NEW!

OS9 Underground -On-Disk

Programs from "OS9 Underground" Magazine will now be available on Disk!

This disk is published quarterly, the first one being available Mid-August. This disk will contain SRC and Binary code for programs published in the last three (3) issues. A Yearly Subscription (4 Disks, 1 every 3 months, aprox 12-16 progs/disk) are $35.00/year and Single Disks will be available for $10.00.

SPECIAL INTRODUCTORY OFFER

Get a 1 year subscription on disk for only $29.95 (4 disks/year). Single copy (also available in August) for Only $9.95

This Offer will expire August 30 1992 and will revert to the regular price of $35.00/year or $12.00/single disk after that date.

Name _____

Address _____

City _____

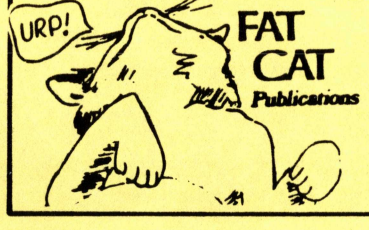
State/ Province _____ Zip/Postal Code _____

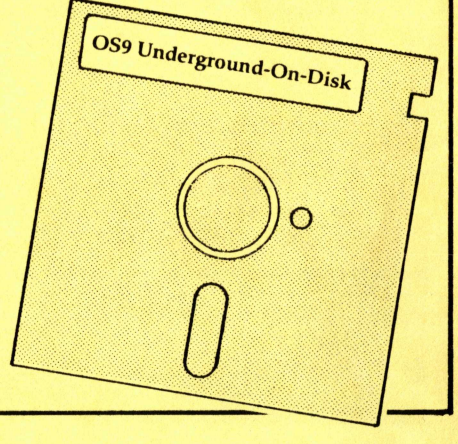
Phone _____

(Check or M.O. should be made out to AniMajik Productions)

Send your check or M.O. to:

"The OS9 Underground Magazine"
Fat Cat Publications
4650 Cahuenga Blvd.. Ste #7
Toluca Lake, Ca 91602

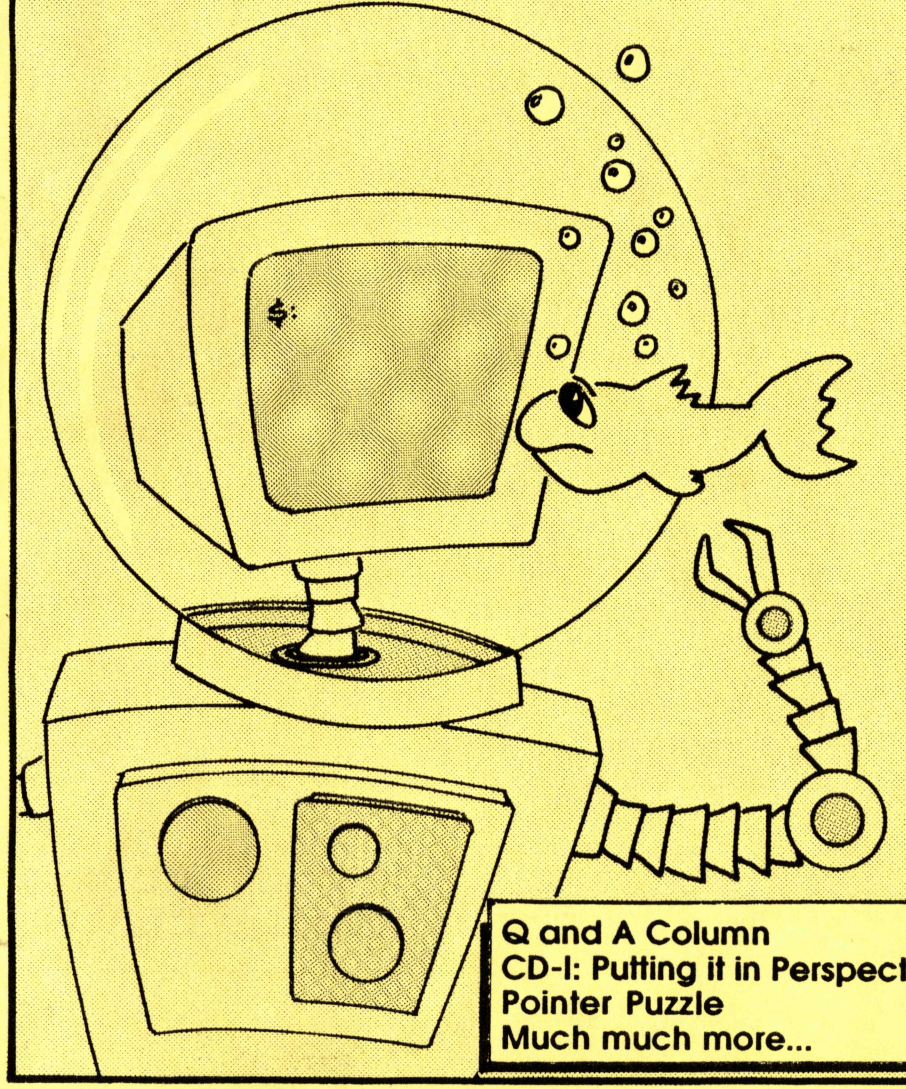




The "International" $3.00 OS9 Underground®

Magazine Dedicated to OS9/OSK Users Everywhere!

UNDERWATER ROBOTS NV RAM and OS-9



Q and A Column
CD-I: Putting it in Perspective
Pointer Puzzle
Much much more...

SYSTEM IV

The 68000 Computer serving customers world-wide

This high-quality, high performance 68000 computer was designed for and is accepted by industry. Perfect low-cost work-station, development platform or fun machine. Powerful, flexible and expandable inexpensively. Run MS-DOS software with the optional ALT-86 card. Supports up to 4 operating systems.

Prices start at $999.00 with Professional OS-9

FREE

G-WINDOWS DEMO

for SYSTEM IV and PT68K4/2 owners.

(Offer ends July 31, 1992)

Multi-tasking - processes continue running when windows are made inactive or are hibernating.

Windows may be re-sized, moved, overlaid, etc.

GUI to start processes by selecting an icon or, start processes from your custom menu or from the command line.

Copy and Paste between windows.

Adds command line editing, command history, and file name expansion.

Runs existing OSK software without modification.

Number of windows and processes limited only by your memory.

Includes GIF viewer.

Includes G-VIEW demo.

G-WINDOWS with DESKTOP	$199.00
G-WINDOWS Developer's Pak	$299.00
LIMITED TIME OFFER - both for	$399.00

(Offer ends July 31, 1992)

OS-9/68000 SOFTWARE

QUICK ED - Screen Editor and Text Formatter
VED ENHANCED - Text Editor
SCULPTOR - Development and Run-Time Systems
FLEXELINT V4.00 - The C Source Code Checker
WINDOWS - C Source Code Windowing Library
IMP - Intelligent Make Program
CALC-9 - Spreadsheet
VPRINT - Print Formatter (for VED)
M6809 - OS-9 6809 Emulator/Interpreter
DISASM_OS9 - OS-9/68K Dissassembler
PROFILE - User State Program Profiler
PAN UTILITIES

delmar co

Middletown Plaza - PO Box 78 - Middletown, DE 19709
302-378-2555 FAX 302-378-2556

New Stuff From...

SUB-ETHA SOFTWARE

"In Support of the CoCo"

CheckBook+09 by Joel Mathew Hegbberg
"Point 'n click" checking account manager with graphing! Fully mouse/joystick driven. Visualize your account with Circlr, Bar and Line graphs which can be exported for use with other software. Pop-up currency calculator with memory, "cut "n paste" editing sorting, printer support...plus more! (RS-DOS version reviewed October 1991 Rainbow, page 54.) Req: 512K CoCo3, OS9 Level 2, 80 Column Monitor $24.95

MiniBanners09 by Allen C. Huffman
Print Single or multiple line banners on ANY printer!
Supports graphi or non-graphic printers (like Daisy wheels, plotters, or even non-standard printers like the TP-10)! Independently sized text on each line (up to 16). Comes with over 30 fonts. Fully menu driven and easy to use. Now output to any device or file! (RSDOS version reviewed October 1991 Rainbow page 54.) Req: 512K CoCo3, OS9 Level 2 Any Printer $19.95

MAC to DMP Print Utility by Carl England
Dump MACintosh picture files direct from disk to your DMP printer! Now even a CoCo 1 or 2 (which cannot adequately view a MAC picture) can print them out. Comes with 6 sample pictures to get you started. Req: CoCo, OS9 Level 1 or 2, DMP 105/106/130 Printer 14.95

Worlds at War by Trevor Milne
A complete WAR GAME DESIGNER for OS-9!
NEW! War game fans unite! Creat multi-color icons for game maps and pieces. Define terrain (damage, supplies, shielding, etc.), artillery (attack, distance, units, fuel capacity,cargo payload, fire power, etc.) and more! Multi-screen playfield. Powerful graphics interface. Plays and looks great.
(For two players) Req: 512K CoCo3, OS9 Level 2 ..... 29.95

Document Printer by Carl England
Print booklets on your DMP/Epson printer!
NEW! Feed it text files and it prints sideways on a Tandy DMP or Epson printer. Fold the sheets and instant booklets. (OS-9 and RS-DOS versions included.)
Req: CoCo3, OS9 Level 2 or RS-DOS $24.95

Sub-Etha Software
P.O. Box 152442
Lufkin, Texas 75915
(409) 639-ETHA [3842]

Call or Write for Information!
Add $2.50 S&H and $4.50 C.O.D.
Texas residents add 8.25% tax.
"Don't Panic — We ship Fast!"

LETTERS - (Continued from page 5)

to those still interested in OS9 (the REAL operating system). However, I do have one minor gripe also. The letter spacing (kerning I think) should be increased (if possible) slightly so that during the printing process, they do not run into each other. It make the overall look of the page unsightly and in some cases difficult to read."
"Anyway, I think this first issue was a great effort and I hope to see it make a difference. Keep up the great work!"
-Corey Lee

• • • • •

"I really enjoyed the first issue of the OS9 Underground! For the most part the magazine's content were extrememely educational and in my opinion the writers put some work into some research with each article. Though we were under deadlines it does not appear that the magazine was haphazardly put together. I personally learned from Mike's hardware column and Leonard's and Andy's programming collumns. A special "thanks" to those three writers."
"The only sugestion that I have for future issues is that if possible can the line spaceing be increased so that there is more white-space per page? It would increase readablility by a great factor! Thanks Alan for a great magazine!"
-Jim Vestal

[Thanks again for the comments and critique. (See my above comment about line spacing) Hope this spacing is a bit easier on the eyes.]

• • • • •

TOO MANY COMMERCIALS

"I've carefully reviewed the article 'Copact Disc-Interactive Cogg which appears in the May/June, 1992 issue of 'The 68xxx Machines'. Taking the article as a whole, it is commercial in nature but written in the guise of an article. This is evidenced from the paragraphs beginning 'FHL's PLANS' on - which are blatantly commercial."
"I would hope that, in the future, you screen articles more carefully. If you do decide to publish articles of this type, I believe you should add a disclaimer to your readers cautioning them of the commercial nature of the article."
-Edward Gresick
DELMAR CO

[Editors Note: This is in response to Frank Hogg's CD-I article in the June issue of "68xxx Machines".]

• • • • •

Next Month...

Chicago CoCoFest Report!

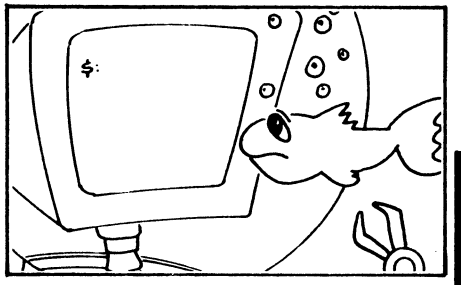
Advertiser's Index

The OS9 Underground®

Magazine Dedicated to OS9/OSK Users Everywhere!

July 1992 Vol 1, Issue 2

CONTENTS



Now Incorporates "68xxx Machines"!

Editor/Publisher:
Alan Sheltra (Zog)

Associate Editor:
Jim Sutemeier

Review Editor:
Jim Vestal

Technical Editor:
Paul Pollock

Contibuting Editors:
Ron Anderson
Rush Caley
Leonard Cassady
Andy DePue
Ed Gresick
Scott Griepentrog
J. Scott Kasten
Randy Krippner
Bob van der Poel

Art and Typsetting:
Alan Sheltra

Printing:
Pink's Printers

The OS9 Underground Magazine is published monthly by Fat Cat Publications, 4650 Cahuenga Boulevard Suite #7, Toluca Lake, CA 91602. The OS9 Underground and Logo types are registered trademarks. The OS9 Underground is $18.00/year shipped first class in the U.S., $23.00 in Canada and $27.00 overseas (airmail add $22.00/year). Article and program submissions to the OS9 Underground are always welcome and may be sent to the above address in electronic format or to our BBS (818) 761-4135 (8pm to 8am PST). Advertising rates are reasonable, please call or write for our rate card (818) 761-4135 (9am to 7pm PST).

MAIL ROOM

LETTERS TO THE EDITOR

ATTABOYS

Dear Editor:
"Well I just got the first issue recently and my reaction can only be summed up in 3 words; "I Love It!". This is easily the future of OS9 and I hope to see it carry us through the 90's§and I think we CAN last that long¶. I see it as Simple and To-The-Point yet still With an Attitude! Let's see some more!!!!!!"
-Chris Perrault
Bradford, MA.

●●●●●

Dear Editor:
"The Underground is fantastic. Strictly devoted to the OS9 that we are all familiar with...well written, nice print format, and the colorization of certain pages make it stand out quite nicely. I can see that a lot of thought and work went into this magazine, and you are to be thanked (and congratulated) on your efforts for the OS9 community. Just getting together some of these writers took a herculean effort." "It's nice to see Paul Pollock back in print again." "Thanks for a job well done."
-Jim Sutemeier
North Hills, Ca.

[Thank you Jim. I'm sure our Q&A man, Paul Pollock appreciates the kinds words too.]

●●●●●

Dear Editor:
"Totally, Totally, Totally Awsome!"
"... the Rainbow's in deep s___!"
-Lawrence Lasher
Tucson, Az

[Ahem.. er, thanks. Glad I don't own a shovel!]

