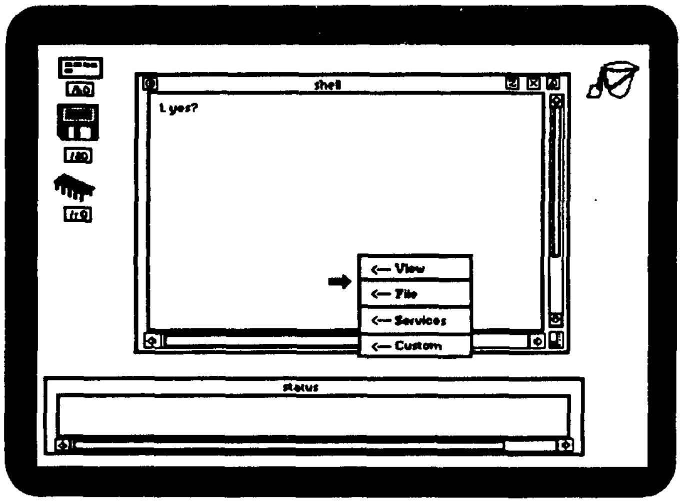
the world of 68' micros

Supporting 68xx and 68x0x computers

01 February 1994 Vol. 1 Number 5 $4.50 Canada, $4.00 US



Take a quick look at G-Windows...

now available for Delmar Systems, KiX, MM/1, and OS-9000! (article begins on page 12)

- *What happened to Puppo board? (4)*
- *Is the MM/1 back? (27)*
- *Use a CoCo for dedicated controller (9)*
- *Spotlight on Mid Iowa & Country Coco Club (7)*
- *Start using Basic09 (21)*
- *Smash! Great new game for OS-9 (25)*
- *much more!*

CONTENTS

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

BULKRATE

the world of 68' micros

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

Editor: F. G. Swygert

Subscriptions:
$23/year (8 issues) US; $30/year for Canada/Mexico ($12 US, $16 C/M for six months- four issues). Overseas $35/ year ($18 for four issues) surface. Add $8/year, $4 six months for air mail.

Advertising Rates:
Personal Classifieds- $0.20/word, $4 minimum. Name, address, and phone number free. FREE for subscribers.
Display Ads- $15 1/6 page, $20 1/4 page, $35 1/2 page, $60 full page, copy ready. Add $10 for special placement, $10 for typesetting ($5 1/4 or less). Dot matrix will be typeset if deemed unacceptable and submitter billed. 10% discount for four or more appearances.

All trademarks/names property of their respective owners.

The publisher welcomes any and all contributions. Published material becomes the property of FARNA Systems unless other, prior arrangements made. Submission constitutes warranty on part of the author that the work is original and not copywritten by another party. All opinions expressed herein are those of the individual writers, not necessarily the publisher or editor. FARNA Systems reserves the right to edit or reject any submitted material without explanation. Renumeration discussed on an individual basis.

Back issues are $4 per copy. Overseas add $1 each for airmail delivery.

Newsstand/bulk orders available. Dealers should contact the publisher for details.

Problems with delivery, change of address, subscriptions, or advertisers should be sent to the publisher with a short description.

The publisher is available for comment via Internet (dsrtfox@Delphi.com) or Delphi E-mail (DSRTFOX). The CoCo and OS-9 SIGs on Delphi are also frequented (The Delphi SIGs are sponsored by Falsoft).

The editor speaks...

F.G. Swygert

I had a dream...

Now as unbelievable as this sounds, this is a true story! I awoke on the morning of 07 December after having an unusual dream. The day before I had actually just finished getting the 15 December issue ready for mailing on the seventh, so maybe that influenced my sleep.

In my dream, I met an engineer who worked for Tandy right outside my home. I didn't know he worked for Tandy, he was visiting someone who lived nearby. We met, and a neighbor told him I was into computers, and we ended up mentioning the CoCo. He said something about it, I don't exactly know what, and I got excited (not angry, just excited!) and told him he hadn't seen a CoCo until he saw one with ADOS... the way Tandy SHOULD have done it.

I took him inside and showed him my system. He was impressed with the ADOS boot-up 80 column black-on-light blue screen (my default) and the on-screen clock (I use a Disto SCII with 4-n-1 board). He liked the Tandy 2000 case also, and the hard drive. In fact, he was enraptured by the whole thing! I then told him about the aborted Burke & Burke "Rocket" (mentioned that it would have made a great Tandy option...), and that Tandy should have at least supported the refresh rate and sockets for 256Kx4 bit DRAMs for 512K, and then added a board for the 2MB that the GIME would actually support, either using 1MBx 1 bit DRAMS or even SIMMs. Trivial changes that could be made with another GIME run only!

Well, by this time he was totally engrossed with my system and ideas, and started mumbling something about a new motherboard with built-in floppy controller, add-on IDE board, four SIMM sockets in two banks, 68008 socket instead of Rocket board, reuse old 1000HX case... then I woke up.

We all know something like this will never happen, but it sure was a nice dream! Just pop in a 68008 and flip a switch for an instant upgrade to OSK, start off with 512K (two 256K SIMMs), add up to 2MB for the CoCo (two 1MB SIMMs), 16MB for the 68008 later (4MB SIMMs in all four SIMM sockets). I suppose the IDE board would be similar to the 1000HX upgrade boards, and I guess they would probably keep the game port also. To bad I woke up before it went into production, I'd like to know how much it would have sold for!

There was, of course, another influence on that dream. I recently received a call from someone about the "under $1000 OSK system" I proposed in the last issue. It seems this person was working on a low-cost system also, to include OSK and a GUI (graphical user interface)/windowing system. This sounded a little better than my proposal, at near the same price. He intended to accomplish this by making a new, highly integrated motherboard. Most importantly, he had a license to the necessary parts of OS-9/68K and had written most of the remaining utilities and drivers. I hadn't received enough feedback on my low-cost proposal yet (it was on telecom networks about a month before printing in the last issue), so I decided to drop marketing on my proposal (I will still print an article about it!). Keep in touch for more news as it develops. This may be the best hobby system yet... IF it goes into production! If not, it's back to the PT68K2 motherboards.

I do hope that everyone had a happy holiday season! I sure did. I took a much needed two week vacation over the Christmas/New Years holiday. That, coupled with the fact that this is the first issue being sent third class, probably made this issue about a week late. I'll get future issues out more on time! The vacation is also why some orders for other FARNA Systems items were a little longer in being delivered. When I got back, I had 6" of mail stuffed in my Post Office box! It took about a week to catch up with that, then be able to finish this issue. Good thing I planned ahead and started before I left on vacation!

< 268'm >

If you just want OS-9...

GET DOWN

TO THE UNDERGROUND!

The "International" OS9 Underground Magazine

$18.00/Year (12 Issues) U.S.
($23. Canada, $27. overseas)
4650 Cahuenga Blvd., Ste 7
Toluca Lake, CA 91602
(818) 761-4135
Write or call for Info

BlackHawk Enterprises can now make the MM/1 *available to the public!*

We have 50 systems ready at the factory, and need cash to pay for them. In order to get the required financing, the bankers want to see verified orders. All we need from you is a written order and a $10 deposit (counts against the purchase price of any system) to prove to the bankers that you are seriously interested in this machine. In appreciation of your gracious assistance, we are cutting prices! Systems will be available 60 days after we receive financing. This offer is limited to the first 20 orders *ONLY!* Offer expires March 31, 1994. Contingent on financing.

Get your order in ***today!***

Help us get this fantastic machine back on the market!

The Developers System

MM/1 Extended with 1 Meg memory, floppy drive, and Case and power supply. Includes OSK 2.4 with C & Basic, Microware documentation, and all available K-Windows documentation.

Suggested Retail Price	$1125
Introductory price	$975
You save	***$150***

The Professional System

A Developers system, minus only the Microware documentation!

Suggested Retail Price	$1025
Introductory Price	$875
You save	***$150***

The Extended Kit

MM/1 Extended board set - no floppy drive, case, or power supply. Includes OSK 2.4 with C and Basic, all available K-Windows documentation.

Suggested Retail Price	$900
Introductory Price	$825

BlackHawk Enterprises, Inc.
P.O. Box 10552
Enid, OK 73706-0552
Phone 405-234-2347

Letters to the Editor

New CoCo3 User!

My daughter was given a Tandy Color Computer 128K. We have been unable to find software for this computer. I am aware that this is an old computer, but if you have any software available or information on how I may order by mail I would appreciate your assistance.

Deldena Graham
1202 Macree Terrace
Florence, SC 29505

I wrote Mrs. Graham and sent her a copy of "68' micros". How many people do you know in this situation? Give them the address and I'll send them a free issue also. The CoCo may be "old", but it is certainly still supported!

The Editors Did It!

I am writing to clarify any confusion that may have emerged due to your response to Robert Gault's letter to Joel Hegberg which was published in the 15 DEC issue. As Mr. Gault notes, the use of mouse signals in Basic09 code was demonstrated by Dale Puckett in the JUL 88 "Rainbow". That article provides an excellent background for understanding the techniques demonstrated in Joel's column (with my code) for the simultaneous processing of keyboard and mouse signals. The omission of a citation of Dale Puckett's article in your September issue, however, was not my fault; indeed, mention of it was made in the original material submitted to "68' micros" via Joel Hegberg.

the missing text:

"... My introduction to these techniques was provided by Dale Puckett ... in KISSable OS-9 in the July 1988 Rainbow."

