128 MBytes of 0 wait-state memory, SCSI and IDE interfaces, 4 serial and 2 parallel ports, 5 16-bit and 2 8-bit ISA slots and much more. The **SYSTEM V** builds on the design concepts proven in the **SYSTEM IV** providing maximum flexibility and inexpensive expandability. Optionally available at 33 MHz.

AN OS-9 FIRST - the MICROPROCESSOR is mounted on a daughter board which plugs onto the motherboard. This will permit inexpensive upgrades in the future when even greater performance is required.

G-WINDOWS benchmark performance index of a SYSTEM V running at 25 MHZ with a standard VGA board is 0.15 seconds faster than a 68030 running at 30 MHz with an ACRTC video board (85.90 seconds vs 86.05 seconds).

For less demanding requirements, the

SYSTEM IV

The perfect, low cost, high-quality and high performance OS-9 computer serving customers world-wide. Designed for and accepted by industry. Ideal low-cost work-station, development platform or just plain fun machine. Powerful, flexible and expandable inexpensively. Uses a 68000 microprocessor running at 16 MHz.

G-WINDOWS - a PROVEN WINNER

FOR OS-9

G-WINDOWS is now AVAILABLE for OS-9000 from *delmar co*

Available for the **SYSTEM IV** and **SYSTEM V** computers, the PT68K4 board from Peripheral Technology and the CD68X20 board from Computer Design Services. Resolutions from 640 x 480 x 256 to 1024 x 768 x 256. Support for multiple VGA cards running different processes, different portions of the same process or both.

Support for generic VGA boards and SUPER VGA boards including ET4000 (Tseng Labs), OTI067 (Oak), CT452 and CT453 (Chips and Technology), GENOA, WD90C11 (Paradise) and S3 (S3 Inc.) for modes from 640 x 480 to 1280 x 1024 depending on board.

Distributor of MICROWARE SYSTEMS CORPORATION Software

This ad was prepared and printed using QuickEd under OS-9.

delmar co

PO Box 78 - 5238 Summit Bridge Road - Middletown, DE 19709

302-378-2555 FAX 302-378-2556