Dear Editor:
"I'm really impressed. The layout, the pictures, evrything looks good."
-Terry Laraway
Bremerton, WA

Dear Editor:
Hi. I just got the OS9 Underground today. It looks really good! Excellent work Alan.
-Wes Gale
Vancouver, Canada

●●●●●

Dear Editor:
"Thanks for your reply and promptness. I mailed a check for a 1 year sub day-before-yesterday. Glad to hear about combining OS9UG and 68xxx. I too was contemplating subscribing to both but now I dont have to ... THANKS!"
-Ray Mayeux
Asheville, NC

●●●●●

Dear Editor:
"...hope the magazine will never die! BTW, have canceled my sub-sciption to the Rainbow Mag. Nothing in there anymore for an OS9 user... G'Day!"
-Dieter G. Rossmann
Lethbridge, Canada

●●●●●

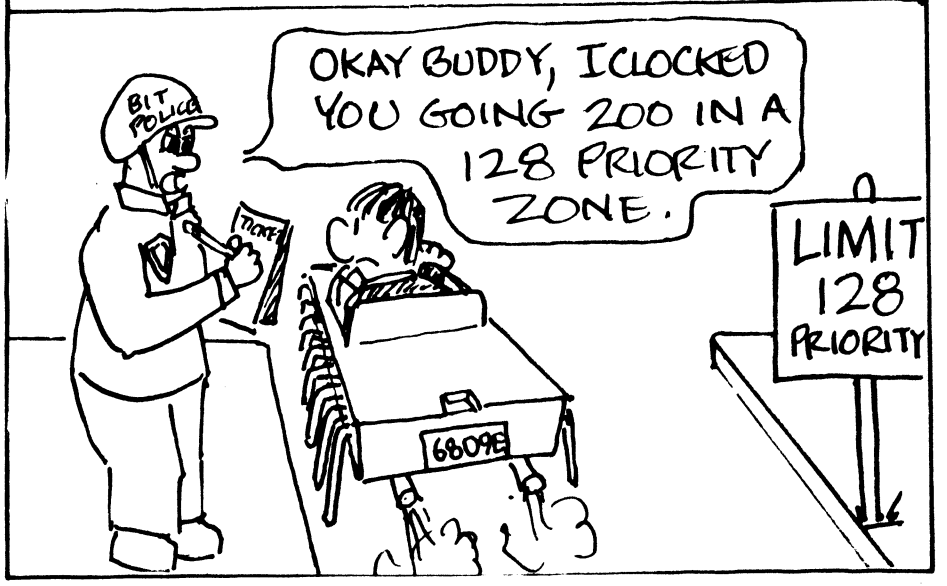
Dear Editor:
"I got my first issue this morning and I liked what I saw. I hope that they are successful. I have subscribed to almost every COCO magazine, (really liked Ted Paul's COCO Clipboard). I also recently renewed my subscription to Rainbow and it arrived yesterday shredded. I wish that they would either put it in an envelope or a plastic bag. I would send them a supply for my issues if I thought they would use them. With all best wishes."
-Br.Jeremy, CSJW (REVWCP)
Racine, WI

●●●●●

Dear Editor:
Just got the Premier Issue of OS9 Underground yesterday. For a first issue, I am duly impressed. I think this Mag. is going to be a real winner. Just the thing for an OS-9 Maniac like myself.
-Ken DeRoche
Tampa, FL

●●●●●

SHELL GAME by Alan Sheltra



C POINTER PUZZLE

```
char *pt[8]={"OS9"," UN","DER","GRO","UND"," MA","GAZ","INE"};\n");
printf("  char **p=pt;\n\n");
```

```
 1)  **p-13          =
 2)  *(*(pt+5)+2)    =
 3)  *(*(pt+4)+1)    =
 4)  *pt[2]          =
 5)  **pt-10         =
 6)  **p+3           =
 7)  (*(*pt+1))      =
 8)  *(*(pt+1)+2)    =
 9)  *(*(p+6)+1)     =
10)  **p+5           =
11)  (*(*(pt+2))-1)  =
12)  (*(*(p+7))-1)   =
```

Here's a little Pointer Puzzle for you C Gurus to solve...

Write your answer to each of 12 questions and fill in the boxes below to spell out the Mystery Word!

Good Luck!

1	2	3	4	5	6	7	8	9	10	11	12

Send in your answer to the OS9 Underground and get a free issue! (or your subscription increased by 1) Send a postcard or letter to :

"The OS9 Underground Magagzine"
4650 Cahuenga Blvd. Ste #7
Toluca Lake, CA. 91602

First five (5) correct entries will get a Free Issue (or subscription increased by 1). Must be in before August 15th, 1992.

* Contest not open to Underground Staff

THE RUMOR IS TRUE

Dear Editor:
"Hi Guys and Gals, the message about "68xxx" being taken over by "OS9UG" is correct. Of course this bring many questions. I guess it was an oversight That the questions weren't answered when the announcement was made. So what are the answers? WHY? Simply put, the size of the subscriber base. There are enough subscribers for one publication but not for two. Alan is hot and ready to go, produces a quality product and agreed to the deal. "68xxx" is not in trouble. The ink is black, though not a whole lot of it. The printer and post master have always been paid up front. Also of great importance, the re-subscription rate is very high. So, all is strong and "thumbs up". "OS9UG" will be infused with all this established strength. That makes them very strong coming out of the gate. They should do very well..."
"...I will not be going away. I hope to be contributing to "OS9UG" where and when I can. While not having any offical control, I at the mimimum, will be a $strong$ 68K advocate. I thank all that helped and supported THE 68xxx MACHINES. I had a good 16 months getting it off the ground and flying it. Now there are back issues to contend with and its time we all get behind THE OS9 UNDERGROUND. I wish all the best." -Jim DeStafeno
Wyoming, DE

[Jim is the editor of "68xxx Machines", which has now become part of this Magazine.]

● ● ● ● ●

READS LIKE A "BUFFER CAPTURE"

Dear Editor:
"OS9Underground reads like a 'buffer capture'. It just doesn't seem professional with all the (grins) and (!!!)s in the text. Those not connected with the online community (and there are quite a few as Dave Myers found out when he went from BBS-only to magazine advertising) will just not get it. So...just a suggestion...make the text more text-like. (Myself included...grin!)"
-Allen C. Huffman
Sub-Etha Software

[Some of the authors are still new to print media and most of their writing experience has been on BBSes and on-line services. So cut 'em a little slack, huh? :) Seriously, I will pass this on to the various authors Allen. I appreciate the comments.]

● ● ● ● ●

HARD ON THE EYES

Dear Editor:
"Uh, I only scanned an ish briefly at the fest...but I can't say I agree with you in your assessment that Underground is "nicely printed". The cover was nice and all, but the body text was painful to read, IMHO. As Alan is a graphic designer, I suspect (at least hope) that this will improve. But using bitmapped fonts, with excruciatingly tight leading, is not what I would consider "nicely printed" or "readable". All this strictly my opinion, of course. Potential advertisers DO look at production values (or lack thereof) when deciding whether or not to buy space in a new mag." "I will hastily add that this was only the first ish, and Alan was under a tight deadline to get it out for distribution at the show...and, as well, likely had some other time constraints due to the aftermath of occurrences near his homebase in Hollywood. Future issues should tell a bit more ;) -Dave"
-Dave Meyers

[Dave, I totally agree that the first issue was not perfect. The readablity of the type (line spacing) was terrible. I have adjusted the spacing, as you can see, in this issue. I wish I could blame this on the earthquakes and riots, but this one was my fault!]

● ● ● ● ●

"Well after having perused the premier issue of "The OS9 Underground," I'm quite impressed with it given the resources at hand. Content wise I think it will prove to be a great asset

(continued page 50)

Under it all...

I'm sure the first thing you may have noticed about this month's issue is the size. It's just about doubled since the last (premier) issue. That's for a good reason. "68xxx Machines" Magazine has now merged with the "OS9 UNDERGROUND"!

Former subscribers to 68xxx Machines will not get left out in the cold, your subscription will now continue with this greatly expanded issue with all the articles and columns you were used to. Subscribers to The OS9 Underground will now get many more articles to read! So no one looses here. Jim DeStafeno, editor of 68xxx Machines pledges his support and has helped in making the transition a smooth one!

I'd like to welcome the authors and writers of 68xxx Machines to The Underground and want to make them feel right at home! Joining us this month are Ron Anderson with his "Beginner's Corner" assembly language tutorials, Randy Krippner's "BUI" Basic09 User Interface Series, Scott Kasten's "TECH CORNER" and a Fractal Print Program, Rush Caley's "Downtime" and 'toungue in cheek' discussion of the English Language and Bob van der Poel's continuation of "USING THE TERMCAP LIBRARY". Welcome guys!

Also in this month's issue our feature story by Scott Griepentrog, "UNDERWATER ROBOTS, NVRAM and OS9" will literally take OS9 to great depths.

Ed Gresick reports on CDI and how it relates to OS-9. Jim Sutemeier starts his series on "HASHING" and our "Q and A Man", Paul Pollock, actually gets to answer some of those hardware questions you've sent in.

Jim Vestal, Leonard Cassady and Mike Ortloff continue their fine tutorials. Allen Huffman gives us a first-hand report of the Chicago Coco Fest.

- - -

I must say subscription response to the OS9 Underground has been nothing short of phenomenal! I want to thank everyone for all the kind words and support I've received.

-Alan Sheltra

program or system "crash". Sometimes the programmer may be lucky and the pointer will point to an area of memory not in use, however, even in this case, there is no "null" character to signify the end of the character array. The result of operations using the pointer will be unpredictable.

As the discussion of pointers continues, we'll explore in greater detail how to efficiently use pointers for character array operations.

BASIC

Often, BASIC programmers complain about the lack of character string array functions in C Language. It is sometimes necessary to create specialized functions to emulate operations builtin other languages.

Next issue, we'll continue the discussion of pointers. Until then, examine the following examples which make use of pointer operations. There are no standard C functions to perform these operations, so it is necessary to create our own. It is important to note that similar functions in BASIC return a value that is assigned to a character array.

```
BASIC Format: A$ = MID$(B$,start,count)

     10 B$ = "OS9 Underground"
     20 A$ = MID$(B$,7,3)
     30 PRINT A$
     40 REM Displays "der"
```

The C version returns no value, instead, it sends a copy of the substring to the memory location provided by the argument.

```
C Format: midstr(a,b,start,count)

main
{
     char a[18], b[18];

     strcpy(b, "OS9 Underground");
     midstr(a,b,7,3);

     /* displays "der" */
     printf("%s\n", a);
}
```

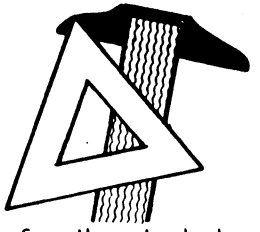
```
/* function midstr() */
midstr(a,b,start,count)
char *a, *b;
int start, count;
{
     --start;
     count += start;

     while(start < count &&
          start <= strlen(b))
     *a++ = *(b + start++);
     *a = '\0';
}
```

Both examples extract the character array "der" and assign it to 'A$' and 'a' respectively. The next example replaces a substring within the string.

```
BASIC Format: MID$(A$,start,count) = B$

     10 B$= "der"
     20 A$ = "OS9 Un***ground"
     30 MID$(A$,7,3) = B$
     40 PRINT A$
     50 REM Displays "OS9 Underground"


C Format: midfunc(a,b,start,count)

main()
{
     char a[18], b[18];

     strcpy(b, "der");
     strcpy(a, "OS9 Un***ground");
     midfunc(a,b,7,3);

    /* displays "OS9 Underground */
     printf("%s\n", a);
}

/* midfunc function */
midfunc(a,b,start,count)
char *a, *b;
int start, count;
{
     int i;
     i = strlen(a) - 1;
     count += start;
     --start;

     while(start <= count &&
          start <= i && *b != '\0')
     *(a + start++) = *b++;
}
```

<EOF>

SOFTWARE ENGINEERING



By Leonard Cassady

The C Language is used by most of today's major software developers for writting applications. C is portable, fast, efficient, and flexible. The authors of C wisely kept the language small by providing a limited number of functions. Building on this set, many specialized functions to perform more elaborate and hardware dependent operations are possible.

Pointer operations in the C Language generally give the beginning programmer the most difficulty. This is especially true of BASIC programmers. Although pointers are used in BASIC, they are usually hidden from the programmer. The C language allows pointers to be specifically programmed and used.

A pointer is a special type of variable that "points" to a specific place in memory. The pointer points to the location in memory where a value is stored. It is NOT directly equal to any defined value.

The following example shows how a 'character' pointer is declared and used.

```
main()
{
     char *ptr;

     ptr = "OS9 Underground";
     printf("%s\n", ptr);
}
```

First, the variable "ptr" is declared as a 'char' pointer. The unary operator, represented by an asterisk (*), specifies a pointer as opposed a standard character variable.

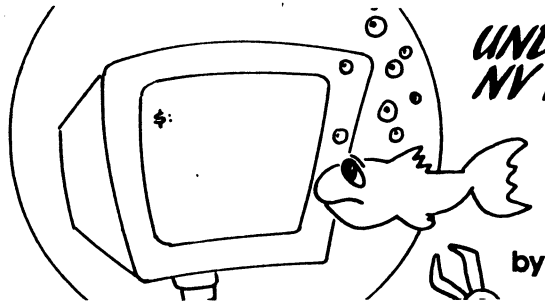
When a pointer is declared, it points to random location in memory where the actual memory location of the character array is stored. It is NOT equal to the character array, but contains the memory address where the computer has stored the constant character array.

A pointer differs from the standard variable in that it can point to something stored in memory, or to a memory location that has been set aside for future storage. Let's examine this in more detail. Assume the constant "OS9 Underground" is stored in RAM memory at location $1000.

RAM memory	stored char
$1000	O
$1001	S
$1002	9
$1003	space
$1004	U
$1005	n
$1006	d
$1007	e
$1008	r
$1009	g
$100A	r
$100B	o
$100C	u
$100D	n
$100E	d
$100F	\0

There is one additional character, stored at $100F. This is the 'null' character (½0), which signifies the end of the character array. Let's say that the RAM memory location of 'ptr' is $2050. The RAM memory address $1000 is the value placed in memory location $2050 when 'ptr' is assigned to the constant array "OS9 Underground".

Character arrays and character pointers must be thought of differently. The pointer has no data storage space set aside and the character array is a variable with a programmed amount of space set aside for storage. A better way to think of it is that unassigned pointers may point anywhere in memory, even a location already is use. Improper use of pointers could cause memory locations already in use to be overwritten, usually resulting in a



UNDERWATER ROBOTS NV RAM and OS-9

by Scott t. Griepentrog

After a long string of very disappointing events, including numerous contract jobs that fell through and the untimely un-arrival of a certain (now un-) magazine last October, I am sad to admit that I was very much burnt out on OS9. There was quite a number of months there that I didn't do much at all with either OS9 machine. Of course, I was dead broke too, and I had not as yet made a penny off anything I'd done with OS9. But in what must have been my first lucky break of the whole last year, I called an old friend and managed to pick up a job working on Unix machines at the local campus, I.U.P.U.I. (Indiana U & Purdue U @ Indy), pronounced UU-ee-PUU-ee. And after working there for a few months one of the guys pointed me towards a local company that needed some work done on MS windows.