Ted Jaeger

Ted, I apologize sincerely for tarnishing your image! Joel did state that he accidently deleted the acknowledgment during his editing. No harm was intended... I should have waited to print a reply, so the fault is mainly mine! Maybe this will teach me to follow up BEFORE trying to answer in the future. Again, my apologies!

Holiday Wishes... Good News for Vendors!

I'd like to wish you and your family the best of Yule Cheer, and a Happy New Year too! You did a lot this year: through your tireless efforts (and that of your staff and other writers), ripples of joy have spread from Georgia across CoCoLand.

Based in large part on your reviews, I've been in touch with your advertising vendors, placing orders with Disto, Chris Dekker, CoNect, Northern Exposure, and Rick's Enterprises. About $650 worth of soft and hard ware. Still looking for a comprehensive general database manager, maybe Elite File. I found your brief reviews to be more help than many of the simpy, inane endorsements in the Rainbow. What can I say?

Henry Harwell
2110 West Roma Ave.
Phoenix, AZ 85015-4445

Thanks Henry! I hope the vendors read this, it sure does speak well of the CoCo community... and proves that they DO still buy!

Marty Goodman CoCo Book???

I really enjoyed "Tandy's Little Wonder", and look forward to getting "the world of 68' micros" every six weeks.

I have a few questions: What publications could I get to master DECB? Does Dr. Goodman have any plans for a compilation book on his articles on the CoCo? Are there any publications on EPROMs, their burning, compatible EPROMs to the 6809, and applications< Where can I get schematics for a Korean CoCo (26-3134A) and the FD502 disk controller?

Dan Concepcion
1639 East Park, J-4
Valdosta, GA 31602

Dan, the FD-502 is essentially the same as the FD-501 controller (as printed in TLW). The main difference is the little interference suppresion coils used on the controller to drive lines in the 502. The major differences in the Korean CoCo 2A and the American CoCo is that only one 16K ROM is used and memory consists of two four-bit chips rather than eight one-bit DRAMS. The Korean made CoCo 2B used a MC6847T1 instead of the MC6847. The T1 can be used in older models, but the standard 6847 can't be used in newer models. Why? Three support chips (IC14,15,16 in CoCo 2A schematic... 74LS623, 74LS273, and 555 timer, respectively) were eliminated from the board! Apparently, the T1 revision didn't need these chips, but their presence doesn't affect operations. If you really need schematics for the Korean models, send me $3 each to cover reproducing and shipping. In all respects except as above, the CoCo 2 models are essentially the same.

I'll work on an EPROM article for a future issue. The idea of Dr. Goodman printing a compilation of his articles is a good one! I don't know, however, if he has the rights to reprint articles that were submitted to "Rainbow", which might be a problem. I'll be in touch with him on that subject!

Classified Ad Question

My ad for selling or trading software was omitted in your DEC 93 issue. Several people have requested the list but of yet have not made offers. What is required to keep the ad running?

John R. Mott, Jr.
9822 North 15th Street, Apt. B
Phoenix, AZ 85020-1810

John, you did all you needed to do, drop me a postcard saying you wanted it to continue! I usually run classifieds in two issues unless I here from the advertiser. Unfortunately, I usually don't, whether something sold or not. I'll include your ad in two more issues.

Good Comments In General...

Love your Pub.! I was happy to see your "demographics breakdown" by percentages. I suspected the Canadians were up there, but Phoenix AZ was a surprise.

I'm 72 next March, and "highlight" the text which I may need to refer back to, so I must commend you on the quality of the paper used... so far no highlight bleed-through has ruined a single page. Your three column print size is small but the pape "whiteness" seems to be clear and legible enou even for my old eyes.

Dr. Marty Goodman's "Hardware Hacker" is a very welcome column. The December article covering CoCo 3 service and repair is more to my liking. A lot of good usable info. But, I beat his advice long ago and have two CoCo 3 512K models. Now, how about a 3.5" disk upgrade article?

Rick Ulland's "Beginning OS-9" cleared several points for me. I'm looking forward to the next installment on Multi-Vue. Your "Repackaging" installments are very interesting too. As you can tell, my background is electronics and hardware.

These are only a few things I like about "^8' micros". There are others but I've no more room on this card. By the way, where did Ken-Ton and RGB-DOS go? Keep up the good work!

Henry J. Watkins (W7ZWJ)
429 South 750 East
Clearfield, UT 84015

Would you believe Henry put all this on a 3"x5" index card? And I shortened it a bit to boot! Henry, I could print a point size smaller than I do and doubt you would have trouble reading (but I won't... not everyone is so blessed)! I'll work on finding Ken-Ton's address and doing an article on using 3.5" drives.

What Happened to Puppo Keyboard Interface Project Board?

Buzz Jones
Seattle, WA

This question came as a telephone call which my wife answered. All Buzz asked is that I print something about this project. Well, the offer of a PC board was a bit premature. The person working on it was unable to complete due to business reasons, other projects had to be taken care of first, and an unexpected problem cropped up which meant a total redesign of a money making project, so the Puppo board had to be shelved. I did hear from another individual who was working on a board. In fact, I sent him a copy of the schematic (he had heard that I printed it and called to inquire) and asked that he write back as soon as he had a working board design.

I apologize for getting so many people's hopes up. The schematic was printed mainly to help those with the adapters in repairs. If anyone out there DOES design a new board, please let me know! I'll be happy to print an article and let everyone know where they can order bare boards (and possibly kits?) from. If I can be of assistance, just call or write.

Letters are printed on a space available and popular subject matter basis. If you don't want your letter printed, or wish to withhold your address or name and address, please state so when writing. In some cases, letters are slightly edited for space and/or clarity. If a personal reply is desired, please enclose an SASE.

SK*DOS/68K (and 6809)

Peter Stark

An alternate, mature disk operating system for 680x0 (and 6809) based computers.

Overview

SK*DOS/68K is a single-user disk operating system for computers using Motorola 32-bit CPUs such as the 68008, 68000, 68010, and 68020. It is ideal for applications in industrial control, scientific computing, turnkey systems, and just plain hobbyist use.

It provides the power of a full DOS, yet is simple and easy to use, and will run on systems from 32K to 16 megabytes and more. Because SK*DOS is easily implemented on a new system, we call it "The Generic DOS" which allows programs written for one system to be run on many others.

SK*DOS/68K is designed to make future extensions easy. The structure of SK*DOS/68K is very similar to that used in our 6809 SK*DOS. It uses the same disk format, and so can read and write 6809 SK*DOS (and therefore also 6809 Flex) disks. Likewise, its appearance is the same, even the interface to user programs is very similar. Obviously, a 68000 DOS must be different from one designed for the 6809, but the differences have been handled in such a way that a user familiar with SK*DOS 6809 (or Flex) will feel right at home with SK*DOS/68K.

Design Criteria

We obviously would like SK*DOS/68K to become widely used. For this reason we designed it with three criteria in mind. First, SK*DOS/68K should be affordable, so that even users of small or unpopular systems could have SK*DOS/68K for their systems. Second, SK*DOS/68K should be adaptable to a wide variety of systems, so it provides a common disk operating system which allows a program developed on one system to be run on many others. Finally, SK*DOS/68K should be simple to use. Although SK*DOS/68K contains quite a variety of advanced features, these are totally transparent to a beginning user, and are not necessary for actual use. They are there for those who want them.

System Compatibility

SK*DOS/68K can be used in systems from as small as 32K up to 16 megabytes or more. It is available for a variety of systems, such as the following:

PT-68K-1 68008, PT-68K-2 68000 from Peripheral Technology (the latter also used in the Radio-Electronics Magazine series starting in October 1987)

Mustang-08 from Data-Comp Inc.

ESB-I 68008 from Emerald Computers Inc

M7950 68000 from NCR

GRIFON-08 68008, GRIFON-20 68020 from Ralph Allen Engineering

Microbox 3 68000 from Micro Concepts,

68000 from VCS Computer Systeme GmBh

68000 from Dominic Evans Industries,

...and others. Some of these versions are available from us, others are supplied by the equipment manufacturers.

There is also a 'generic' version which is user-configurable for other computers. This version includes a Configuration Manual, which describes how to implement SK*DOS/68K on a new system. It comes with a disk with source code for the terminal and disk drivers, boot program, disk format utility, and other valuable information. Builders of small systems will be especially interested in the source code for 68K HUMBUG, a boot ROM which provides for system start-up and debugging while adapting SK*DOS for new systems. The disk is available in either SK*DOS format (which can be read by 6809 SK*DOS or 6809 Flex), in Radio Shack Color Computer format, or in IBM PC format.