Well, I wasn't really all that familiar with MS windows programming then, but went for an interview anyway. During which I basically rattled off a long list of all the little things I'd done here and there over the past ten years - and then listed all the machine languages I new, all the higher level languages I new, and of course all the operating systems I new - like oh, MS dos, uh, a little unix, and, oh yah, OS9 too.

Much to my surprise the person running the little software shop and the two programmers helping interviewing me lit up like a Christmas tree upon mention of OS9. So I took advantage of their reaction to further detail my involvement with the only operating system that (as it would seem) only 5% of the world is even conscious of. I mentioned, in as much of an offhanded way as I could muster (and trying to keep my usual poker face), that I not only had published a magazine dedicated to OS9, but that I was also considered by some to be sort-of a guru when it came to that operating system.

Turns out that one of the programmers was working on a particular OS9 application at the time, and not having much luck with it at all (his education in computers being solely intel-ish). In order to get the project moving, they were going to dedicate him to the OS9 work (to his considerable dismay) and bring someone else in to replace him on MS windows. So in my second lucky stroke of the year, I jumped into their OS9 project and not only got the base code functioning in a few days, but have been improving upon it since.

The product they sell is actually rather interesting. The concept goes something like this: If you have some complex machine or process that you want to control via computer, you take an OS9 machine (or an MS dos machine, or a micro-controller) and connect up digital and analog inputs and outputs to whatever needs sensing or controlling, and then you sit down and write a program to run it. And then you test it and re-write it over and over until you think maybe you've covered every situation - but of course there's always some new problem that crops up or some new control or sensor you have to add to make the machine really work right. So it becomes a real (bear | bore).

This particular company, called Event Technologies Inc. (ETI for short), came up with an easier way to create such programs (especially if halfway into the project you realize you've got to switch to a different machine or cpu in order to get the feature you need). They call it GELLO, which means something to the extent of 'Graphically Enhanced Ladder Logic

and worst qualities or things they don't like about themselves on the other. After ten minutes, collect all the papers.

What you will find is that most people will have a much larger list of "bad" qualities, and very few items, if any, listed in the "good" column. You also will notice that there is an extreme clarity and specificity about the description of negative qualities, while often the positive statements are inscribed in bland generalities.

This illustration points out a very unique quality of our language. We are more imaginative in our words of negative connotation than we are with those of positive connotation. I wonder if this is a phenomenon unique to our culture or if this disparity occurs in other languages as well. Is it ingrained in human nature?

I'm interested in what you think. Try paying close attention to what you hear and read. Notice things such as the fact that "good" and "nice" don't really convey anything about a person. Notice that English has only one word for "love" while other languages have many. See what you find. Perhaps Jim could give a free issue of the magazine to the person who can relate a true life case where imagination and power of thought are transmitted in a word, phrase or sentence that says something truly positive. -RC-

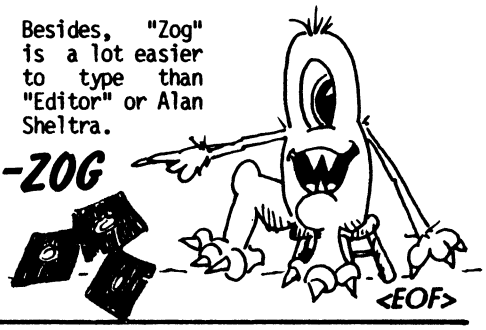
[Editor's Note: Any takers? I'll honor that challenge. Send your responses to Rush Caley, c/o the OS9 Underground and get a Free issue (or your sub "Bumped-up" by 1 issue). -Zog]

<EOF>

Who is this "monster" character we see popping up all over the pages of this magazine? What's he doing here anyway? Is he a just a one-eyed "Mr. Potato Head"? (Yes, I CAN spell "POTATO"!).

No, this one-eyed, green (yes, he's green) monster is named "ZOG" and is just the handle of your Editor. So don't be suprised if you see the name "ZOG" and "Editor" used interchangably here in future issues.

Besides, "Zog" is a lot easier to type than "Editor" or Alan Sheltra.



<EOF>

DOWNTIME

by Rush Caley

I have always been fascinated by words and language. They encase the power of our thoughts and transport them on the wind to those around us. I'm especially intrigued by the words and phrases we use to indicate that we are in a position of power or that we are somehow "better" than another. Sometimes these are clever put-downs or thinly veiled threats; other times the words just sound funny and their meaning is questionable. These words do not normally have a specific target. For example, they are not aimed at an unfortunate physical characteristic of an individual, like "Tin Grin" or "Lardbucket." Nor are they the sweeping type of character condemnation such as the famous "Nattering Nabobs of Negativism" as uttered by one of our illustrious former Vice Presidents. The words I love are, for the most part, what I term "generic derogatories."

Last week, I heard someone use a term I had not heard in years. As it turns out, this word has survived the ravages of time and become commonplace again. The word is pipsqueak. "Shut up you little pipsqueak" was a phrase I became quite used to when I was growing up. The dictionary defines it as a derogatory term denoting a small, ineffectual, person of little or no consequence. It's a funny sounding word for such a forceful thing to say about a person. Anyway, I was so thrilled to hear the word again, it prompted me to recall some of the more popular words and phrases from my formative years that I think can still be put to good use. Some of these require a minimum context, but most are self-explanatory.

I'M GOING TO CLEAN YOUR CLOCK FATHEAD WIPE THAT SMILE OFF YOUR FACE PINHEAD TAKE YOUR MEDICINE LIKE A MAN NUMBSKULL I'LL GIVE YOU SOMETHING TO CRY ABOUT SELFISH LITTLE BRAT DON'T MAKE ME HAVE TO COME IN HERE AGAIN GOURDHEAD TAKE A POWDER CLUMSY LUMMOX LAZY GOOD-FOR-NOTHING BUM GEEK MORON SPASMO TWERP JUDAS PRIEST APESHAPE SNOT DOPE

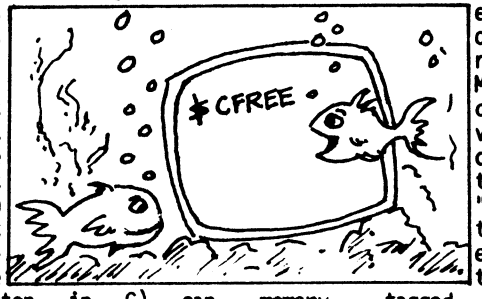
Try forming a few sample sentences and see if it doesn't make you smile. For example, a parent or other such adversary was always most formidable when they combined many of these terms into one long harangue. "Listen, you mindless little twerp, you'd better wipe that snotty smile off your face or I'm going to clean your clock! Judas Priest! Can't you quit acting like a selfish little brat for one minute? Now don't make me have to come in here again or by God I'll give you something to cry about....."

These types of words, which might have been relegated to the status of anachronism, have increasing value in todays word market. Why? Because the minions of "political correctness" have made it nearly impossible to speak our minds about anything or anybody. For the slightest difference of opinion one might express, the punishment migh trun the spectrum from "a funny look" to a lawsuit or societal ostracism.

The days of use for terminology like "Com-Symp-Radic-Lib", "Feminazi", "Homo", and the like, are numbered at best. But use of generic derogatories such as I have decribed is the best protection against the frenzied accusations from the politically correct. If you refer to anyone, no matter what their special little cause, as "pipsqueak", "jerk", or "twerp", they may not like it. But be assured, there's no cause for lawsuit, or arrest.

I also wish to examine the other side of this particular fascination I have with language. That is, the positive side. Here's a little exercise you can try. Most often its used by teachers with their young students; but adults will produce similar results. Have a group of people take a blank sheet of paper and draw a vertical line down the middle of the page. At the top of the left side of the page, write "My Good Qualities." On the right side, write "My Worst Qualities." Instruct the participants to list good qualities and things they like about themselves on the one side,

Objects', or something like that. Anyways, the gist of GELLO goes like this: take your average PC running MS windows and connect it to your target cpu (the one that's actually going to run your machine) with a serial line (or ethernet, or other connection). Now run the GELLO program on the PC, put the switch symbol in one box on the screen, put the relay symbol in another box, connect the two (this is all done with the mouse), then download the program you just wrote into the target cpu (in my case, OS9) running an interpreting engine (which is what I work on). Toggle the switch on your i/o and the relay toggles with it. Of course, a full set of binary gates, analog math, and floating point functions are implemented. The language in which you write programs is really a set of interlinked data records that define each object and which object's output each of it's inputs comes from. The graphical interface makes it easy for an expert on whatever machine or process you want to control to understand (without learning programming), and the interpreter (written in C) can be-ported to just about any processor, allowing the same GELLO program to work on almost anything.



They had working engines (the GELLO interpreter) for DOS and certain micro-controllers, but in order to further tap into the industrial controller market (as well as to satisfy a particular customer) they needed a working port to OSK. This particular customer wanted two such engines on OSK (specifically GESPAC SBS-6A, a 16 Mhz 68000 with 256K of non-volatile ram), having matching inputs and outputs on each and telemetry between over a single serial channel. Application: underwater robot. Whatever the operator switches up top is relayed down to the robot, and whatever is sensed down below is relayed back up to the operator. Plus, the capability to write some intelligence into the robot on the fly without even having to bring it up. Pretty neat huh? Well, so I thought too until they threw a couple more requirements at me.

First, the software had to be put in rom - operating system, engine, possibly even the GELLO program itself. Okay, I thought, no problem. OS9 itself can be easily put into rom, and any program that can be run normally (as long as it doesn't want to access any files if you have no RBF, which is the case) can too. And as long as I store the GELLO program as a standard OS9 module, that too can be put in rom (whenever they decide to stop changing it). Okay, one down, one to go.

The second requirement was that the GELLO program could be saved in non-volatile ram, so that in case of power failure (and before it was burned into rom) the robot would at least still function when the power came back on. This one shouldn't be a problem either - in going over OS9 V2.4 I had noticed that Microware added a definition for non-volatile ram in the colored memory table. In the file "/dd/defs/sysglob.a" there is a B NVRAM entry that specifies that a section of memory tagged as such will be "searched for modules" upon startup. Great, I thought. All I have to do is create a colored memory type for the NVRAM on the machine, save the GELLO program as a data module into that type, and after reboot link to it again. Piece of cake. Or so I thought.

So I wrote first a program called cfree in order to not only identify what colored memory sections where defined but also how much memory was available in each (see program #1). Cfree works by requesting the free memory list from OS9 the same way that mfree does - but then looks up each section of memory in the colored memory table (defined in the INIT module) and displays the description string and type code that it belongs to. I also wrote a variant on the load program that uses the new colored memory load call called (what else) cload (see program #2). By supplying a type code

as an option, it will load a module into that section of colored memory. Now I had the necessary tools to make a test of the NVRAM option (one should always test any such feature before writing it into a program first and then discovering you just wasted a day or two).

I created a colored memory type 127 ($7F) that covered the battery backed up static ram on the cpu board, and rebooted with that new init module. Then I issued the command 'cload -127 cfree' and hit reset. OS9 came back up, and I typed cfree. Not found it said. Hmm. I know it loaded it. So I tried again. I cload'd cfree again, but this time did a 'mdir -e' and wrote down the address of the module before rebooting. I then started up debug and told it to dump memory at the address the cfree module was at - and sure enough, it was there. Valid header, program, even the CRC was correct. So why didn't OS9 find the module? I tried switching the definition of that colored section from NVRAM to ROM (which OS9 also searches for modules). After reboot, I entered cfree again - and it ran!. Sure enough, section 127 had 0 bytes free, but the module was found. So I tried setting both the NVRAM and the ROM bits, but that didn't help either. This was getting mighty curious. Did I have an older version of the kernel by mistake? Maybe there was a mistake in the definition file? I know it's got to be there somewhere, 'cuz they defined it!

After years and years of working with OS9 and generally sticking to the rules, but once in a while using a command that, while it wasn't even mentioned in the manual was nevertheless defined in some file in /dd/DEFS, and how and why and when to use it was obvious (as was why it wasn't in the manual), but you could do so, it worked, and even more importantly, it made sense - it never occurred to me (and I still didn't believe it right away after I found out), that Microware would do such a naughty thing as define something and NOT IMPLIMENT IT!?!?

But as with the best of all such fairly tales, this one two ends with a happy ending. Being the ever vigilant do-code-er that I am (well, actually, I'm just darn stubborn), I spent the entire next week (and with only one helpful hint from the big M.W. I might add) coming up with a solution. And it works too! But we're out of time (and space) for now, so tune in again next month - same batty magazine, same batty author - for the answer. - StG
[Scott Griepentrog can reached for comment c/o this magazine or through the StG Net as SysOp@StG.]

```
*** PROGRAM #1 - cfree.c ***

/*
 * colored memory free list
 *
*/

char *copyrite="Copyright (c) 1992 by StG Computers inc.";

#include <stdio.h>
#include <module.h>

#define ERR (-1)

char *malloc();
mod_config *modlink();

extern int errno;

mod_config *psInit;          /* init module */

typedef unsigned char B;
typedef unsigned short W;
typedef unsigned long O;
```



```
 move.l #$FFFFFFFF,4(a1)
* lea 8(a1),a5
 sub.l #8,a5
 bra.s restor
* clean up stack and normalize
clean move.l (a5),(a1)
 move.l 4(a5),4(a1)
cleanl sub.l #8,a5
 bsr normal
fixsin move.b sign(pc),d7
 or.b d7,(a1)
```

```
move.b exp(pc),7(a1)
rts
```

Well, that's about it. You can see that the main loop got a lot smaller because we have bigger and better operations available when we do arithmetic using data registers. Also the operations are faster than when we use address registers for register indirect addressing into memory. -RA-

<EOF>

(Continued from Page 14)

```
*** PROGRAM #2 - cload.c ***

/*
 * colored memory load
*/

char *copyrite="Copyright (c) 1992 by StG Computers inc.";

#include <stdio.h>
#define ERR (-1)
extern int errno;

main(argc,argv)
int argc;
char **argv;
{
    int color=0;     /* default color is any */
    int attr=5;      /* default load is exec dir */

    while (*++argv)
    {
        if (**argv=='-')
        {
            if (*++*argv=='?')
            {
                printf("use: cload [-color] [-d] (module) ...\n");
                printf("\n%s\n",copyrite);
                exit(1);
            }
            if (**argv=='d' || **argv=='D')
            {
                attr=1;      /* load from data dir */
            }

            color=atoi(*argv);

            if (color<0 | color>0xFFFF)
            {
                printf("color value out of range\n");
                exit(1);
            }
        }
        else
        {
            if (modcload(*argv,attr,color)==ERR)
                exit(errno);
        }
    }
}
```

<EOF>

decimal numbers to and from BCD is much easier and faster than converting them to and from binary representations. If you are running calculation intensive programs, you soon realize that most of the calculation goes on without actually outputting any data. I/O applies only to a small fraction of the calculations and therefore doesn't have a large effect on the execution time. Generally programs with a lot of I/O don't have much calculation going on, though there are exceptions.

In these days when supercomputers are rated in terms of "megaflops" (millions of floating point operations per second), these numbers are pretty tame. Our fast multiply is about 5000 flops, and the add is about 6500. In terms of megaflops that is 0.005 and 0.0065 respectively. The divide now hits 0.0015 megaflops. Still, for a single user personal computer that is probably fast enough for most problems we want to solve.

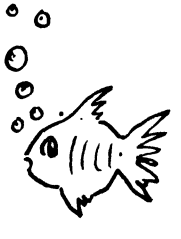
I hate to admit it, but an 80386 with an 80387 co-processor can easily beat these results by a very large factor. As noted above, these tests were run on a 10 Mhz machine. If I had been using a 16 or 20 Mhz 68000 machine the times would reduce by factors of 0.62 and 0.5 respectively; to say nothing of a 68888 co-processor for the 68000 or 020.

The improved FPDIV routine follows. The only changes are in the setting up of the repeat loop, the use of registers within the loop and the "unloading" of the registers after the loop.

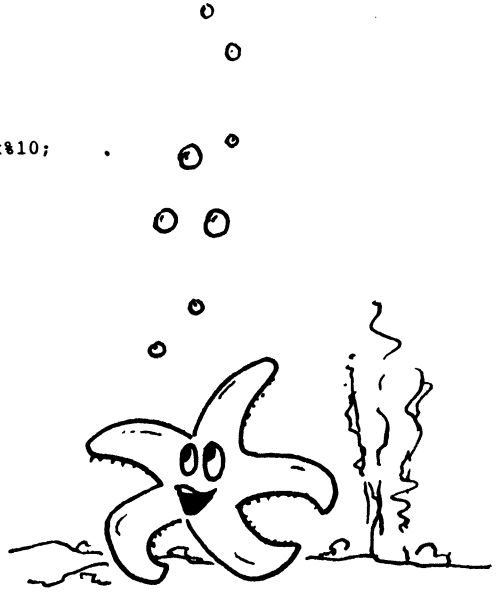
```
FPDIV equ *
 movem.l d0-d7/a2,-(a7)
* clear result area at (a5)
 clr.l (a5)
 clr.l 4(a5)
* test for zero
 lea -8(a5),a1
 bsr tstzer
 beq dzero denominator is zero
 sub.l #8,a1
 bsr tstzer
 beq nzero numerator is zero
* subtract exponents for result
 move.b 7(a1),d7
 sub.b 15(a1),d7
 add.b #$80,d7
 lea exp(pc),a2
 move.b d7,(a2)
* sign of the result
 move.b (a1),d7
 and.b #$80,d7
 lea sign(pc),a2
 move.b d7,(a2)
 move.b 8(a1),d7
 and.b #$80,d7
 eor.b d7,(a2)
 move.b #$7f,d7
 and.b d7,(a1)  clear sign bits of args
 and.b d7,8(a1)
* clear exponent bytes for the divide
 clr.b 7(a1)
 clr.b 15(a1)
* now we are ready to divide
* divide routine - first set up
* calculation loop
 move.b #55,d1 loop count
 move.l (a1),d7   num hi
 move.l 4(a1),d6  num lo
 move.l 8(a1),d5  den hi
 move.l 12(a1),d4  den lo
 clr.l d3 quot hi
 clr.l d2 quot lo
 move.l a5,a1 point at result area

* this is the main divide loop executed
* 55 times
div1 cmp.l d5,d7
 bcs.s div3
 bne.s div20
 cmp.l d4,d6
 bcs.s div3 can't subtract
div20 sub.l  d4,d6 result in d6
 subx.l d5,d7
 add.w #$100,d2
div3 lsr.l #1,d5
 roxr.l #1,d4 shift denom
 lsl.l #1,d2
 roxl.l #1,d3
 sub.b #1,d1
 bne div1

* end of loop, now put result on stack
 lea -16(a5),a1
 move.l d3,(a1)
 move.l d2,4(a1)
 bsr.s clean1  clean up
restor movem.l (a7)+,d0-d7/a2
 rts

* numerator zero, return zero
nzero sub.l #8,a5
 bra.s restor

* denom zero return largest number
dzero lea -16(a5),a1
* return large pos number
 move.l #$7FFFFFFF,(a1)
```



```
struct sCMem
{
    W wType;     /* color type code */
    W wPri;      /* usage priority */
    W wAttr;     /* attribute bits (...|SHARE|NVRAM|ROM|PARITY|USER) */
    W wSize;     /* search block size */
    O oLow;      /* search start */
    O oHigh;     /* search end */
    W wDesc;     /* offset to description */
    W wrsvd1;
    W wTran;     /* address translation */
    W wrsvd2;
    W wrsvd3;
    W wrsvd4;
    W wrsvd5;
    W wrsvd6;
} *psCMem;

B bFree[0x800]; /* found free for colored entry */

struct sMem
{
    O oAddr;
    O oSize;
} *psMem;

O oMin;      /* minimum allocation size */
O oFrag;     /* number of fragments */
O oRam;      /* total ram at startup */
O oFree;     /* current free ram */
O oCFree;    /* total free in colored type */
O oCSeg;     /* largest segment in type */
```



```
int
getmem(ptr,size)
char *ptr;
int size;
{
#asm
    move.l (sp),a0
    move.l 4(sp),d1
    moveq.l #0,d0
    os9 F$GBlkMp
    move.l d0,oMin(a6)
    move.l d1,oFrag(a6)
    move.l d2,oRam(a6)
    move.l d3,oFree(a6)
    move.l d1,d0
#endasm
}

char *
commas(x)
long x;
{
    static char tmp[32];
    int d=16;

    while (d)
    {
        if (d&3)
        {
            if (x || d==15)
            {
                tmp[--d]='0'+x%10;
                x/=10;
            }
            else
                tmp[--d]=' ';
        }
        else
        {
            if (d==16)
                tmp[--d]=0;
            else
            if (x)
                tmp[--d]=',';
            else
                tmp[--d]=' ';
        }
    }
    return(tmp);
}

disp(ps)
struct sCMem *ps;
{
    printf("\n%5d %s (",ps->wType,(B*)psInit+ps->wDesc);
    if (ps->wAttr&0x0001) printf("USER,");
    if (ps->wAttr&0x0002) printf("PARITY,");
    if (ps->wAttr&0x0004) printf("ROM,");
    if (ps->wAttr&0x0008) printf("NVRAM,");
    if (ps->wAttr&0x0010) printf("SHARED,");
    if (ps->wAttr&0xFFE0) printf("Othr=%04X,",
        ps->wAttr&0xFFE0);
    printf("Pri=%u)\n",ps->wPri);
}

#define dash(x) while (**++x=='-') while (*++*x) switch(tolower(**x))
```



appear to be the place to start.

I had used register indirect addressing with offset for the operations with the dividend, divisor, and quotient. After a little thought I decided to save all the data registers on entry and they would be available for the divide operation. I used pairs D7,D6 for the dividend, D5,D4 for the divisor and D3,D2 for the quotient or result. There is some overhead in moving the arguments to the registers and clearing the result registers, but those are all done outside the loop. Within the loop the operations are much faster.

The rotate operations can be done on long sized chunks rather than word sized, so there are fewer operations. At the end we simply store D3.D2 at the result location and proceed with the cleanup.

After a little bit of debug, I found the time for the FPDIV again with about half ones and half zeros in the result (i.e. 27 subtractions out of a possible 55), the time was now 660 microseconds, more than three times faster than the older approach.

I'll list the new one here. Realize that now we have to save all the data registers for the other two operations too, so we can use a common clean up routine. I will also look at the multiply routine with an eye toward using registers there. I doubt there would be much improvement in the add subtract routines since there is no loop and little repetitive operation. I have doubts about pushing two registers and loading them, shifting and storing would make ASLF and ASRF any faster, but I'll try that too.

As you can see, there are many ways to code things in assembler and first tries can usually be improved upon rather easily unless you are a whiz at the particular processor and have memorized which operations are faster. The general approach of fixing the worst operation (in terms of time) first and then going after smaller gains usually works well. Attacking loops that repeat a number of times is usually very fruitful too. The new FPDIV is better than 0.0015 flops, a little more than three times slower than FPMUL now.

As you can see, the binary package is generally about six times faster than the BCD in BASIC. I am a bit surprised that the times are not a lot worse for the BCD arithmetic. Of course I hasten to point out that the BCD is 13 digits and the assembler binary version is almost 16. Doubling the number of digits requires about four times as much processing, so correcting for the differences in resolution would make the numbers farther apart. Of course I should add that if you are running a calculation intensive program a speed gain of six times is nothing to sneeze at.

At this point I got interested in how long the empty loop of the assembler program would take. I had used D3 as a counter, initialized to 1000000 and decremented by a sub.l #1,d3 instruction. When using ASMK, that automatically becomes an SUBQ.L instruction. The time for 1,000,000 times through the loop was 1.9 seconds, making the time for the loop counting just 0.19 seconds for the multiply and add tests, and 0.019 seconds for the divide test. That is, the time for ADDQ.L was 1.9 microseconds.

I got to wondering how much faster the ADDQ instruction is than the normal ADD.L instruction. ASM allows me to specify which instruction I want to use so I assembled LOOPTEST with that using the long instruction. The time was 2.74 seconds for 1,000,000 passes, or 2.74 microseconds each time through the loop. Clearly the loop counter didn't add much overhead to our numbers for the binary math tests.

Then I got to thinking about all the pointer bumping that is done in the math routines so I changed a lot of the lea -8(a5),a1 types of instructions to add.l #8,a1 and sub.l #8,a1. Again, ASMK generates the code for the addq instruction.

I thought I might notice some improvement, particularly in the divide routine where I thought I could take most of a microsecond out of the 55 times through loop in several places. My timing tests after the changes were within measurement accuracy of the tests before the changes. I guess the improvement got lost in the overall execution time. Perhaps with a more accurate way of timing, I would have seen a difference, but not a large one.

You might ask if these comparisons are really fair. After all, converting

triplicate the code, we'll tack it on the FPDIV and branch here from the other routines. Alternately we could include CLEAN as a separate part of the code and branch there from each of the operations.

When I had written this column to this point, I decided to give FPDIV a more complete test. I found that it didn't work for a zero argument. 12 / 0 worked okay, producing a very large number, but whenever the result should have been zero the program would simply hang up. I traced the problem to one of the two routines we are going to talk about next time, the one that converts our floating point numbers to ASCII so we can print them out. It seems that my BINASC routine got stuck processing zero forever. I added a test for zero and returned 0.000000000000000. Now it works fine though my little calculator program that I have been using to test it still goes west on an error. I'll eventually figure out how to make it idiot proof.

Meanwhile I am glad I didn't try to discuss the conversion routines this time. Maybe in a while I'll get them working flawlessly so we can talk about them.

BINASC is a little obscure. It uses an algorithm that is not obvious. I've been looking for a better (shorter, faster or simpler) one for a long time with little luck. The BINASC routine is not yet converted to switch to scientific notation when numbers get too big or too small, nor do I have an easy way of telling the math package how many digits I want to print. These are not trivial matters, but they are all "doable".

Well, that's about all there is to the FPDIV. Though the code is shorter than the FPMUL code, the 55 times through the subtract and shift loop do take longer than the 10 partial multiplies and adds of the FPMUL. Since we have some space here, but I really want to let the ascii - binary conversions rest a while, I am going to do some timing tests. These are on my relatively slow 10 Mhz 68000 system. Results are based on a loop to do the operations many times so I can time them with a stopwatch.

```
Multiply:  1280 microseconds
  Divide:  3838 microseconds
     Add:  684 microseconds
```

I had expected the divide operation to be slower than the multiply, but not by that large a margin. The three functions were tested with the two values 5 and 9. Multiply time is independent of the values of the arguments so that ought to be a constant. Divide time is a function of the number of subtractions that can be made. 5/9 is a continuing fraction with about half 1's and half 0's in binary. To be precise, the hexadecimal value of the quotient of 5/9 is 471C71C71C71C780. The 80 on the end is the exponent. I count 28 1's. That means this time is about the average for a divide operation. Add time again doesn't vary with the argument.

Next I decided to test the new PT-BASIC with its 13 digit BCD arithmetic and I arrived at the following, having subtracted out the time for an empty for-next loop (that really becomes the add test in BASIC):

```
Multiply:  210 microseconds
  Divide:  2023 microseconds
           (2.023 milliseconds)
     Add:  165 microseconds
```

You might eye these suspiciously and say that, for example, the divide only took as long as five or six add operations, but we know the divide probably had 27 subtractions in it. I point out that the subtracts in the divide operation were just that, only the subtraction of two binary numbers, while the add operations that were timed included testing both arguments for zero, denormalizing the smaller one, and normalizing the result, plus the handling of signs and exponents.