Some Technical Data

Our aim in writing SK*DOS has been to develop a powerful DOS, but one which is simple to use. Many users have told us to "Keep It Simple, Star-K!" That is obviously not totally possible with a powerful CPU like the 68xxx, but SK*DOS/68K is set up so that it defaults to a very simple system. Unless the user specifically calls its more advanced features, it is:

1. Single-user and single-tasking
2. Non-relocatable. It always occupies a fixed location in memory.
3. Disk format is upward compatible with 6809 SK*DOS. 68K SK*DOS can read 6809 SK*DOS disks, and vice versa, both for text and most binary files, but there are two exceptions: 6809 SK*DOS cannot read those 68xxx binary files with load or transfer addresses of 64K or above, and it can only read those 68xxx files which are in the root directory of the 68K disk.
4. File Control Block structure is very similar to that of 6809 SK*DOS.
5. 68K SK*DOS functions are very similar to those of 6809 SK*DOS, and current 6809 SK*DOS (or Flex) users will find it an easy transition to the 68K SK*DOS.
6. Because 68K SK*DOS is used very much like 6809 SK*DOS, many existing programs can be translated on an almost line by line basis.
7. User-written device drivers are easily installed to provide customization to different terminals or other I/O devices.

To run SK*DOS/68K, a computer requires a minimum of 32K of RAM, with 128K or more being preferred. But the old maxim of 'the more, the better' certainly holds.

Although SK*DOS is not relocatable, we can provide it at a variety of different memory addresses on special order (and it is therefore also potentially ROMable). Since most 68000 computers will have RAM beginning at address $0000, the most likely memory configuration will look as follows:

$0000 - 03FC	Trap & interrupt vectors
$0400 - 0FFF	Boot routines & stacks
$1000 - 5FFF (approx)	SK*DOS/68K
$6000 to end of memory	available to user programs

As indicated above, this is just one possible memory configuration, but the most likely one. Different configurations might be possible for systems which do not have available memory beginning at $0000.

All memory above SK*DOS/68K is available for user programs and utilities. It is possible, however, to load some programs into memory and leave them there. For example, SK*DOS/68K is supplied with a RAM disk program which can reserve from 16K to 1MB of memory as a RAM disk. This program and its data area would normally be loaded once and then stay in memory above SK*DOS/68K; SK*DOS/68K maintains memory pointers which define the bottom and top of free memory.

Unlike SK*DOS/68K itself, disk-resident commands (such as CAT or LIST), as well as user-written programs, are written in position independent code. When executed, these are loaded by SK*DOS/68K into the free space above SK*DOS/68K itself (and above any memory-resident programs that may already have been previously loaded.) Hence such user programs and utilities will run in any SK*DOS/68K system, even one where SK*DOS/68K has been located at some other address.

Since SK*DOS/68K may not lie in the same place on all systems, user programs must be position independent. Moreover, all calls to SK*DOS/68K, as well as all references to SK*DOS/68K variables, must also be position independent. Hence all such calls are through exception vectors, and all references to SK*DOS/68K variables are through relative addressing.

DOS Functions and Commands

SK*DOS/68K is provided with a full complement of utilities, including those needed to format, copy, backup, and test disks; display disk contents; build, list, append, rename, or delete files; and change system parameters. This includes the following commands:

ACAT alpha sort directory
APPEND two files into third
ASM 68000/10 assembler
BACKUP a disk
BEEP at command completion
BUILD a text file
CACHE disk cache program
CAT display disk data
CHECKSUM a disk's content
COMPARE two disks
COPY files
DAMON monitor disk accesses
DELETE a file/files
DEVICE install a device
DIR display directory
DISKNAME display/change disk name and date
DOSPARAM display/change DOS parameters
DRIVE install a drive
EDLIN text line editor
ECHO to the display
FIND a text string
FORMAT a floppy disk
FROMSDOS read MS-DOS disk
FTOH for single/drive backup
GET load binary into memory
GETX similar to GET
HELP help utility
HTOF use with FTOH
LINK set up boot disk
LIST contents of a disk file
LOCATE file load addresses
MAKEMPTY make an empty file
MON return to boot ROM
NOBEEP cancel BEEP
PDELETE prompted file delete
PEEK into memory
POKE into memory
PROMPT change prompt
PROTECT write/delete protect
RAMDISK initialize program
REDOFREE relink a disk
RENAME a file
RESET trap vectors
S1TOCOM convert S1 format
SAVE memory to disk
SCAT sequence disk directory
SEQUENCE number change
SK*DOS09 emulate 6809 SK*DOS
STEPRATE display/change drive rate
SYSTEM select system drive
TCAT disk directory by time/date
TOMSDOS write MS-DOS disk
TRACE start program tracing
UBASIC Basic language
UNDELETE a deleted file
VERSION display file version
WORK select work drive
XEQ repeat previous program

Additional commands such as TIME, HDFORMAT, and PARK are present in systems having clock / calendar chips or hard disks.

Some of these functions are quite useful, and available with some operating systems only at extra cost. For example, SK*DOS is supplied with a line editor, a full 6800/68010 assembler, and a (somewhat limited) Basic language interpreter, which allow you to immediately get started learning the power of SK*DOS. Other programs and utilities are supplied to make SK*DOS more convenient or to speed up its operation. The RAMDISK command allows you to set up a RAM disk up to 1 megabyte in size, which behaves like any other drive; the CACHE command sets up a disk cache which tremendously speeds up access in floppy - based systems.

A rather unique command is SK*DOS09. This is a complete 6809 SK*DOS and a 6809 emulator program. It allows you to run standard 6809 programs on your 68K system (although they do run at reduced speed). For those users who have a substantial base of 6809 software, this allows them to run such popular programs as 6809 editors, spreadsheets, assemblers and cross-assemblers, word processors, compilers, and more.

Like its 6809 predecessor, SK*DOS/68K contains program-callable functions to do file maintenance; open, read, write, and close sequential and random files; read and write individual sectors; access the disk directory; input and output characters, strings, and numbers; process command line arguments; parse file names; report errors, and more.

SK*DOS/68K has also been designed to make expansion easy. A standardized system of calling disk and terminal drivers makes it possible to add additional device drivers without great complexity. A system can contain as little as one drive (plus perhaps a RAM disk), or as much as ten physical disk drives. Furthermore, the DEVICE and DRIVE commands allow devices and drives to be reassigned as needed, so additional devices and drives can be brought in as needed.

SK*DOS/68K supports I/O redirection from the command line, using the < and > symbols as in Unix or MS-DOS, to either disk files or I/O devices. It also supports batch files, and batch files can call other batch files. In addition, application programs can pass a command line to SK*DOS as if it came from the console and have it executed. Combined with the fact that it is easy to reserve memory for resident programs, this makes it easy to add an alternate command processor or shell which totally changes the user interface.

Files written to disk normally contain the file creation or last modification date. In addition, each file entry may also contain a creation or modification time if the system contains a clock/calendar IC. If such hardware is not available, then SK*DOS/68K defaults to sequentially numbering files each day, and records the sequence number rather than the time. Disk catalog utilities are provided which list disk contents either in the order on the disk, in alphabetic order, in order by date and time, or in order by date and sequence number; the latter two show the latest files first.

Software Availability

Software availability for a new DOS is always a problem, especially for a DOS which is not in the mainstream of computing (i.e., is not PC-compatible) In this respect, SK*DOS has some advantages over other systems.

SK*DOS/68K was designed as an upward path for industrial users and hobbyists who were using the 6809. Although it contains many extensions beyond the 6809 world (and is very different in those areas where staying with 6809 conventions would have limited future progress), nevertheless it contains enough similarities that 6809 users find it relatively easy to modify their 6809 software and move it to the 68000, especially since the 68000 CPU has many instructions similar to those of the 6809. (We even have a code converter which translates 6809 assembly language to 68K assembly code, though not perfectly.)

This, along with the availability of several fairly inexpensive 68K systems, has resulted in a number of 6809 programs being converted to SK*DOS/68K. Some of these are:

Small C compiler

CMODEM, a modem program with file upload and download facilities for XMODEM protocol.

NRO, a text processor for formatting and printing text prepared by an editor.

RASM, a relocating assembler and linker

RBASIC, a full Basic interpreter

Spell'N Fix, a spelling checker modified from 6809 code.

Two full C compilers

... and many more available now.

Several disks of public domain software are available for $5 each from the SK*DOS Users' Group in Georgia. User support is also provided around the clock through the Star-K BBS at 914-241-3307. Set your system from 300 to 2400 baud, 8 bits, no parity, one stop bit. You may use this BBS to contact with other users, obtain information or help, upload or down load programs, and more.

continued on page 8

Any and all CoCo/68K/OS-9 clubs are invited to send in an article describing their operations. The club needn't be a large one, just one that either meets regularly or puts out a regular publication. Let everyone know you are out there and what you do!

CLUB SPOTLIGHT ON:

Mid Iowa & Country CoCo

Terry Simons

From the beginning we were more than a Des Moines club. Stretching to the surrounding areas, we began with the name Mid Iowa CoCo, and within a year were living up to it with members over 100 miles away. While the meetings gave the common thread to draw members; what ever night we chose was always inconvenient to some. By the end of the first year I started the UPGRADE newsletter. I enjoyed doing it, and it eliminated any legitimate excuse for not joining. Using a newsletter made it possible to reach out to those who couldn't attend meets.