After looking at these results I decided to attack the problem of getting the math operations in assembler to run faster. The obvious place to start is with the FPDIV operation since it is so terribly slow. Within that operation is a loop that is executed 55 times, and that would


```
int eflag=0;

main(argc,argv)
char **argv;
{
    int i;

    dash(argv)
    {
    case '?':
        printf("use: cfree {-e} - displays free memory by color\n");
        printf("\n%s\n",copyrite);
        exit(1);
    case 'e':
        eflag=1;
        break;
    default:
        printf("unknown option: %s\n",*argv);
        exit(1);
    }

    /* first link to init module to get colored memory table */
    psInit=modlink("init",0x0C00);
    if ((int)psInit==ERR)
    {
        printf("cant link to init module\n");
        exit(errno);
    }

    psCMem=(B*)psInit+psInit->_mmemlist;

    i=100;  /* start with 100 fragments */
needmore:
    psMem=malloc(i*8);
    if (!psMem)
    {
        printf("ERROR - can't malloc %ld bytes\n",i*8);
        exit(errno);
    }
    if (getmem(psMem,i*8)>i)
    {
        free(psMem);
        i*=2;
        goto needmore;
    }
/*  printf("Minimum allocation: %ld\n",oMin); */
/*  printf("Memory fragments  : %ld\n",oFrag); */

    printf("    Total Memory :  %s\n",commas(oRam));
    printf("    Current Free :  %s\n",commas(oFree));

    i=-1;
    oCFree=oCSeg=0;
    while (psMem->oAddr)
    {
        if (i==-1 || psMem->oAddr<psCMem[i].oLow ||
              psMem->oAddr>=psCMem[i].oHig
        {
            if (i!=-1)
            {
                printf("      Total Free :  %s\n",commas(oCFree));
                printf("      Largest Seg:  %s\n",commas(oCSeg));
                oCFree=oCSeg=0;
            }
```

```
            /* either first or out of curent psCMem[i] range, find new match */
            i=0;
            while (psCMem[i].wType)
            {
                if (psMem->oAddr>=psCMem[i].oLow &&
                    psMem->oAddr<psCMem[i].oHigh) break;
                i++;
            }
            if (!psCMem[i].wType)
            {
                i=-1;
                printf("ERROR - address '%08lX' not in INIT colored memory
                    list!!\n",psMem->oAddr);
            }
            else
            {
                disp(&psCMem[i]);
                bFree[i]=1;
            }
        }

        if (eflag)
            printf("        $%08lX   %s\n",psMem->oAddr,commas(psMem->oSize));
        oCFree+=psMem->oSize;
        if (psMem->oSize>oCSeg)
            oCSeg=psMem->oSize;
        psMem++;
    }
    printf("      Total Free :  %s\n",commas(oCFree));
    printf("      Largest Seg:  %s\n",commas(oCSeg));

    i=0;
    while (psCMem[i].wType)
    {
        if (!bFree[i])
        {
            disp(&psCMem[i]);
            printf("      Total Free :  %s\n",commas(0));
        }
        i++;
    }

    munlink(psInit);
}
```

(Continued Page 43)


```
 lea sign(pc),a2
 move.b d7,(a2)
 move.b 8(a1),d7
 and.b #$80,d7
 eor.b d7,(a2)
* clear exponent bytes for the divide
 clr.b 7(a1)
 clr.b 15(a1)
 move.b #55,d2 loop count
* now we are ready to divide
div1 move.l (a1),d7
 cmp.l 8(a1),d7
 bcs.s div3
 bne.s div20
 move.l 4(a1),d7
 cmp.l 12(a1),d7
 bcs.s div3 can't subtract
div20 move.l 12(a1),d7
 sub.l  d7,4(a1) result in 4(a1)
 move.l 8(a1),d6
 move.l (a1),d7
 subx.l d6,d7
 move.l d7,(a1) result in (a1)
 add.b #1,6(a5)
div3 lea -8(a5),a1
 bsr lsrf
 lea (a5),a1
 bsr aslf
 lea -16(a5),a1
 sub.b #1,d2
 bne div1
* clean up stack and normalize
clean move.l (a5),(a1)
 move.l 4(a5),4(a1)
clean1 lea 8(a1),a5 point past result
 bsr normal
fixsin move.b sign(pc),d7
 or.b d7,(a1)
 move.b exp(pc),7(a1)
restor movem.l (a7)+,d6-d7/a2
 rts

* numerator zero, return zero
nzero SUB.L #8,a5
* zero value is already in the right
* place
 bra.s restor

* denom zero return largest number
dzero lea -16(a5),a1
 move.l #$7FFFFFFF,(a1)
* return large pos number
 move.l #$FFFFFFFF,4(a1)
 lea 8(a1),a5
 bra.s restor

aestk ds.b 128
```

Again we start with the two arguments pushed on our stack pointed at by a5. We back a1 up by 16 locations to point at the start of the first argument and then begin. We save some registers that are used in FPDIV so they can be restored later. Then we clear an area for the result beyond the end of the stack and begin processing by testing for either argument being zero. Zero divided by anything is zero so if the numerator is zero we branch to NZERO which returns zero immediately.

On the other hand if the denominator is zero, we should have a divide by zero error. I've chosen instead to return the largest number that we can represent (of the proper sign).

If neither argument is zero, we subtract the exponent of the denominator from the exponent of the numerator and save it, clearing the exponent locations in both arguments.

Now we are ready for the shift and subtract algorithm. We have a sign bit which already has been stripped away, and 55 bits of mantissa to process, so we set up a loop count of 55 in d2. Now we do our best to compare the numerator and denominator as quickly as possible. We compare the high order 32 bits. If the numerator is less than the denominator we know we can't subtract; so we skip the subtract immediately.

If the numerator is larger we know we can subtract so we skip to that immediately. While, if the high order parts are equal we must compare the low order. Again, we decide whether we can subtract or not. As we said, if we can subtract we put a 1 in the quotient and do the subtraction. If not we put in a zero.

Next we shift the denominator to the right, shift the result to the left, subtract one from the loop count and go around again. When we're done, we have the quotient just above the end of the stack.

The label "clean" is the start of the cleanup. We move the result to the ntos position, move the stack pointer back, fix the sign and exponent of the result and exit. The labels 'clean' and 'clean1' are going to be entry points for the finish of the FPMUL and FPADD routines. We essentially do the same thing at the end of each of the operations for cleanup. Rather than

BEGINNER'S CORNER

by Ron Anderson

The largest positive exponent is $FF which represents 2¢127. The smallest non-zero number that can be represented will be 2¢-127 ($01)...". Somehow the 2 (up-arrow)127 became 227 and the 2 (up-arrow) -127 became 2. 127!

This time we are going to complete the four math functions in Assembler for our floating point math package serie. All that is left of the math package proper is the FPDIV routine. As usual we will start with a simple problem to show the idea. Let's divide 70 by 7 in binary:

```
70 = 1000110 and 7 = 111
   therefore 70 / 7 is 1000110 / 111

        01010
111 ┌ 1000110
      000
      ---
      1000
       111
      ----
       00111
         111
        ----
          0000
```

In decimal arithmetic it is more complicated because we have to see "how many times the denominator 'goes into' the numerator". In binary it either goes or it doesn't. We bump the denominator along the numerator and if we can subtract we put a 1 in the quotient and if not we put a zero in the quotient. If we align the 111 with the first three digits 100, we see we can't subtract so we put a zero in the quotient. Moving one place to the right (or "bringing down a zero" as I was taught long division a long time ago) we can subtract 111 from 1000 (seven from eight) to get a difference of 1. We put a 1 in the quotient and "bring down" a 1. We can't subtract 111 from 11 so we put a zero in the quotient and bring down another 1. Now we can subtract 111 from 111 to get zero, so we put another 1 in the quotient. We have a zero left so we bring it down and can't subtract 111 from the 0000 remainder. We therefore put a 0 at the end of the quotient. We've divided 70 by 7 and the quotient is 10, which is the decimal equivalent of the quotient 01010.

It occurs to me on writing this that perhaps some of the younger readers are not from the B.C. era (Before Calculators) and perhaps they never really learned "long division". If that is true of you, you'll have to study this a little harder. (I have an old slide rule in my desk. I call it my solar powered calculator. All it needs is light to operate. (And a clear head. Ed) In the days before calculators, a good slide rule cost more than a good scientific calculator does now!)

The divide algorithm is simple. Starting at the high order end of the numerator, slide the denominator along from left to right and subtract whenever we can. If we can't subtract we put a zero at the right end of the result. If we can, we put a 1 at the right end of the result. The whole thing lends itself to a shift algorithm and that is what we do. The routine is presented below:

```
* FPDIV floating point divide 64 bit
* version divides ntos by tos, leaves
* result tos

FPDIV equ *
 movem.l d6-d7/a2,-(a7)
* clear result area at (a5)
 clr.l (a5)
 clr.l 4(a5)
* test for zero
 lea -8(a5),a1
 bsr tstzer
 beq dzero denominator is zero
 lea -16(a5),a1
 bsr tstzer
 beq nzero numerator is zero
* subtract exponents for result
 move.b 7(a1),d7
 sub.b 15(a1),d7
 add.b #$80,d7
 lea exp(pc),a2
 move.b d7,(a2)
* sign of the result
 move.b (a1),d7
 and.b #$80,d7
```


OS9: *the Q&A* by Paul Pollock

This month's Q & A got 2 interesting questions, which eventually led to quite long answers. Not what I had originally expected, but hey... that's the way it sometimes goes in this business.

Q I'm trying to build an RS-232c pack with two(2) ports, but I'm a bit fuzzy about what the pak does with interrupts, and once sent, what does the CPU do with it?

-Dean Lieber
Tarzana, CA

A Yes, you may set up either 2 ports in one pack, or 2 separate packs in separate MPI slots. The only thing you have to make sure of is that each port is decoded to a different set of I/O registers; AND you must ensure that IRQ's generated by either/both ports will get to the CPU (via the MPI hack, or building both ports into the same pack).

OK, basically the question is fairly simple, but the answer is either simple or complex depending on where we start to look. Firstly though, I think we should talk about the serial port hardware/software handling in more general terms.

Firstly, the serial-pack (those containing the 6551 ACIA) does NOT have to send an interrupt, and in many environs, it doesn't. The only provision for a true interrupt (or IRQ) is the receive/transmit IRQ's (each is separate, and may be engaged separately), which can be flagged on/off when setting up the port.

However, the IRQ is tied to pin eight(8) of the slot the serial-pack is plugged into. This is then carried back (via the MPI card-edge, also on pin 8) to the GIME chip, and then resent to the CPU (pin 3, the INT pin), which is then handled by whatever software routine is toggled via the appropriate Vector Table offset.

[NOTE: This all assumes the MPI has been properly 'hacked', by tying all four(4) cartridge slots (at pin 8) together, and that the CLOCK module or the CPU interrupt hack has been performed; so they all have a simultaneous chance to be properly returned to the CPU.] (See CPU Interrupts elsewhere in this issue)

Let's use a sample scenario, to describe the mechanics of a serial-port interrupt. Assume that we are using receiver IRQ (normal for OS9), and that the serial-port has just received a character in it's buffer, without errors. Nextly, let's make clear that the operating system serial port driver software is loaded and running. And let's further assume the operating system is OS9.

The first thing that will happen, is that the IRQ will be forwarded to the buss. The buss will carry the IRQ on pin-8 of the MPI, or main buss; eventually reaching the CPU (at pin-3).

The next thing the operating system does, is to establish a special group of vectors, called the: 'Interrupt Polling Table'.

When an IRQ is read, the system checks the clock, and uses it later to interface to other software functions. It then checks the IRQ Polling Table' which will provide a system level vector to an entry within the responsible driver. The table also provides the hardware addresses for I/O to the specific hardware, flagged by the IRQ. This tells the driver which hardware (if there is more than one port per driver) to control.

While all this is going on, OS9 has a process table, which defines a group of information called a: 'Virtual Process Mask'. This mask contains everything required to identify everything the OS9 needs to know about a specific process, and any hardware, and software necessary. The tick clock normally defines which process is identified, and therefore, which mask is in place at a particular time-hack.

The upshot of these masks, is that when a character in the serial port, causes the port to send the IRQ, the CPU goes to a Vectored address defined by the appropriate mask. This vector mask, then is used to check the process mask, for that particular clock tick, which then tells the operating system

not only to call the driver to process that character from the identified serial-port, but then the driver now has someplace to send the character, ie; the appropriate applications program (and it's associated process).

These function so smoothly, that several serial-ports, may make use of the same driver, and generate the same interrupt type. They are then identified as unique, merely by assigning the actual hardware to specific/unique address registers for I/O and control.

THE ACIAPAK DRIVER

The ACIAPAK driver is responsible for operating any 6551 ACIA via hardware address routing, and a specific hardware interrupt. It has one important caveat.

Firstly, ACIAPAK assumes that the MPI *WILL* be used, and that the associated pack will operate in SLOT-0 (front-most on the MPI). Actually, since the serial-port is a 'fully address decoded' device, the MPI select coding is/was unnecessary.

Since it is there though, it is a minor impediment to having two unique serial-port packs in the system, placed in different slots, even if they are assigned to different I/O blocks (this is the primary reason for tying all pin-8's, in the MPI together).

ACIAPAK is responsible for reading/writing data, buffering I/O from the port, and handling hardware and software errors. Because the base address of a given serial port is 'soft-coded' into the device descriptor, the driver CAN control many 6551 ports, simultaneously (assuming the interrupt hack is performed on the MPI, or that all ports originate from the same MPI slot).

Any time a specific serial-port descriptor is called, by opening paths to it; the driver then receives from the descriptor a base address to initialize all hardware specific to that port. It then sets up unique buffers in system-space for all paths open, unique to each port being used. In this way, data from one port, will go to only its buffers, and travel then only to the calling program, via the process and path descriptors uniquely.

IRQ's, beyond those considerations, are handled entirely by the operating system and the ACIAPAK driver (or SACIA, if that's what you're using). You need not concern yourself beyond this point.

Q I have located a used MPI. But when it's plugged into the system, all I get out of the system is a blank screen, with no signon. I have had it checked for voltages, by a friend of mine, and he couldn't find anything obvious wrong with it. What could be the problem?

-Timothy Mohr
Marysville, CA

A This is purely a hardware discussion. Firstly, let's talk a little theory. By itself, your computer is OK, so the ROM is doing it's job, and I assume that when you plug your disk controller into the computer; it too has no difficulties. This means that two things are true. The computer is mostly operating, and that the cartridge interrupt routine and SCS reserved routine is being serviced properly by the rom, and responsible hardware.

This being true, we can then assume that the main difficulty is in the MPI. But, just to be on the safe side, we should discuss simple fixes; just to cover the bases.

NOTE: The PAL chip does not have an effect as to whether the computer will work or not; in absolute terms. Even a brand new MPI of either type, unmodified to work with a Coco-3, will at least startup and operate.

1) The MPI has a complete power-supply (+5vdc, +12vdc, -12vdc), but the 5vdc buss is paralleled with the 5vdc buss in the computer. If the 5vdc in the MPI is inop, is turned on, while plugged into the computer, or is operating without a significant load against it; it is easy to fool the operator into thinking the power-supply is good. When connected to a fully loaded computer, the cartridges plugged into the MPI would cause the computer's power-supply to load down, and only the analog circuits would start.


```
RUN gfx2("revoff")
RUN gfx2("curxy",ItemLoc(CurMenu,1,
    Curitem),ItemLoc(CurMenu,2,Curitem))
RUN gfx2("revon")
PRINT MenuItems(CurMenu,Curitem);
RUN gfx2("revoff")
OldMenu=CurMenu
OldItem=CurItem
RETURN

1000 REM execute function selected by user
IF CurMenu=1 THEN 1100
IF CurMenu=2 THEN 1120
IF CurMenu=3 THEN 1130
IF CurMenu=4 THEN 1140
REM if we get here something went wrong
PRINT "Error. CurMenu\1 or >4"
STOP
1100 REM DIR functions
ON CurItem GOSUB 1111,1112,1113,1114,
    1115,1116
RETURN
1111 REM Disk directory
RUN DiskDirectory(dum$,FALSE,"")
RETURN
1112 REM Change dir
REM not done yet
RETURN
1113 REM Copy directory
REM not done yet
RETURN
1114 REM Create directory
REM not done yet
RETURN
1115 REM Delete directory
REM not done yet
RETURN
1116 REM end program
SHELL "tmode echo"
END

1120 REM file functions
ON CurItem GOSUB 1121,1122,1123,1124,1125
RETURN
1121 REM copy file
RUN CopyFile
RETURN
1122 REM rename file
REM not done yet
RETURN
1123 REM Delete file
RUN KillFile
RETURN
1124 REM list file
REM not done yet
RETURN

1125 REM attributes
REM not done yet
RETURN
1130 REM System information
ON CurItem GOSUB 1131,1132,1133
RETURN

1131 REM Free memory
RUN MFree
RETURN
1132 REM module directory
RUN MDir
RETURN
1133 REM Processes
RUN Procs
RETURN

1140 REM Disk functions
ON CurItem GOSUB 1141,1142,1143
RETURN
1141 REM Free space on disk
RUN Free
RETURN
1142 REM backup file
REM not done
RETURN
1143 REM format diskette
REM not done yet
RETURN
REM * end select *

9000 REM Display main screen
RUN gfx2("clear")
RUN gfx2("curxy",25,3)
RUN gfx2("boldsw","on")
PRINT "Basic09 User Interface";
RUN gfx2("boldsw","off")
RUN gfx2("curxy",22,19)
PRINT "Left/Right Arrows to select group";
RUN gfx2("curxy",22,21)
PRINT "Up/Down Arrows to select item";
RUN gfx2("curxy",24,23)
PRINT "ENTER to activate function";

REM second line of titles
RUN gfx2("curxy",8,7)
PRINT "Functions";
RUN gfx2("curxy",25,7)
PRINT "Functions";
RUN gfx2("curxy",45,7)
PRINT "Information";
RUN gfx2("curxy",61,7)
PRINT "Functions";

FOR CurMenu=1 TO 4
FOR CurItem=1 TO NumItems(CurMenu)
RUN gfx2("curxy",ItemLoc(CurMenu,1,
    CurItem),ItemLoc(CurMenu,2,CurItem))
PRINT MenuItems(CurMenu,CurItem);
NEXT CurItem
NEXT CurMenu
FOR Knt=1 TO 4
RUN gfx2("curxy",MenuTitleLoc(Knt,1),
    MenuTitleLoc(Knt,2))
PRINT MenuTitles(Knt);
NEXT Knt
RETURN

10000 REM DATA statements for menus

RESTORE 10001
FOR Knt=1 TO NumItems(1)
READ MenuItems(1,Knt)
READ ItemLoc(1,1,Knt)
READ ItemLoc(1,2,Knt)
NEXT Knt
10001 DATA "Disk Dir",8,9,"Change Dir",8,10
DATA "Copy Dir",8,11,"Create Dir",8,12
DATA "Delete Dir",8,13,"* Exit *",8,14
REM File func menu data
FOR Knt=1 TO NumItems(2)
READ MenuItems(2,Knt)
READ ItemLoc(2,1,Knt)
READ ItemLoc(2,2,Knt)
NEXT Knt
DATA "Copy File",25,9,"Rename File",25,10
DATA "Delete File",25,11,"List File",25,12
DATA "File Attributes",25,13
REM System info menu data
FOR Knt=1 TO NumItems(3)
READ MenuItems(3,Knt)
READ ItemLoc(3,1,Knt)
READ ItemLoc(3,2,Knt)
NEXT Knt
DATA "Free Memory",45,9,"Module Dir",45,10
DATA "Processes",45,11
REM Diskette function menu data
FOR Knt=1 TO NumItems(4)
READ MenuItems(4,Knt)
READ ItemLoc(4,1,Knt)
NEXT Knt
DATA "Free Space",61,9,"Backup",61,10
DATA "Format Disk",61,11

FOR Knt=1 TO 4
READ MenuTitles(Knt)
READ MenuTitleLoc(Knt,1)
READ MenuTitleLoc(Knt,2)
NEXT Knt
DATA "Directory",8,6,"File",28,6
DATA "System",47,6,"Diskette",62,6
RETURN
```

<EOF>

```
PROCEDURE Bui

REM predeclare all variables

DIM MenuItems(4,6):STRING[11]
DIM Menutitles(4):STRING[10]
DIM ItemLoc(4,2,6):INTEGER
DIM CurItem,CurMenu:INTEGER
DIM char:STRING[1]
DIM ChNum,LeftAr,RightAr:INTEGER
DIM NumItems(4):INTEGER
DIM DownAr,UpAr,ENTER:INTEGER
DIM MenuTitleLoc(4,2):INTEGER
DIM Knt,OldItem,OldMenu:INTEGER

REM declare constants used in pgm

NumItems(1)=6
NumItems(2)=5
NumItems(3)=3
NumItems(4)=3
LeftAr=8 \RightAr=9
DownAr=10 \UpAr=12
ENTER=13

REM Read in DATA used for menus

GOSUB 10000

REM Main Control
REM *NOTE* Make sure line numbers
REM are typed in correctly

1 CharNum=0
OldMenu=1 \OldItem=1
CurMenu=1 \CurItem=1
GOSUB 9000
CurMenu=1 \CurItem=1
GOSUB 500

10 char=""

REM Turn off echo to eliminate prob
REM with up arrow key

SHELL "tmode -echo"
WHILE char="" DO
RUN inkey(char)
ENDWHILE
CharNum=ASC(char)

IF CharNum=LeftAr THEN
CurMenu=CurMenu-1
CurItem=1
IF CurMenu<1 THEN
CurMenu=4
ENDIF
GOSUB 500
GOTO 10
ENDIF

IF CharNum=RightAr THEN
CurItem=1
CurMenu=CurMenu+1
IF CurMenu>4 THEN
CurMenu=1
ENDIF
GOSUB 500
GOTO 10
ENDIF

IF CharNum=DownAr THEN
CurItem=CurItem+1
IF CurItem>NumItems(CurMenu) THEN
CurItem=1
ENDIF
GOSUB 500
GOTO 10
ENDIF

IF CharNum=UpAr THEN
CurItem=CurItem-1
IF CurItem<1 THEN
CurItem=NumItems(CurMenu)
ENDIF
GOSUB 500
GOTO 10
ENDIF

IF CharNum=ENTER THEN
SHELL "tmode echo"
GOSUB 1000
SHELL "tmode -echo"
GOSUB 9000
CurMenu=1 \CurItem=1
GOSUB 500
ENDIF
GOTO 10
```

```
REM Update menu display

500 RUN gfx2("curxy",MenuTitleLoc(OldMenu,1),MenuTitleLoc(OldMenu,2))
RUN gfx2("revoff")
PRINT Menutitles(OldMenu);
RUN gfx2("curxy",ItemLoc(OldMenu,1,OldItem),ItemLoc(OldMenu,2,OldItem))
PRINT MenuItems(OldMenu,OldItem);
RUN gfx2("curxy",MenuTitleLoc(CurMenu,1),MenuTitleLoc(CurMenu,2))
RUN gfx2("revon")
PRINT Menutitles(CurMenu);
```

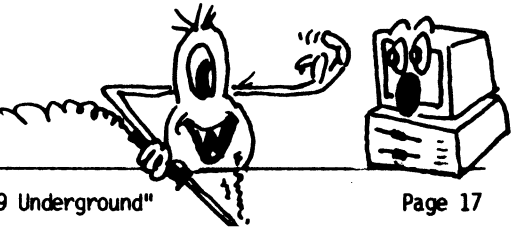

Test this by NOT connecting the MPI to the computer. Check the power-supply voltages, both with cartridges in the MPI, and without. If there is a significant difference (more than .25vdc), or if any voltage has significant AC ripple (caused by a bad filter cap, or a defective rectifier, in that supply), the power-supply is defective. Determine the defective component (regulator, pass transistor, filter cap, or rectifier) and replace it/them. Recheck for proper voltages, and then retest the complete computer system for proper operation.

2) The card-edge connector (the end that plugs into the computer), in the MPI, gets very dirty/corroded. It's plated with tin. When tin comes into contact with a dissimilar metal (copper, gold, silver) and mixes with moisture; a chemical reaction takes place. This reaction generates powder-corrosion on the tin plating. This corrosion causes intermittant or completely open circuits in one/several connections in the card-edge.

This can be serviced by taking the eraser from an ordinary pencil, and using it to rub the card-edge clean. Do this to both sides of the card-edge (top/bottom). Retest the computer, when you are certain that the card-edge is clean.

3) The front-panel switch may be defective. It must be placed in position #4 (slot-3, numbered 0-3), for the MPI to operate correctly with a disk-controller. But, even if it were not in position #4; assuming the switch were working, the computer would startup 'as if' the disk controller were NOT plugged in. A defective switch (totally open), would cause the rom driven cartridge interrupt routine to open up for the appropriate ram-hook to load the rom in the cartridge, but finding none, would get lost and disappear. Never to return (a blank green screen).

Service this switch, by first eyeball inspecting it. This switch is an open-frame variety, easy to verify. Make sure the inside slide contacts are working. Then, clean these contacts by spraying 'tuner cleaner' into the switch and forcibly sliding the switch back and forth. Retest the computer system and check for correct operation.



4) All cartridges placed in the MPI will face away from you. The label side will face towards the back of your computer table. This seems a simple fix, but several people have found that an MPI *CAN* allow a cartridge to be plugged in, completely backwards! Normally this causes no perminent damage, but it CAN damage the cartridge, MPI, or computer. Ensure that your cartridges are plugged in correctly (also pre-check that they all work correctly in the computer, without the MPI).

MORE DIFFICULT SOLUTIONS

Firstly, as we both know, there are two varieties of MPI. The old largecase MPI; and the new, smaller MPI. The largecase MPI (26-3024) used only 'off the shelf' components, except for the pre-programmed PAL chip which drives a matrix controller chipset, which then selects the MPI slots.

The smallcase MPI (26-3124) had three(3) custom IC's on the board. The significance of this fact, is that while the older largecase MPI can be fixed by the replacement of easily found components, the smallcase MPI may need the replacement of the entire circuit board, as these custom components are no-longer-available via TANDY, as separate components. This may not be necessary though, as the majority of parts are still common components.

The discussion that follows assumes that the simpler solutions have already been tried, and that all other faults have been ruled out.

For each slot in the MPI, there is an associated 'hex-gate' buffer chip. This is used to select one of three(3) lines for each slot. These chips are normally very close to the main card-edge connector of the MPI. These can be tested and/or replaced; to restore a defective buffer latching condition.

If any combination of these chips are defective, the trouble can mimic an open slot-select switch. This will either cause the computer to fail to start, or if it does, to operate in a peculiar fashion.

Unfortunately, these latches are soldered into the board on both MPI's. They are a pain in the neck to change. But they are the most common form of hardware defect, second only to a power-supply failure.

The +5vdc is driven by a LM723 IC regulator. It uses a large transistor to pass current. In the older MPI, this was a big round transistor with a rectangular heatsink. The newer MPI has a transistor and heatsink which hangs outboard of the board.

While the regulator and transistor is very rugged, they are fused by a 1-watt 6.2vdc zener diode. This diode burns out because of a short duration short at any point in the MPI power +5vdc buss. Normally it's caused by a intermittant break/make condition at the card-edge mating the computer and MPI while power is one, or when a cartridge is plugged in or removed from the MPI while power is on.

Test/replace this zener diode with a 5-watt +5vdc zener. This will usually restore a confirmed bad +5vdc power supply, and will also be a much more durable circuit.

The +12vdc and -12vdc voltages can also become defective, and are solved in a more conventional method; however, they are much more durable and don't normally fail. But even if they do, they normally will not prevent the computer from starting.

Normally these supplies become defective because of a bad filter cap or rectifier. The regulators are LM7812, LM7912 IC's; or like items. They are monolithic and have no pass transistors external to the chip. They are internally protected from short-circuit conditions, so they are usually very rugged. They also have no safety zeners, so they usually don't get shorted out easily.

If your MPI is more broken than the trubles described, it's real broke and should be taken to a qualified technician for repair.

Your questions to Paul Pollock sent via E-mail (See E-Mail Address page 50) or c/o this magazine.

<EOF>

BUI: A B09 Interface to OS-9

by Randy Krippner

Welcome "OS9 Underground". This series is generating a Basic09 User Interface to OS-9. The idea is to make it easier for proficient CoCo RS-DOS user to begin using OS-9. We are more than half way through the series. There are my procedures already presented in past issues of "68xxx" needed to make the code presented below run.

Space is a little tight this month so this time the code is presented and next month I'll explain it. Howver, this one overview comment is necessary. The procedure is has a lot of GOTOs and such, but it works. For my money, structured programming is all fine and dandy, but when it comes to getting the job done, "pretty" code just doesn't hack it in the real world.

This month we have a brand new menu procedure, called BUI. Scrap the entire menu that was used in the previous version of the program and use this one instead. It's a lot easier to use, multiple menus on one screen and functions that weren't accessible in the previous procedure. Also note that not all of the functions are "live" yet. Those functions will be added in future articles.

The menu procedure is called BUI (see listing next page) for the simple reason that it's easier to work with that way. Calling the procedure that is to be executed first by the same name as the program has advantages that should be obvious. Next time we'll take a look at exactly what the code is doing in detail, and perhaps add a new procedure or two if there's enough space. -RK-

As always, questions, comments, suggestions, bug reports etc. should be sent to me at 1014 W. Hwy. 114, Lot 29, Hilbert WI. 54129. An SASE would be appreciated if you want a reply.

As to which technology will 'win' (if there is a winner) remains to be seen. A better technology or larger size Company does not insure winning. BETA lost out to VHS. Unknown Lear took on the giant Phillips in the 50s and 60s with his 8-track audio tape format vs Phillips 4-track format and split the market pretty evenly for many.

CD-I MACHINES TO COMPUTERS

Will new CD-I machines be adaptable to conversion as computers? Certainly, a competent hardware hacker can do remarkable things. But they will still have the problem of obtaining and adding the necessary drivers, utilities, etc. to make their conversion useful. As pressure to reduce the price increases the machines are unlikely to include provisions for a 'plug-in' expansion port. Manufacturing costs will be at the top of the agenda and the manufacturers will not include anything not necessary to the primary function of their equipment.

A high-level source at Opti-Image believes that if anyone wants to convert a CD-I player, they should obtain one of the current machines on the market. He believes the design of the new machines from Sony and Magnavox/Phillips will be such as to make such a conversion impractical. (No mention was made of other suppliers.) But, they will face the software problem mentioned above.