For the next few years we sped along, like most CoCo groups we picked up members as the market grew. In the early years I also dealt in CoCo hardware (TDP Products) and software. The slight discounts given by software dealers made selling more of a favor than a business proposition. But this and my own attempts to market "Home-Pac" (a home financial organizer) left open another doorway to be used later.

Problems do come up. Ours were quickly resolved by letting all officers vote for the good of the group. We are more concerned for the group as a whole rather than democratic rules. All members including myself and the other officers pay dues; seems to hold down elitism. At one point, I was trying to do too much, and was about to hit burn out. I stepped back, took a look, and came back to do what I could comfortably handle. Someone else would do the rest or else... People like Gene Lund, Glen Frizel, John Schoettmer stepped in to take over the local meet and library. And did it superbly.

In the fourth year market saturation of the CoCo was taking place, and membership was slowing. I wanted to change that, but how? Remember that doorway? My CoCo business was more of a hobby anyway, so I recruited some of my out of state friends. They helped recruit a few more. By our fifth year, our out of state membership was such that we had to have a name change.

What to name the group though? We liked the informality of our Mid Iowa group so we kept that part and added "& Country". Unfortunately many just see the Mid Iowa, and overlook the "& Country", and are surprised to learn that we have quite a large membership nationally.

By this time OS-9 interest was growing. James Jones, a Microware programmer and nationally know on telecommunications networks and at 'Fests, stepped in to help as OS-9 coordinator. James (JEJONES on Delphi) is a wonderful person who will step in to help OS-9ers at all levels.

Meanwhile, a member in Florida sent me a cute letter on disk, graphics and all (it's in our UPGRADE #1), which gave me the idea of putting our UPGRADE newsletter on disk. There are pros and cons to disk VS hardcopy. Hardcopy is nice to lay down/ pick up/ leaf through, etc., more easily. You cannot, however, get the animated full color graphics or short programs (without typing). Besides, the UPGRADE will dump to hardcopy on your printer. But the bottom line, I think, was that a disk magazine is much easier (to produce and copy). With volunteer time at a premium, and many clubs loosing their newsletters due to the lack of same, the UPGRADE on Disk has become a permanent but reliable feature.

For the last three (or is it four now?) years we have been building a national following. We were the first club with a paid ad in Rainbow, and there were several after. We managed the Middle America CoCo Fest in '92. Though light in attendance by all accounts, the many that did come said it was as nice as any. There was a plush floor, curtains and draped booths, satisfied vendors and attendees, and... the nicest Fest "T" shirts ever!

How about group services, especially to out of state members? We like to think our UPGRADE Newsletter is nice, but if a user group is no more than a newsletter, you might as well subscribe to one and forget it. From the inception of offering out of state memberships, like out of town memberships, we fully expected to make the club available to "every member". Or at least as much as is practical.

A real Library

Most CoCo groups through the years have had a library. But, if you can't travel to a meet or the librarians house, well, it's not available to you. Many joining are surprised to find ours is! A full list is given to each member, and the price is the same for all. A library? How many disks? Well, that could easily be an ocean of six to eight hundred disks with a brief DIR listing and many duplicate or similar programs. I mean, how many disks of disk utilities do you want? Two that have the best performance with a couple alternatives is what most supporters want. So, two to three disks of the best is all we offer. That eliminates the hodge podge in an endless list, and allows you to know when ordering a MI&CC Library disk that you're getting something worthwhile. We consider our 120 to 150 disk library quite large. And it is when you consider it's the "select best". Disks are categorized so you can quickly select what you want. Categories include Christian, CoCo 2, CoCo 3, OS-9, Applications, HAM Radio, etc.

Many CoCo dealers as well as users find themselves very surprised to learn that MI&CC is not, nor never has been a pirate organization. We just didn't think it belonged as a part of a group that promotes the CoCo. It would have been a poor thank you to Radio Shack and Rainbow for referring us members. Neither are we to dictate standards in a users home. To say more could be a 20 page article, as the arguments are endless, and go nowhere. Perhaps this is why we unashamedly present "Orphanware", another pleasant surprise.

Orphaneware: software that was once commercially available, but can no longer be purchased new and is not currently supported by any distributor or author. Authors have even said to me

"Once we quit marketing it, I considered it Public Domain". Others take the view of the dog who doesn't want his bone, so buries it, and then growls when anyone comes near. In either case, for certain it can not be sold. Can you pass it out for free? Well, it's kinda' like this; if a product isn't made available for sale, it can not generate profit; thus no profit can be lost. While the legal beavers take the "dog and the bone" position, we at MI&CC don't. The problem is that some users do still want some non-available software, and would be willing to buy it in most cases but can't. It's of course the dealers prerogative to go on to better dollars elsewhere or stop marketing if enough profit is not coming in. We just don't think this is sufficient reason to deny the availability of good software. Consequently, members see some (once) pretty big name software offered at the same backup/copy fee as many Public Domain disks. A side note to this is the fact that dealers seeing this are more often selling rights to other dealers thus keeping it available. That's all we really wanted in the first place. Any software vendor, author, or dealer with a legitimate complaint can let us know the software is being marketed or otherwise made available, and we'll take it off our "orphanware" listing.

At times we have quite a listing of CoCo hardware by members wanting to sell and we have members looking to buy. RS-232 Pac's for $25, a spare CoCo, a Multi-Pac, used drives or hard to find books, and more. We've sold 'em all and will have 'em again. Need a ROM burned? Want to use the backside of your 502? Assuming you already own Disk BASIC, as an MI&CC member you can pick up a modified DECB 1.1. ROM and have 40 track, double sided, 6 ms. disk access on start up.

Our Christian sub-chapter was formed to find and provide Christian oriented software. Actually there is very little, but what there is, we offer. The reality is that most any software can be used for Christian purpose, it's just not labeled specific for that use.

A lot of our success comes from the fact that we just won't adhere to the all DECB or OS-9 only thinking. We intend to provide a full range, and let the user decide what they can use.

How do you have a popular club? The key here begins in our first year. We created a newsletter to serve more. We created a library to serve more. Do you get the picture? The wider the variety of services you can provide, the wider your group will spread. And it eliminates elitism, the demise of many groups. Next comes "who's gonna do it"? And they must live within at least local phone distance. Most begin by donating their time, and wind up paying a lot of small stuff out of their pocket. In MI&CC we have no qualms about reimbursing these petty expenses that both add up and get old. Gosh, the time is enough, and you need these people, so take good care of them.

There you have it folks, work, consideration for all, and people who can work together. It does have it's rewards. The friendships I've gained are far more valuable than all the technology offered.

< 268'm >

SK*DOS... *continued from page 6*

Price

SK*DOS/68K is available for single-copy or dealer sales, as well as OEM licensing. Single copies cost $75, which includes the line editor, 68000 assembler, UBASIC, and all of the utilities listed above, as well as free updates. The 6809 version (will run on a CoCo- mention CoCo when ordering) is $39.95. User support is provided through the Star-K computerized BBS system, accessible to all registered users.

A Configuration Manual is available for users who wish to adapt SK*DOS to run on a new system. It includes source code for HUMBUG, a ROM monitor program for booting the system, as well as source code for SK*DOS BIOS, FORMAT, and other useful programs. The code is provided on a disk in either SK*DOS, Flex, or MS-DOS format. The Configuration Manual costs $100 alone, but is offered at a special price of $25 to SK*DOS purchasers. SK*DOS adaptation may be done on 68K system running SK*DOS, a PC/clone, or a 6809 system running SK*DOS, STAR-DOS+, or Flex.

< 268'm >

Mid Iowa &
Country CoCo Club
c/o Terry Simons
1328 48th
Des Moines, IA 50311

Do you want support tomorrow?
Stay with a group that keeps you in touch!

Annual Membership:
$16 US, $21 Canada

The "Upgrade" disk based newsletter requires a CoCo 3 with 80 column capable monitor (RGB preferred for best viewing of graphics) and at least one 40 track capable (FD-501 or third party drive) disk drive. Terry Simons is the treasurer and newsletter editor for the club. His phone number is 515-279-2576 (after 8 PM CST). The newsletter is published monthly from approximately September through May, once during the summer (9-10 issues per year).

EDTASM6309 Version 2.02

This is a major patch to Tandy's Disk EDTASM and offers many improvements over the original version and EDTASM6309 V.1.0. The program suppoorts all CoCo models. The CoCo 3 version uses an 80 column screen and runs in 2MHz mode. YOU MUST ALREADY OWN TANDY'S DISK EDTASM TO MAKE USE OF THIS PRODUCT. It WILL NOT work with a disk patched cartridge EDTASM.

Robert Gault
832 N. Renaud
Grosse Pointe Woods, MI 48236
313-881-0335
$35 + $4 shipping & handling

Some Changes to EDTASM:
1) Tape no longer supported
2) Buffer increased to over 43K bytes
3) Enter V command directly from Editor & ZBUG
4) Multiple FCB & FDB data per line
5) FCS supported
6) SET command works properly
7) Screen color same as that set in Basic
8) Symbol table printed five per line (cc3 only)
9) Actual errors printed with /NL option
10) Warning on long branch where short possible
11) ZBUG defaults to numeric mode
12) Supports all RGBDOS drive numbers
13) Latest 6309 opcodes supported for assembly and disassembly.
14) Auto detects 6309, traps illegal opcodes
many more improvements!

The Hardware Hacker

Dr. Marty Goodman

Interfacing a CoCo to the Real World

I've often found the Color Computer to be an especially desirable device for use as a "dedicated controller", doing one or another "real world" task. This because the CoCo is physically small, quiet, uses little power, is highly reliable, and, though no longer in production, still relatively cheaply available via BBS's, garage, and flea market sales. A CoCo 2, for example, is commonly available for $7 to $15 in my area at local thrift shops. A 64K CoCo 2 with a ROM based, auto-executing program and some I/O lines added makes an excellent "dedicated controller". Those of you with full blown CoCo systems, including disk drives, editors, assemblers, and debuggers have all the tools needed for development of software or firmware for such dedicated applications.

Because the CoCo no longer is in production, it's not really suitable for any application that is to be mass produced. However, a CoCo is perfect for those one or two of a kind applications, such as a personal alarm / control system, amateur radio tasks, darkroom applications, etc. In many cases the ROM-Basic in the CoCo is all you need for programing your operation. Even in cases where you intend to go into mass production, one can sometimes develop a system on the CoCo, then put together a simple system using a 68B09E, a 32K static RAM, a 32K ROM chip, and a couple of PIA chips (using a single GAL or PAL chip for address decoding and "glue") that will act just LIKE a CoCo without a monitor and without some of the memory mode switching options. In fact, a micro controller is available that essentially IS a 32K CoCo (32K DRAM and 32K ROM) without the ROM ($60 in sample quantity from J&M Microtek, Inc., 83 Seaman Road, West Orange, NJ 07052)

There are some extremely simple real world control applications where you can use a CoCo with virtually NO modifications or substantial additions. An excellent example is the case of a chap who recently called me for help in using his CoCo to control the light on his enlarger. He was comfortable writing the required program in BASIC that would calculate the required exposure for various input parameters, and that would make an appropriate time delay. But he needed a means of controlling the 110 volt AC-powered lamp in the enlarger. I explained to him how he could use the cassette motor relay in the CoCo to control an external relay that was rated at sufficient voltage and power as to be capable of controlling the enlarger bulb. All he needed to do was use the cassette relay contacts (pins numbers 1 and 3 of the five pin DIN cassette connector... the two pins immediately on either side of the "notch" one sees at 12 o'clock on the cassette port connector) as a switch that turns on or off a 12 volt DC supply (that one can get from a wall transformer) that in turn powers the coil of one of a number of relays that can be purchased at Radio Shack whose contacts are rated for handling 110 volts AC. It is helpful to put a diode (such as a 1N4001, 1N4002, or 1N4004) across the contacts of the coil of the external relay, with the cathode of the diode hooked to the side of the coil that is connected to positive voltage via the cassette relay switch. This gets rid of sparking in the CoCo's cassette relay caused by "back EMF" from the external relay's coil when the circuit is broken.

Other real world applications using an unmodified CoCo involve using the zero crossing detector of the cassette port, or the four-way multiplexed 6 bit A to D converter that is part of the joystick circuitry. A glance at the schematic for the CoCo, a quick read of the service manual, and a look at the assembly language code used by BASIC to read the joystick will give you an idea of how that A to D converter is used. Note that this converter is "software driven"... you need IN SOFTWARE to guess at the conversion, check a comparator to see whether your guess is more or less than the actual value, frame another guess based on that, and check again until you have your six bit accuracy. The WEFAX program for the CoCo (still available through me for $10) allows the CoCo to process the audio from shortwave WEFAX transmissions and receive a satellite image picture or synoptic chart, using almost no external hardware. It employs the zero crossing detector of the CoCo, in conjunction with some VERY sophisticated, real time assembly language code for turning zero crossings into analysis in real time of the frequency of a signal.

What if you have a more complex input/output task, where you need many lines for sending data to and from the CoCo? In those cases, I use one of two I/O cards that I made up for my CoCo. These cards have on them two 68B21 PIA chips and support 32 lines of I/O. Due to the nature of the 6821 PIA chip, ANY individual line among those 32 can be made, under software control, into either an input or an output. If all you need are 16 lines of general purpose I/O, you can make a VERY simple I/O card, that uses ONLY a single PIA chip, and plugs into the CoCo's 40 pin bus. With such a card, you need a Multipak Interface to do development work on a disk-based system, but the card can plug right into any target dedicated CoCo system, or be combined with another card that has the ROM software your task requires with a Y cable.

The 6821 PIA chip was specifically designed to be especially easy to connect to Motorola CPU systems. In the case of a simple 16 line I/O card for the CoCo, the 6821 is hooked to the CoCo's bus as follows: D0 thru D7 lines are hooked to the data lines on the CoCo bus. Reset, R/*W, RS0, RS1, and E of the PIA are hooked up to Reset, R/*W, A0, A1, and the E clock of the CoCo. CS0 and CS1 are tied high (connected to +5 volts via a 4.7K resistor). *CS2 is connected to the *SCS line of the CoCo. Ground and Vcc of the PIA are hooked to ground and +5 volts.

If your application does not require interrupts, that's all there is to interfacing a 6821 PIA to the Color Computer system bus. At this point, all you need do is use the PA0 - PA7 and PB0 - PB7 lines as general purpose I/O. The Motorola data sheet for the 6821 has all the information you need to do this. I'll give a simple example:

Before you can use the general purpose I/O lines of a 6821, you need to PROGRAM the 6821 chip so as to define the lines as inputs or outputs. If you don't do that, the lines will be (at power up or after a reset pulse) all INPUTS. To program a port, you first write to the control register for that port a "0" into bit number 2. This turns the I/O port into a "data direction register". You THEN write a byte into the I/O port in question (which at the moment is a data direction register, actually) such that you put a "1" into every bit that you want to program to be an OUTPUT, and a "0" into every bit you want to program to be an INPUT. After programing the data direction register, you then write a "1" into bit 2 (the third bit, remember, for the bits are numbered 0,1,2, thru 7) of the control register to turn the main PIA port from being a data direction-setting register into a general purpose I/O register.

Let's say you hooked up the PIA as I described above. In that case, the PIA will be addressed to $FF40 in the CoCo's I/O address space. Port A of the PIA will be located at address $FF40, the control register for port A will be at $FF41, Port B will be at $FF42, and the control register for port B will be at $FF43. Now, let's say you want to program Port A of that PIA so that the first four bits of it will be INPUTS, and the higher order 4 bits of that register will be OUTPUTS. In assembly language, this would accomplish that:

```
INIT  LDA #0
      STA $FF41     set port A
(SFF40) to become a data direction reg
      LDA #%11110000
      STA $FF40     program hi
nibble as outputs, low nibble as inputs
      LDA #%00000100
      STA $FF41     return port to
normal function
```

In BASIC, you could write:

```
10 POKE &HFF41,0
20 POKE &HFF40,&HF0
30 POKE &HFF41,&H04
```

At this point, any effort to READ the low order four bits of $FF40 will reflect the state of signals on the PA0 thru PA3 pins. Anything that you WRITE to the high order four bits of $FF40 will appear as high or low signals on pins PA4 thru PA7.

I've deliberately kept things VERY simple here. I've refrained from talking about any of the other seven bits in the control register, which mediate interrupt handling and signal transition detection functions of the CA1, CA2, CB1, and CB2 pins of the PIA chip. In my simple applications, I don't use any of that. The CoCo's internal PIAs, of course, DO use all of the CA and CB pins, and so setting them up is a bit more complicated.

In the I/O cards that I made for my home use, I was just a bit more elaborate than in the example I described above. I used a single GAL chip to decode addresses (I could have done the same thing with no more than two simple small scale logic chips, but I happened to have GALs and a GAL programmer lying around) and TWO PIA's, which I addressed to $FF70 thru $FF78. By "fully decoding" the ports used for the PIA instead of using the convenient *SCS line of the CoCo, I wound up with an I/O card that, if needed, could be used in parallel with a disk controller card using a Y cable. I've used this I/O card in a number of home projects and applications:

Reading the data out of Hewlett Packard Laserjet FONT cartridges.

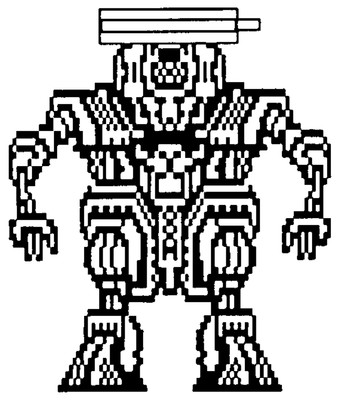
Receiving data from a DTMF decoder chip to monitor what touch tones are pressed by those using a given amateur radio repeater over the course of days or weeks, or decode what numbers were dialed on a telephone that is under surveillance.