The author made no mention of CD-I machines available from Opti-Image, Philips and Sony which already include the additional hardware and ports for use as computers. These are available only on special order from the respective manufacturers and so far as I've been able to ascertain, do not include the additional software required (excepting the Opti-Image hardware). And, the price of these machines is very high.

I think it unlikely we'll see any expansion kits for CD-I players offered by the respective manufacturers. Of course, they could put one of their high end machines into production but this wouldn't be an expansion kit.

In any event, there is absolutely no way "we would have 20 million or more OS9/68000 users in 3 to 5 years". Of course, if you count some of the automobiles and household appliances, we may have over 20 million OS9/68000 users already.

VENDOR RESPONSIBILITY

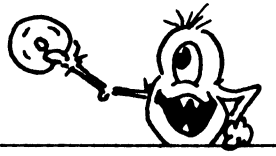
Now please, don't attack me for being pessimistic or negative about CD-I. I'm not. I am very 'negative' towards any vendor in this market who makes irresponsible predictions or claims. Re-read the article. If you are on friendly terms with the owner of a 'brown' goods retail establishment, take it to him and get his reaction. Odds are his first question will be, "what is CD-I?". Go to your local Sears, Circuit City or Radio Shack and get their reaction to CD-I sales.

As a vendor to the OSK market, I feel it is our responsiblity to insure our customers and prospects have the best and most accurate information we can provide. Nothing wrong with predictions so long as there is a valid basis for them. Had the author of the article taken the time to talk to retailers and spent a little time reading trade publications, he 'could' have found what is happening in the market place.

I am surprised the author of the article didn't try to convert a CD-I machine and write an article about the results, especially since he claims designing and building an interface to his machine is very simple.

In addition, I wonder why the Editor of this publication printed such an article without asking the writer to substantiate his numbers and claims. I'm not aware of any electronic consumer product that has ever come close to achieving the acceptance forecasted in the article.
[Editor's note: Above reference, was to editor of '68xxx Machines'. Also see Ed's comments in "Letters to the Editor"]

Ed Gresick DELMAR CO
PO Box 78 Middletown, DE 19709
302/378-2555 * FAX 302/378-2556



<EOF>

DD BASIC TRAINING

By Jim Vestal

Hi, another month has gone by since we last talked about structured programming. I have no idea what your thoughts as to the format of this column. Due to the lag time from submission to publication I'll go on the assumption you like what's here and are learning from it.

PROGRAMMING'S FOUR LETTER WORD

Programming's four letter word is GOTO. GOTO allows the program to jump from one line of code to another unconditionally. It is a very convienient command to use when you need to form a loop within a program. When BASIC was first developed, programmers knew how to prevent writing what is known as spaghetti code. Spaghetti code refers to a program that jumps from one spot then jumps again to another spot not following any logical order or direction. If a line were drawn from line number to line number where all the jumps occured you would find that the lines crossed, sometimes many times over. Thus the result appears similar to long strings of cooked spaghetti. This resulted in a program that could have several bugs in it, even if such a program worked, trying to follow non-consistant logic in a program in order to update the code or add to it was a true pain to say the least.

LINE NUMBERS AND LOOP STRUCTURES IN BASIC09

Basic09 adds many features of other structured programming languages that older BASICs lacked, such features as optional line numbers and built-in loop structures such as REPEAT-UNTIL, WHILE-DO,and LOOP-ENDLOOP/EXITIF allow programmers more flexibility to program more structured code. If a BASIC programmer can grasp the concepts of using NO line numbers and program in a top-down design and only loop using the above mentioned structured paired loop statements; if that same programmer can get in practice of not using GOTO statements in a program.

The possibility of writing spaghetti code is eliminated all together. All habits, good or bad, takes practice to establish. It is not easy to break bad habbits, but in time good habits can replace the bad habbits.

RULES AND THE EXCEPTIONS

Here a list of guidelines that I use when I write programs in Basic09.

1) If I want to write a new program written from scatch:

a) Before actually coding the program, plan ahead and make some notes of what you want your program to do. You DO NOT have to create sophisticated flowcharts or logic charts. I usually create a small outline of what I need my program to do and in what order I want things to be done. That way if I am stuck in knowing what to do next I just look at the outline.

b) Use NO line numbers (period). This makes using GOTO's less tempting. The ONLY exceptions is if you have a need to use subroutines via the GOSUB statement or have a need to trap errors with the ON ERROR GOTO statement.

c) If you find that your subroutines are in fact seperate programs you may want to use the Basic09 RUN statement and make them seperate procedures. I almost alway do this instead of using line-numbered GOSUBS...it's a matter of personal choice though, not a hard rule. In some cases (very few in my opinion) using numbered subroutines are a better choice than a seperate

procedure.

d) Try to take advantage of the loop structure commands, REPEAT/UNTIL and WHILE/DO. If you need to branch out of a loop instead of using a GOTO consider restructuring your program with a LOOP/ENDLOOP and you can use a EXITIF/ENDEXIT to jump of the loop. Also even though EXITIF is usually used only within a LOOP/ENDLOOP loop, it can be used at anytime within a REPEAT/UNTIL, WHILE/DO, FOR/NEXT loops.

2) If you are converting an old program (say from disk basic) that eithor you or another person has written it may not be possible to follow the above guidelines. Here's a few tips though:

a) Re-type the program with no line numbers...if GOTO or GOSUB statements exist then only number those lines that are jumped to. You may find that a well written program may only have three or four linenumbers, if not less than that.

b) If a program can be re-written using REPEAT/UNTIL or WHILE/DO loops, please consider doing so. You may find that the reason a programmer used a GOTO statement is because BASIC did not have a loop structure that could handle the logic without using a GOTO statement, or they did not know about the structured loop statements. Take a look at this example program segment:

```
100 read#path,variable
110 IF EOF(#path) then
120  goto 180
130 endif
140 test=variable
150 gosub 3000 \ (* file routine *)
160 gosub 4000 \ (* check routine *)
170 goto 100
180 (* the rest of the program *)
```

Notice that the reason the programmer used a goto in line 130 is 'cause their was an end of file condition that meant that there was no more data to processed in the file, in that case program continues after the processing statements, after processing a record the program jumps back to line 100 which reads another record to be processed. Now in this case both GOTO statements and all the line numbers can be eliminated by re-coding the program using a WHILE/ENDWHILE structure as follows:

```
READ#path,variable
WHILE NOT EOF(#path) DO
  test=variable
  RUN file(test)
  RUN check(test)
  read#path,variable
ENDWHILE
(* rest of program *)
```

Or you can use what I call the "cop-out" method of loop structure, the LOOP/ENDLOOP with an EXITIF condition.

```
LOOP
 read#path,variable
 EXITIF EOF(#path) THEN \ ENDEXIT
 test=variable
 RUN file(test)
 RUN check(test)
ENDLOOP
(* rest of program *)
```

Next month I'll cover in more detail all of BASIC09's structed loop statements. Until then I suggest that all BASIC09 programmers review their copy of Dale Pucket's Basic09 Tour Guide, one of the best books to teach BASIC09.

Jim Vestal may be reached on Delphi as: JIMVESTAL or on the StGNet as SysOp@Narnia, or letters may be sent c/o The OS9 Underground and forwarded to the author.

<EOF>

first thing retailers will do is check on the past sales of CD-I, the reports in the trade journals, etc. This doesn't mean they won't buy if the information isn't exciting, simply means they won't buy as many, will demand better terms, won't push it as hard, etc.

CD-I is a product I will probably have to consider selling sooner or later. I am market driven, too. But, I've tried to illustrate segments of the process I go through in selecting products to sell. I will follow a similar process when my distributors (or Radio Shack) offer CD-I players. And, you may be certain other retailers go through a similar process. The main difference is how we weigh the different factors. I would love to see the 20,000,000 new OSK users as predicted in the article, but I don't see that happening.

FUTURE OF CD-I

Where is the CD-I market going? Based on the few articles in the trade journals and conversations I've had with several individuals at MW, Opti-Image (the Company set-up jointly by Microware and Phillips dedicated to selling and supporting the CD-I operating system to manufacturers) and several other Companies, it appears that the game market (Nintendo) will provide the largest number of CD-I based machines initially. Nintendo has announced they expect to have a high end game machine utilizing CD-I for the '93 season. I have no idea as to the numbers they expect to sell. They may have some units for XMAS, 1992 but that is spectulative. As yet they haven't shown anything using CD-I at the trade shows.

Two other markets are expected to emerge. One is the catalog market (Sears is rumored to be considering CD-I for catalogs in their stores) and the second is the education market. Until full motion video is available, I don't think we can expect any kind of consumer acceptance. Of course, there is always the possibility of a use we haven't even considered which could sky-rocket sales. I don't even want to try to hazard a guess as to numbers that will be sold, but I will be (pleasantly) surprised if more than 1,000,000 are sold in 1993.

WHAT TOOK CD-I SO LONG?

An interesting exercise may be to speculate on why Phillips delayed the release of CD-I and then, rather suddenly, released it. According to a series of articles in a Des Moines newspaper where Ken Kaplan was interviewed, Mr. Kaplan indicated there was quite a bit of dissension within Phillips, between Sony and Phillips and between them and Microware. This was probably the major cause of the delay. I'm not aware of any technology in the machines that hasn't been available for several years so technology was not the cause of the delay.

Were it not for Commodore, CD-I might not yet be released. In an aggressive move, Commodore designed CDTV around their 500 computer and released it last summer. It is safe to assume that Phillips was aware of Commodore's plans long before Commodore released their machine and started their manufacturing. When Commodore released their machines the price was $1200. Months later, Magnavox/Phillips released their machine at the same price. Commodore immediately slashed their price to $800 forcing Magnavox/Phillips to follow suit. And while Sears and some of the other retailers have held to a price near $800, Radio Shack has slashed the price of their CD-I player to $500 but there has been no stampede by the public to buy. Of course, there has been almost no advertising of this price cut, either.

Rumors from the Commodore camp claim Commodore is working feverishly on a new CDTV machine for introduction this fall. Target price is $300 although they will probably introduce it at a higher price. It's probably safe to say Magnavox/Phillips are working equally feverishly with similar objectives in mind. Since neither has performed the 'ritual dance' necessary to establish wide distribution, it is unlikely that either will achieve any kind of significant sales this year. Most important, neither has yet released any software (titles) to garner the excitement of the consumer.

number on hand.) We have a large Sears store not too far away and I am friendly with one of their salesmen. He doesn't know (or won't say) how many they have in stock, but he doesn't believe they've sold any in his store.

His statement may not be conclusive but it is certainly indicative that CD-I is not a 'hot' item in his store. Moreover, I haven't seen any significant CD-I advertising from any of these stores or from Magnavox/Phillips. Does this mean CD-I is not a viable product? No, not necessarily.

"I CONCLUDE THE TOTAL MANUFACTURED WAS PROBABLY AROUND 100,000".

Several of the retail trade magazines report monthly sales of various products; i.e., TVs, VCRs, camcorders, laser disc players, etc. I haven't seen any listings for CD-I as yet. Nor have I seen any ads from Magnavox/Phillips in these journals. I suspect the dismal showing of CD-I thus far has nothing to do with the quality of the product. Rather, it's a reflection of what happens when a product is rushed to market. It takes time to do the necessary marketing 'homework'. Also, there are probably no titles of interest to the mass market available which will impact marketing plans.

Now let's talk some numbers. Tandy has a program they use to test market certain new products. They buy a small quantity of the new product from a manufacturer and place them in about 500 selected stores. CD-I is in this category and I'm guessing Tandy purchased 10,000 of them. I do know Tandy still has about 2000 sitting in a warehouse in Texas. They are listed in the 'stock deck' as 'SOWG' (sold out when gone). The price has been reduced to $499.00. For those who want to check further, the Tandy stock number is 16-375. I could get no information as to whether Tandy will continue the CD-I product line. The rumor mill says they will. I'll have better information in August when the new product line is shown at the annual Tandy dealer meeting.

Let's look at Sears and the others. Admittedly, I'm on shaky ground here. I can only speculate but I don't think I'm too far off. (This is the kind of process I go through in selecting products for my store.) Sears probably sells a comparable dollar value of electronics in this category (TVs, VCRs, etc.) as Tandy. So, I would assume they've ordered a similar quantity. I suspect the others (Circuit City, Silo, etc.) probably ordered smaller quantities; i.e., between 1000 and 5000 each. From this I conclude the total manufactured was probably around 100,000. I don't think more were made and it could've been considerably less. (I doubt anyone can get real figures.) I've seen no evidence that any wholesalers or distributers have them. And these are the main distribution channel to most medium and small sized retailers - the ones that comprise the bulk of the retail stores referenced.

So, after almost a year's time, we may conclude with some confidence that no more than 100,000 CD-I players 'could' have been sold and, since many stores still have substantial stock on hand, probably considerably less have been sold. This is a far cry from the 5 to 36 million predicted in the article for the first year. Not to be redundant, but the first year is almost over.

Another clue to the success of a product is the excitement it creates at the retailers' Las Vegas Electronics Show in February and the retailers' Consumer Electronics Show in Chicago in May. CD-I created no excitement at either show. It received almost no coverage in the trade journals following the Las Vegas show and from the preliminary reports I've received from the Chicago show, it will probably continue to receive the same amount of coverage - nil. While there are many factors which can excite retailers, probably the most exciting is the manufacturer who puts together an aggressive marketing plan which convinces the retailer the product will be a 'seller'; i.e., the manufacturer who will 'make' the retailers' market.

If and when a serious sales campaign is ever mounted to sell CD-I, retailers will have to be sold first. And the

USING THE TERMCAP LIBRARY PART II of V

by Bob van der Poel

Last issue we discussed the itermcap library and promised to develop a menu program to illustrate the use of a termcap library. So let's get busy!

First off, let's determine the format of the menu database file. This is a text file we'll call "menu.txt" and it will be stored in /dd/SYS. This file will consist of a series of menu "pages" each with a series of choices. Each choice consists of two items: the description to display on the screen and the action. An action can be a OS9 command to execute or a new menu page to display.

The file uses single character "keywords" to describe its parts. These keywords must start a line.

\# this introduces a menu name. The name should follow immediately after the "#" and be terminated by a carriage return. A subsequent "#" line indicates the end of a menu page.

* a comment. Lines starting with this character are ignored by the menu program.