Interfacing to a Verification-Protected PAL chip to run it thru all its combinations in order to decipher the logic equations for that chip

The first of these applications required using all 32 lines of my I/O card. I even multiplexed some of those lines, for the HP cartridge has MORE than 32 combined address and data lines (Actually, had I been just a bit more clever, I could have used a counter chip to control the address lines on the HP cartridge, and gotten away with far fewer I/O lines. But having gobs of I/O lines available made my one-of-a-kind project easier to program.)

Some Sneaky Tricks:

If you are using a CoCo for a dedicated application that does NOT require use of the entire keyboard, and needs only 15 general purpose I/O lines, you can use the KEYBOARD PIA chip as your general purpose I/O lines, and just connect into those lines via the keyboard connector on the CoCo. You may have to develop your application using an external I/O card but, when it is up and running, you need only switch addresses to the keyboard PIA and you have a CoCo running as a controller with the 6821's already supplied! You can even use SOME of those keyboard lines to read a small key pad. For example, you'd need only 7 lines to read a "touch tone" style 12 button key pad, and you'll still have 8 lines of general purpose I/O available. When doing this, you need to be a little more careful about how you program the PIA, to make sure not to mess up the CoCo's programing for the CA and CB lines of the CoCo's PIA pins.



CoCoBot?

<268'm>

The Industrial OS-9 User

F.G. Swygert

G-Windows - is it what you need or want?

First of all, I must admit to having no firsthand experience with G-Windows. The following article is based on information obtained from GESPAC, Delmar Co., and Mr. Steve Adams, the program author.

What is G-Windows?

"G-Windows is a powerful windowing and graphical user interface building environment for 680x0 OS-9 and 80x86 OS-9000 systems. Hardware support is provided for G-64, VME, [Euro buss (KiX)] and PC compatible systems [including 680x0 systems with a PC compatible expansion buss (System IV, V)]. The software features extraordinary ease of use, high speed, compact size, and is fully ROMable."

GESPAC Today, Spring 1993

The above lists most of the reasons industrial users have embraced G-Windows. With just a few lines of programming code, a complete graphical interface can be created to run almost any program. This cuts development time drastically, making the end product more cost effective to both the producing company and the end user. The operating system (OS-9) can be completely transparent in embedded systems designed for specialized tasks. This means the end user never has to deal with OS-9's sometimes cryptic command structure, they just point and click the "buttons" on the screen. Since both OS-9 and G-Windows are ROMable, the combination is perfect for embedded applications. The entire package is written in C, so C programmers can most easily integrate their creations with G-Windows. Other programming languages can also be used with a little more work.

What G-Windows Consists Of

There are three major components of G-Windows:

Window File Manager (WMF), is a graphics I/O manager that supplies the basic windowing capabilities of the system. It comes complete with a supporting library of functions that handle all windowing tasks. C programmers can easily use the library to create graphical user interfaces for their programs in a very short time.

G-View is an application utility used to develop program interfaces. The window editor in G-View essentially does most of the work for a programmer. A programmer simply draws what he wants the user interface to look like using a tool box similar to any graphics creating program.

I/O Gadgets (graphical elements, such as "buttons", that are used to control the underlying program) are easily added to the interface with the editor. Several gadgets come with the system. These can be combined to form more complex gadgets.

Gadgets are stored as self-contained, self-maintained program fragments. G-View automatically recognizes and provides full editing support to a gadget as soon as it is loaded into memory, much like any other command or utility. A developer can therefore easily expand the capabilities of a G-Windows based interface by creating entirely new gadgets. These new gadgets are then seamlessly and automatically incorporated into the G-View editor.

Rather than use the drawing primitives in the tool box, bit-mapped images or GIF files can be incorporated into the interface.

A big problem with the powerful OS-9 operating system has always been a lack of user friendliness. Indeed, almost any system with the power of OS-9 has the same problem!

The **G-Desktop** solves this problem by providing a graphical interface to the OS-9 operating system. Most of the file handling and program execution functions of the OS-9 shell are accessed through pull down menus. Storage devices are shown as simple icons. To change directories merely point and click with your mouse! Many of the features are similar to MicroSoft Windows and Apple Macintosh System 7.

There is no limit to the number of windows that can be opened on one screen except overall system memory. This allows a G-Windows user to easily take advantage of the multi-tasking nature of OS-9.

How Much Memory?

An OS-9 system with G-Windows requires a minimum of 1MB to run. Only 457K is used with one text window open. A development system requires a minimum of 2MB. This is about the same requirement for Microsoft Windows. The big memory savings comes when running multiple windows. MS Windows must load a copy of the basic operating system modules into memory for each DOS window opened. G-Windows modules and gadgets are all shareable, so only one copy of each requires memory regardless of the number of programs or windows using that section of code. By referring to the chart below, one can see that G-Windows itself needs a bare minimum of 400K (ROM and RAM combined) to run. The average non-embedded installation uses about 750K of RAM (1.5-2MB recommended).

G-Windows Memory Usage

Module Memory Requirements	
May be in ROM or RAM	
Minimum G-Windows usage -	195K
includes wfm, qfonts, all drivers and descriptors	
G-Desktop -	91.5K
G-View -	300K
Average font -	4.3K
total for all sizes Times, Helvetica, and Courier fonts -	68K
Average Gadget module -	14.5K
smallest gadget module -	5.7K
largest gadget module -	38.6K
total for all G-View gadgets -	218.5K
320x200 Image module -	31.5K
Average G-View window module -	5K
G-View run-time library -	30K
added to G-View application	
Run-Time Memory Requirements	
RAM only	
Base memory, no windows open -	50K
Each window requires -	6.7K
Additional memory for VT-100 -	4K
G-View window -	21K
with 50 "button" gadgets	
Data space for each G-V window -	1.5K
Avg. data space per gadget -	0.4K

< 268'm >

What G-Windows Means To...

The **industrial user** benefits mostly from the ease of developing a good looking, easy to use graphical interface for his applications. The libraries available with G-Windows and G-View also make application programming much simpler and faster- a graphical gauge on the screen (similar to an automotive analog fuel gauge) requires only three lines of C code. It might take over 100 lines to do the same thing without G-Windows and G-View.

The use of G-Windows obviously increases the cost of a system, and it is necessary to run a G-Windows application. Industrial users will save the initial cost in future application development, since development time is greatly reduced. Even if the end user does no programming of his own, money would still be saved in future upgrades and down-time due to minor programming bugs or adjustments- such things can be handled in a shorter amount of time under the G-Windows/G-View environment.

Personal/hobby users benefit mostly from the ease of use the G-Desktop provides for OS-9. It works much like Apple Macintosh System 7, with some MicroSoft Windows features thrown in- the best of both!

Want to see a drive directory? Click on the drive number icon. Need to delete a file? Find it in the directory listing, click and hold the mouse button, and drag it over to the "bit-bucket" icon- it's gone! Want to move a file from a floppy drive to your hard drive? Click and hold the file, drag it over to the hard drive icon, and it's there!

Since the industrial OS-9 user base has started using G-Windows, home users should eventually start seeing applications that were developed for industry use become available at a price the home user can afford. Likewise, industrial users who have already invested in G-Windows systems should be more interested in purchasing software written for the home market that runs on their systems.

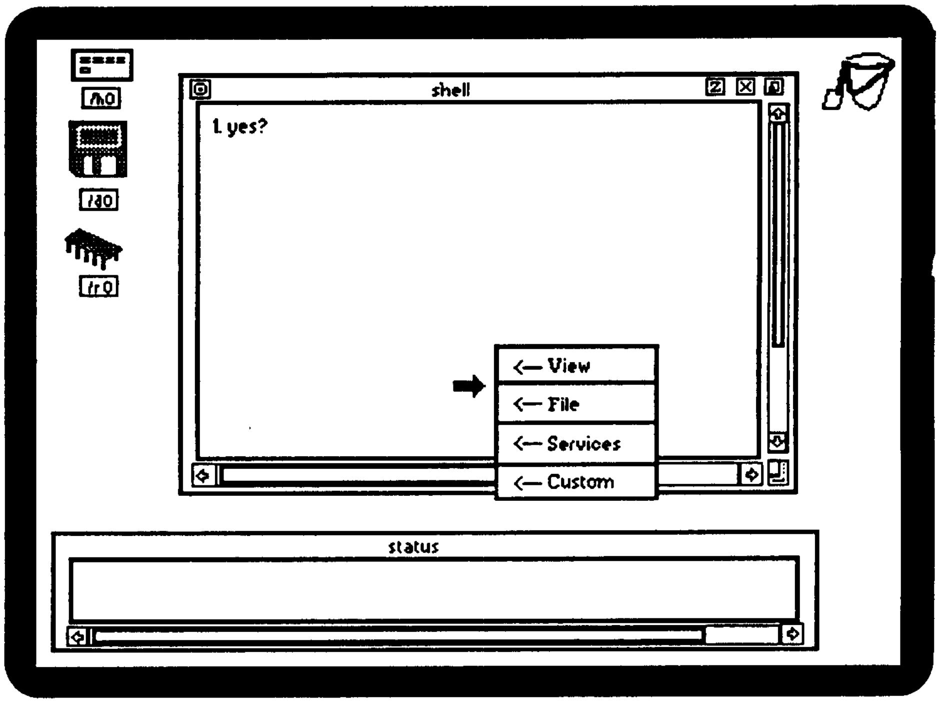
Programmers in either market, especially if they already use C, really get the most benefit from G-Windows. They can now put together professional looking "point and click" programs with much less time and effort involved than required by conventional programming methods. Since development time is a big cost for software, quality software should be more affordable, and better yet, more of it should become available.