= the title for a menu page. This can be included anywhere in a menu page, but it makes sense to have it right after a "#" line.

```
FILE: menu.txt

* menu file for the menu program
* menu 'main' must be defined. This is the first menu
* to be displayed. If there's no 'main' the menu will
* terminate with an error message.
#main
=Welcome to OS9
Games Menu,      #games
Wordprocessing, #wp
Utilities,       #util

* here are some games to play....

#games
=Fun and Games
Sokoban,           chd /dd/games/sokoban;/dd/games/cmds/sokoban
Tetris,            /dd/games/cmds/tetrix
Battle Ship,       /dd/games/cmds/bship
Typefast,          /dd/games/cmds/typefast
Cribbage,          /dd/games/cmds/crib

* this takes us to the user's wordprocessing directory

#wp
=Word Processing
Letters,  chd text/cor; dir; shell
School,   chd text/school; dir; shell

* these are a few common programs this user runs

#util
=Battle Command

Call CIS,    chd /dd/telcom; sterm -1 /t0
Make Coffee, echo "zzzzz"; sleep 1000

* it's important to have this final "do nothing" menu to keep
* things tidy. Call it anything you want.

#end
```

Lines containing text following a "# line are considered to be menu options. Each option line starts with the text to display for that option, a comma, and the action. If the action starts with a "#" we attempt to display a new menu page; otherwise we attempt to execute an OS9 shell command.

Before we write any code, let's write a simple menu file. This file has three menus, each menu has a number of options. Note the use of comment lines in the file! (See listing 1)

Often I find it easier to write a file like this first once I have some kind of idea of how to best store the data, it's easier to write code.

I decided that the easiest way to handle the menu screens would be to read the entire menu into memory. This might not be the most memory-efficient way to do things, and it limits the total number of pages, but it's easier to implement and follow than dynamic allocation.

Maybe you want to tackle this as your own little project. However, the way it's done here isn't all that bad either- memory for the actual menu data is allocated dynamically. Each menu screen is contained in the mn structure. An array of structures is allocated so we can have a maximum of 100 menu pages each with a maximum of 24 lines. This is all defined in the common header file used by all of the menu modules: (Listing 2)

I've added a headerfile to my DEFS directory to handle memory allocation function. You'll notice that in "menu.h" I include a file called malloc.h. If you don't have one already, here is one you can add to your DEFS: (Listing 3)

This keeps the compiler happy when you assign the return value of malloc(), etc. to any kind of pointer.

In our next installment we'll write some code to start the program working. Until then, if you want to contact me. I'm at:
PO Box 355, Porthill, ID 83853 or PO Box 57, Wynndel, BC, Canada V0B 2N0 or Compuserve 76510,2203

<EOF>

```
FILE: menu.h

#include <ctype.h>
#include <malloc.h>
#include <modes.h>
#include <sgstat.h>
#include <stdio.h>
#include <strings.h>
#include <termcap.h>

#ifndef MAIN
#define GLOBAL extern
#else
#define GLOBAL
#endif

/* for termcap, number of rows and columns */
GLOBAL int nrows, ncolumns;
GLOBAL char standout; /* is standout avail? */
GLOBAL int numpages;  /* total number of menu screens */
extern int errno;
char *savestr();

#define MAXLINES 24
#define MAXPAGES 100

struct mn{
   char *pagename;
   char *title;
   unsigned char titlelen;
   unsigned char numlines;
   char *choices[MAXLINES];
   unsigned char choicelen[MAXLINES];
   char *cmdlines[MAXLINES];
};

GLOBAL struct mn menupage[MAXPAGES];


FILE: malloc.h

void *malloc(), *calloc(), *realloc(), *ebrk(), *ibrk(), *sbrk();
void free();
```


sold, half were sold the first 12 months after they were introduced. Most of these were to CoCo1/2 owners. Discussions with other Tandy dealers and Company store managers disclosed similar sales patterns.

Trying to correlate CD-I sales with CoCo3 sales is completely fallacious. CoCo3 sales were to an already existing market and represented only a new model to this market. CD-I is a brand new product (although there are market similarities to VCRs and laser disc players).

The article further stated a number of other companies are 'making' CD-I players, lists some of these companies and then goes on to say he thinks they are waiting for video compression chips. Which is it? And, where may one see these machines? Also, have these companies made any public announcements of their intentions? Which of these companies are thinking in terms of 'hundreds of millions'?

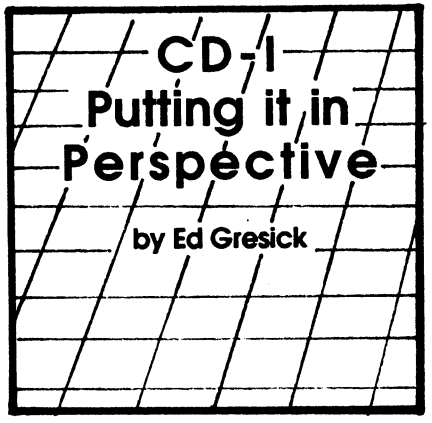
Normally, sales of a new product will start low and grow exponentially until market saturation is approached when growth will flatten or even reverse. I have yet to see a product whose 1st year sales equals the second years which equals the third years sales, as was stated in the article. There is no reason to believe that CD-I players should follow a sales pattern different than the typical sales curve taught in 'Marketing 101' courses.

CD-I SALES HISTORY

CD-I has been avaiable for almost a year; let's look at what has happened in the market place. (The referenced article made no mention of any attempts by the author to investigate this history.) Radio Shack, Sears, Circuit City, Silo and several other large chains have CD-I players. Within Radio Shack, only certain selected stores (their 'top 500') have them. Are CD-I units elling? In my local area, two Company owned stores have them. One has 8 and the other 5. According to the store managers this is what they were shipped last fall and they have not sold any. (This doesn't mean none have been sold. I'm aware of Company stores elsewhere that have sold some but apparantly they still have a large

After reading 'Compact Disc-Interactive (CD-I) and You' by Frank Hogg in the May/June, 1992 issue of 'The 68xxx Machines' I feel it is necessary to challenge the '100,000,000 CD-I units' projected to be sold with-in three years, beginning this year.

According to the 1990 census, there are 92 million households with one or more TV sets. Allowing an additional 50% for homes with multiple TV sets, the total TV market is still under 150 million sets. About 50 million households have VCRs (industry figures). And, it has taken about 20 years to achieve these levels. The author would have us believe that it is possible for CD-I sales to reach two-thirds of the total potential TV market in only three years, and the first year is almost over. Even VCRs haven't reached such a penetration.



FORCASTS ARE OPINIONS

Since it is difficult to get hard, historic figures regarding actual sales of any product and impossible to accurately forecast the future, it is important that the reader be aware of the credentials of the forcaster. Forecasts, at best, are opinions. Here are my credentials.

Since '74, I've had my own Company, DELMAR CO. Most of you may know me best for the SYSTEM IV computer which we distribute. I have a subsidiary, EDELCO, which specializes in custom industrial control software. I'm a VAR and sell VME systems to industry. I also own a Radio Shack dealership (since 1980) and it is the latter business that is pertinent to this discussion.

If you've been following the business news, you've seen that the retailer has been among the hardest hit by the current recession (next to wage-earners). Both large and small retailers have gone 'belly up'. The so-called 'white' and 'brown' goods retailers have been hit the hardest and our business is classified as 'brown' goods.

The retailers still in business are there because they have employed successfull techniques in managing their businesses. They will not respond to words like 'could' in deciding their product lines and quantity to ordered. In my case, I use forecasting methods that have proven reliable to me, read trade journals and competitor ads, listen to distributors, manu-facturers and my sales-people and am constantly trying to be aware of what my customers may want and can afford now and through the next six months.

I am well aware that I do not sell any necessities and I am dealing with my customers' disposable income. The income distribution of the families comprising our local market is near 'average'. We have no nearby 'rich' community. We may have a slightly higher percentage of retirees and these are probably best classified as 'comfortable'. I don't believe I've missed any significant sales opportunities and more important, I haven't overestimated the market and had to resort to distress sales to dispose of excess inventory (except some Xmas toys). While there may be some minor differences in various local markets, most retailers still in business face the same problems I do and react similarly. I am still in business even though profits are very thin and times are very difficult. I believe the above experience and history do permit me to write about this subject with some credibility.

The article states 'There were about 1.5 million CoCo-3s sold ...'. The best information I've been able to get from within Tandy was 600,000 CoCo3s were manufactured. None of this information is official nor will Tandy verify or deny it. Of the total CoCo3s we've

TECH CORNER

by J. Scott Kasten

Well, we have here at last the fractal print program that I promissed in the last edition before we got side tracked. Again, this may be treated as a general utility for printing the simple graphics files that we have been working with. It can also be used to print the results of the graph program shown in this column a couple of months ago.

This utility should work with any 9 or 24 pin printer that has a generic EPSON or IBM emulation mode. Printing graphics is actually made easy with modern printers. There are basicly three things the graphics program has to do:

1) Set a smaller line spacing for carriage returns so there will not be gaps between the lines.

2) Convert the image to graphics print codes.

3) Reset the line spacing to normal for the next project.

In the source code 'print.c', I have labled these areas of the program. The two escape sequences used to set the line spacing should be obvious. What is likely to be more confusing is how the image is converted. The loop structures break the image into strips and convert the color values into alternating on/off's. A print escape sequence is sent to the printer that tells it the next line is graphics data. The value '6' selects a 90 DPI mode. There is then a two character sequence that taken togeather makes a 16 bit binary value representing the number of graphics data bytes to follow. The printer takes each data byte as a bit image of the graphic. Imagine taking each data byte with its on/off bits and standing it verticly with the MSB on top. The printer will lay a set of vertical dots for each bit that is set. Lastly, a CR is sent after the last of the graphics data. It moves the paper up a little for the next line.

Well, there you have it, that's all there is to it! If you have a printer manual, then you might try other print densities, etc. Good luck, let us know your results; and but most of all, have fun with it!

Next month, I'll explain why I'm actually using only 4 of the bits in the data for printing and talk a little bit more about the printer.

The last order of buisness is to include my mailing address for the "What do you want?" responces; let me know. Please remember that I will not usually be able to respond directly, but will respeond in the column when appropriate.

J. Scott Kasten, 307 Whispering Oaks Dr Bethalto, IL 62010-1039 or c/o "The OS9 Underground".

```
/* **** Program print.c **** */

#include <stdio.h>

#define WIDTH   640     /* Width of file in pixels. */
#define LENGTH  480     /* Length of file in data lines. */

main (argc, argv)
int argc;
char    *argv[];
{
    FILE    *infp,*outfp;       /* file pointers */
    int     i,j,k,l;            /* position loop indecies */
    int     color;              /* computed pixel color value */
    char    line[WIDTH];        /* array to store current printer line */
```

(Listing continued on page 29)

("print.c" continued on page 24)

```
    switch (argc) {
        default:
        case 1: fprintf (stderr,"\nprint usage:\n\n");
                fprintf (stderr,"print <path>\n\n");
                return 1;
        case 2: if ((infp=fopen(argv[1],"r")) == NULL) {
                    fprintf(stderr,"print: can't open %s\n",argv[1]);
                    return 1;
                }
                if ((outfp=fopen("/p","w")) == NULL) {
                    fprintf(stderr,"print: can't access print device\n");
                    return 1;
                }
        }

/* Set printer line spacing appropriate for graphics. */

    putc(27,outfp);
    putc('3',outfp);
    putc(12,outfp);

/* Main print loop. */

    for (i=0; i<LENGTH; i+=4) {
        for (j=0; j<WIDTH; line[j++]=0);
        for (k=i; (k<LENGTH) && (k < (i+4)); k++) {
            for (j=0; j<WIDTH; j++) {
                color = getc(infp) % 2;
                l= 128 >> (k % 4);
                line[j]+=l*color;
            }
        }

/* Set printer into graphics mode. */

        putc(27,outfp);
        putc('*',outfp);
        putc(6,outfp);
        putc((unsigned char) (WIDTH % 256),outfp);
        putc((unsigned char) (WIDTH/256),outfp);

/* Print out the line. */

        for (j=0; j<WIDTH; putc(line[j++],outfp));
        putc(13,outfp);
    }

    fclose (infp);

/* Reset line spacing to normal. */

    putc(27,outfp);
    putc('3',outfp);
    putc(36,outfp);

    fclose (outfp);
}

/* **** End code. **** */

-JSK----------------------------------
```

<EOF>

G-WINDOWS for the TOMCAT TC70

Introducing G-Windows, a complete Graphical User Interface for the OS-9 Real-Time Multitasking Operating System. G-Windows is totally multitasking (all windows can be active and updated at the same time). It is extremely memory efficient and can be ROMed for embedded system applications.

G-Windows Technical Features

- Operates on the Tomcat TC70 as well as Gespac, VME and OS9000 and soon the QT PC/30.
- Windows are structured as standard OS-9 SCF devices. Created and updated similarly to OS-9 pipes.
- Very efficient memory usage.
- Totally ROMable
- Full graphics and VT100 terminal emulation support in each window.
- Totally multitasking. All windows can be active and updated at the same time. Enables multiple processes to share the same screen.
- Cut and paste function within text windows
- Includes full featured desk top manager
- Proportional and multi-colored fonts supported
- Imports graphics files from Compuserve GIF format
- Enables creation of images
- Tool-box with pop-up menu, event windows, icons and Input/Output buttons and dials
- Graphics library for development of user applications
- Windows can be reduced to waiting or sleeping icons
- Files can be deleted or copied by dragging the icon
- Easy to configure
- Multiple overlapping
- Quick key controls

The G-WINDOWS windowing software goes far beyond anything ever offered for the OS-9 operating system. Its modular structure, multitasking capabilities, and unique way of seamlessly interfacing with the user's application program make it a breakthrough in the emerging technology of graphical user interfaces.

New Products!

MASTERING OS-9,
SELF-STUDY AUDIO TAPE COURSE

This is a complete self-study course designed to help you master the power and flexibility of the OS-9 real-time, multitasking operating system. The package includes 6 professionally narrated audio tapes (5 1/2 hours). Over 270 diagrams and sample listings housed in two large three ring binders. Diskette contains course examples, libraries and utilities. Greatly reduces OS-9 learning curve.

THE KEEPER™
COMING SOON!

The **KEEPER™** is a complete integrated bookkeeping system with POS. It handles basic accounting for the small business. Including invoicing, inventory, mailing list, fax letter, form letters and is integrated with DynaStar and fbu for ease of use. Plus much much more. In use for over a year **THE KEEPER™** will be available shortly.

LOST SCULPTOR FOUND!

WHILE DOING SOME SPRING CLEANING WE FOUND A SMALL SUPPLY OF V1.14 SCULPTOR FOR OS-9/68000. CURRENT PRICE FOR V2.1 IS $2,500.00. WE CAN OFFER THESE FOR:

ONLY $299

WHILE SUPPLY LASTS!

For more information on any of these products and a complete catalog contact:

FRANK HOGG LABORATORY, Inc.
204 WINDEMERE ROAD
SYRACUSE NEW YORK 13205 USA
Tel. 315/469-7364 Fax. 315/469-8537