Note that other OS-9/68K software will run in a G-Windows window even if it was not written specifically for G-Windows. One can still use terminal based software or even run another GUI through G-Windows.

At What Price...

G-Windows comes with all necessary device drivers and modules, three fonts (Helvetica, Courier, and Times), and the G-Desktop. This package sells from $200-$300, depending on the system it is purchased for. The Program Developer's Pack contains G-View and the C libraries. It usually sells for $300-$400, again, depending on the particular system. The Developer's Pack is only necessary if one intends to write application programs that make direct use of the G-Windows interface. Few users will have a need for the Developers Pack, as BASIC and Assembly Language programs can still be run through a window.

G-Windows is currently available for PT68K based computers (System IV and V) and OS-9000 from Delmar Co. Frank Hogg Labs sells G-Windows for Hazelwood based systems (KiX 20 and 30) and the Interactive Media Systems MM/1. Note that Delmar Co. is also working on a port to the MM/1 at this time.



Graphic representation of a typical G-Windows screen.

82835 8-NOV 04:51 General Information
RE: MM/1 Production (Re: Msg 82822)
From: EDELMAR To: NIMITZ (NR)

> ... But, Gwindows is an INDUSTRIAL SYSTEM. Not intended for personal users.

Wrong! G-WINDOWS was not written with any particular market in mind. It was written to provide a user friendly environment for _any_ user. In this sense, it comparable to Windows, MOTIF, the Mac GUI, etc. It is complete, bug-free, and generally conforms with practices accepted on other platforms. This makes it fairly easy for users new to G-WINDOWS, who are familiar with the other systems, to use it. The acceptance of G-WINDOWS by industry is probably due to its similarity to X-WINDOWS without requiring the resources necessitated by X-WINDOWS. G-WINDOWS is fully supported at 2 levels - first by the Company doing the port and distributing it and second by the author, Steve Adams.

Ed Gresick - DELMAR CO.
Distributor of G-Windows for the System IV & V, PT68K, and OS-9000 *< 268m >*

Beginning With OS-9

NOTE: Though based on CoCo OS-9, this column is good for OS-9/68000 beginners also!

Rick Ulland

Multi-Vue: Yes, it CAN be made to work properly!

Multi-Vue is a bit disappointing on first blush. After struggling though the official install process a few times, one ends up with a painfully slow program that can't find it's own icons, doesn't seem to add very much to the enjoyment of CoCo, causes telecom problems, and doesn't even recognize most of the programs you own anyway. Some folks refer to it as Mutli-Vue! In large part, all of these problems are correctable. Some aren't really problems. But you know how I feel about Tandy manuals!

First things first. MVue (Multi-Vue) will install itself. Sometimes. Doing it manually is no big deal- just replace grfint with windint and add w8-w15 to boot. The rest is just file copying.

As usual, one of the early steps to easier living is having the latest versions. GShell is now up to 1.25, (adds double wide icons on 80 column screens) but 1.24a can also be considered current. As with most system patches, a BBS or PatchOS9 disk will soon have the upgrade in your hand.

If you've been putting off speedup patches like 'Christmas GrfDrv' or even 6309 native mode, by all means reconsider- all GUIs (graphical user interface, or 'gooey') require tons of CPU power. MVue isn't as greedy as most, but CoCo doesn't have any to spare.

The biggest factor is disk speed. MVue wants you to go buy that hard drive- the faster the better. Since it runs exactly as fast as the drive, the 5 times speedup is, shall we say, noticeable. If you have to use floppies, make sure they all step at 6ms, and make the largest drive you have /d0. You aren't supposed to run those 5.25 inch distribution disks anyway. You'll need that big disk to avoid the problem we now address by digression.....

Disk Organization

In the early days of personal computers, disk organization was simple. A program fit on a floppy disk, while it's data either went to that same disk, or a special data disk mounted in another drive. Each program disk had 100% of the code needed to perform it's task, usually as a monolith referred to as a 'program' (Disk Basic assumes this sort of organization).

As PCs got bigger, some of them adopted more advanced operating systems from real computers. These systems require a little more organization than 'run whatever comes off floppy one'.

Enter the tree. Many have described OS-9's hierarchal directory structure at length without really emphasizing one very important fact—we are talking about one tree here.

That's hard to do on small floppies. We do have an advantage- two trees (bushes?) are allowed. OS-9 separates it's system path from the users data path (chx, chd) and allows them to exist on separate devices. Since the chx side can also be loaded into ram, we can double the size of that tree by preloading the box. Remember merged utility files? The personal OS-9 Tandy shipped can be reduced to one 24K file. (dsave and os9gen get pushed off to disk)

Go ahead and load the most used apps from the mother of all start-ups right on the boot disk. Leave about 200K for running things. All this loading does take time- my old floppy boot used about a minute flat. But it frees up /dd for the important stuff. One disk swap and click 'set execute' (to log in the new disk) adds just enough room to run a small OS-9 system the right way.

A MVue dd (default drive) disk is easy to make up: format a new disk, then makdir CMDS, CMDS/ICONS, and SYS. Grab your favorite floppies, and dsave their cmds and sys dirs (chd /d0/cmds;dsave /d0 /d1/cmds ! shell; chd /d0/sys;dsave /d0 /d1/sys ! shell). Just keep piling stuff on. Delete all the stuff you already load at boot from cmds. Now you can organize data disks in any way. Instead of a Dynastar disk, a Supercomm disk, and a Dynacalc disk, you might prefer bowling league, personal letters, and taxman disks.

There may be files that have to go on the data disks as well, for example Dynacalc's .trm file. Sometimes (as in this case) there is a patch to move them to the 'normal' spot in /dd/SYS.... sometimes you have to grin and bear it.

To recap the basic scheme, we have /dd/cmds full of executables, and /dd/sys full of info- any program that needs to know details of the system it's running on can store a file with this info in SYS. All That's missing in this arrangement is a way for the system to know about the program.

AIF Files

Multi-Vue's secret weapon is the AIF file. By specifying window type, size, and color each program gets the correct screen, and the addition of an icon makes these files appear as pretty little clickable things- very nineties. But more importantly, these files assign each application a 3 letter extension. When MVue spots a data file with a known extension, it gets the pretty icon that allows two clicks to open a new window, launch an application, and load the file.

It would be nice (and fast) to have these AIFs and their icons in RAM, but that would be a terrible thing to do to a 256K CoCo. Still, they can't be loaded every time they are needed- not only would floppy systems stop cold, an AIF would have to exist in every directory for every possible application. And copying a file could easily separate it from it's identifying AIF.

The best compromise is used. AIFs are stored on the fly- as you move down through a disk, the number of stored AIF grows. They aren't wiped until a new device is mounted (using drive icons at left).

This can be used to your benefit. For example, now that I find myself writing quite a bit, the old WRITE directory has grown lots of subdirs- CONECT, HUMOR, TW68M, and the like. Some of these have their own subdirs- putting a dynastar aif in each of these would make for about 75 copies. Since I _have_ to move through WRITE to get to any of these sub-sub-sub directories, AIF.doc goes in WRITE.

The possible conflict is obvious. If you have 2 programs that both create xxxxx.cal files they can't both use the same data dir. There is another- can you figure it out?

Clue: 1.(essay) Given the starting point /h0/usr/tcom/dearc, is there any difference between a) clicking on the close dir box that starts the pathlist display 3 times or b) clicking the /h0icon? Explain.

Your job as system administrator is to build and cleverly place these aifs so a user won't be faced with unresolved conflicts. Even if you are the only user. Face it- sometimes it's fun to muck with OS-9, but sometimes there is a job to do. When I'm working on the CoCo, I don't have time to work on the CoCo!

Here are some stupid AIF tricks:

1. Capitalize the AIF in aif. True, either way works, but sort puts capitals first, so your aifs join directories in rising above the 387 ftthhhpppt.dat files to the first screen..

2. Use the implied runb call. Although stock MVue forces you to aif packed basic09 as program=runb with parameters= packedcodename the newer versions allow program=packed codename. This puts the name of the file under the icon, instead of just 'runb'.

3. Here are a couple aif's you might not have thought of:

a. AIF.shl (80x24 text screen with shell)

```
shell
icons/icon.ops
0
7
80
24
1
0
```

b. AIF.b09 (80x24 text screen with Basic09, 32K work space)

```
basic09
#32k
icons/icon.ops
128
7
80
24
1
0
```

If you want to extend the concept further, it's possible to run MultiVue from MultiVue! You may get an error message, but clearing over will show another gshell running. Don't worry if it shows a 'dead gshell' bar- clicking in the correct location will not only pop down the menu, but restore that part of the menu bar to normal. There is a trick to this. The first copy of MVüe has to be already running- else the 'dead gshell' bar is truly dead.

ICONS

MVue is pretty boring without some extra icons. There are two ways to get them- more and more apps are including icons and aifs, plus there are other canned aif/icon sets available from bbs and patch disks. The best solution, though, is to grab one of the public domain icon editors. There are many- it seems every new graphics library comes with an icon editor as demo. There are also plain-jane editors.... any will do fine since making icons is a simple task.

I see a pattern developing here... Operating System Nine is become the first column to feature it's own disk of the month. If you can't easily get these editors and such easily, send 5 bux and I'll fill up a disk with stuff- premade icons, AIF's for common programs, an icon editor, etc.

STARTING UP

The stock issue MultiVue disk takes advantage of OS-9's start up sequence by using the keyname autoex. All they did was copy multistart autoex. If you'd rather it didn't autostart, delete autoex from your boot disk. To save typing, I've renamed multistart MVue, and typing MVue in any shell (even a hotkey shell in gshell) fires up another one.

Even if you NEVER run MVue, you'll want to use control to set up the system. Control sets palette hues, montype, mouse screen res and port, key repeat incept and speed- all with a click. Begin your startup file with control -e. Those clicks become system defaults- even for windows launched in startup. For what MVue brings today, it's worth buying just for control.

OPERATING

The easy part. If set up properly, there isn't a lot to add beyond point and click. There are a few quirks, however.

It's handy to have at least one text shell around. Other than with a shell aif, there's no other way to get a fast text shell from MultiVue. It's also nice to have a different stdfont, so a quick glance reveals window type- some programs you just don't run in a gfx window. Even accidentally.

If you don't have aifs in every directory (likely), catch-14 (thats the chicken/egg thing) will byte. If there was just a xxxx.doc file here I could click it and run Dynastar, which is needed to make xxxx.doc. I s up a shell, and use edit xxxx.doc; space enter; q. Creates a file of one space char with the magic extension. Exit then reenter the dir to get the icon.

To avoid the problem and speed up operation, make creation directories. For instance, when I get a hot new idea, I chd to WRITE. All thats in write are AIFs and the subdirectories where the data is eventually filed, so it comes up fairly quick and I can get to work. At the end of the day the file gets copied to the much more jumbled subdirs.

The exact opposite problem- if a data file ends up with a 'known' extension, many of the file utilities (like print file) won't work since the data is now linked to an app. Usually the easy way out is copy file to root, click drive icon to move to root and flush AIFs.

Running two gshells lets you look at both ends of a disk copy, but don't chd both gshells at once to dirs on the same disk. It will eventually get done... the key word here is eventually. Good stepper motor test, though.

We have reached a watershed. MultiVue marks the end of our blitzkrieg dash through OS-9. Hopefully it now 'begins' pretty much automatically when turned on, and we may concern ourselves with doing useful work. So, wrapping both hands around Frank's throat and jumping about a bit (difficult via email), I calmly convinced him to change the title to Operating system Nine (*editor: In the NEXT issue! I told him AFTER he finished the basics, not NOW! This is what I get for using stubborn egocentric writers... at least he CAN write ...*).

With the title change comes a new skitzoid format. Instead of a topic an issue 'Operating System Nine' will serve up a little bit of everything. Obscure details of an application you don't own make terrible reading so we'll keep it short. At the same time we can fill out our bag of system tricks a trick at a time. I'd like to hear from you. Got a neat system trick? Persistent problem? Write! ***< 268'm >***

Rick Ulland can be reached in care of this magazine, via electronic mail (Delphi: rickuland; Internet: rickuland@delphi.com), or U.S. mail at 449 South 90th, West Allis, WI 53214. Feel free to send questions or comments, but include an SASE if expecting a personal reply. Items sent to the magazine will be considered for publication.

NEW COCO PRODUCTS!

MAX-10
Borders, Boxes, +++

Set of six different style borders and over 40 different boxes, frames, ribbons, scrolls, and lines that can be used with Max-10 to enhance page appearance. Simple to use, with 8 pages of documentation. **$12.00**

Jumpin' Jim's Art Disk

Special set of six flippy disks with two full sides (136 grans) of new, original artwork. Digitized pictures, cartoons, animation, CM3 pix, fonts, borders and letterheads for Max-10, etc. Disks issued one every other month beginning with FEB 94. $30 for all six, $25 with a copy of your "268' m" mailing label. First disk ready *now!*

Jim Bennett
118 Corlies Avenue
Poughkeepsie, NY 12601

Prices include S&H. NY residents add tax.

OS-9/OSK Answers!

Joel Mathew Hegberg

Detect Child Process, OS-9/68K Data Modules, Bells & Whistles for K-Windows

One of the great things about the OS-9 community is the cooperation between various programmers. When one needs a tip or trick, all he needs to do is post a brief message for others to see, and someone quickly answers the question. After college let out for the Christmas season, I had time to work on a few projects and ran into a couple difficulties. I would like to publicly share those problems and the answers my programming colleagues provided, in case anyone else is in a similar situation. I also am including a short program I wrote to play a digital bell-tone under K-Windows, due to popular request. Note that this month's article is exclusively for OS-9/68000 programmers.

I would also like to note that Ted Jaeger DID NOT forget where he was introduced to the concept of mouse and keyboard input. He plainly cited the "Rainbow" article mention in Mr. Gault's letter (Letters, 01 NOV 93) in his original correspondence to me. In the editing process, I neglected to mention that source. My sincere apologies for any inconvenience the omission may have caused!

WANTED: DEAD OR ALIVE

I recently released ModJukeBox for K-Windows, which manages multiple MOD music files and plays them using Russ Magee's 'Strack' program. Russ Magee did an amazing piece of work with Strack! ModJukeBox provides a graphical interface for playing MOD files, and allows the user to choose a sequence of several MOD files to be played automatically.

In order for my ModJukeBox program to know when it was time to move on to the next song, I needed to know when Strack was finished playing the current MOD music file. While I could have simply issued a WAIT(), I wanted ModJukeBox to stay "awake" and responsive to the user while Strack (a forked child process) played the music file. There should be some way to monitor and determine if a child has died yet, right?? (I mean, I saw those Psychic commercials where they depict a mother sensing her child in danger! OS-9 must have something similar, right?!)

Well, Ken Scales came to the rescue! It turns out he had a similar situation while programming KVed, his graphical K-Windows interface for use with Bob van der Poel's VED text editor. He solved the problem by using the _get_process_desc() C function, which provides information about a process (given the process id). When a child process dies, OS-9 holds the information regarding it's death until the parent process picks it up, which prevents another new process from being assigned the process id of the dead child process. So, all that's needed is to check the state of child process occasionally to make sure it's not dead yet. If it is, start up the next music file. Ken provided the following source code to illustrate his example. My thanks to him for allowing me to print this here.

LISTING #1: "chkchild.c"

```
———
#define P_CONDEMNED (0x200)
/* Process state: condemned */
#define P_DEAD         (0x100)
/* Process state: dead */
#include <procid.h>
void main(argc,argv)
int argc;
char*argv[];
{
   int pid;
   procid pbuf;
   unsigned short childstate;
childstate=!(P_DEAD|P_CONDEM-
NED);
/* Intitial assumption is not dead or
condemned */
   ... program stuff ...
 if (_get_process_desc(pid,sizeof
(pbuf),& pbuf)!=-1)
       childstate=pbuf._state;
    else
       childstate=P_DEAD;
       if (childstate & (P_DEAD
|P_CON-DEMNED))
    {
  /* child process has died... */
  /* Do your processing here, includ-
ing issuing a WAIT() call to release the
dead child's process id. */
       }
    }
```

Ken also notes that there is a bit of information on this topic on pages 362 and 370 of "The OS-9 Guru," which is a fantastic book for any serious OS-9 programmer.

OS-9/68000 DATA MODULES

In another program I'm currently working on, I would like to make use of one of OS-9/68000's most useful features... a data module. Data modules allow several programs to share a common dataspace, so more than one program can utilize the information stored within the data module. In my case, I have about 300k of data that subsequent programs I write may wish to use as well. Without the use of data modules, each program would have to load in the 300k of data separately, eating up lots of memory! This way, only one 300k hunk of memory is eaten up by the data module and shared by multiple programs. When the data module is no longer needed (i.e. all the programs using it quit running), it is removed from memory.

"Great!," I thought... "What could be simpler?" So, I threw open my Microware C manual and looked up "_mkdata_module()." It mentioned the header file "module.h" needed to be included (ok), the parameters for the call (ok), and that the function returned a pointer to a "mod_exec" structure (huh??). I took a look at the header file "module.h" and found the mod_exec structure. I was looking for a pointer to the place where I'm allowed to store my data, but found several defined pointers, including _mexec (offset to execution entry point), _mexcpt (offset to exception entry point), _midata (offset to initialized data), and _midref (offset to data reference lists). Once again, "huh??"

My best guess was to try using the _midata pointer, since I was dealing

KiX/20...

32 bits of affordable excellence

32 bit Performance at 16 bit cost!

NEW LOW COST 32 BIT COMPUTER FOR OSK!

The KiX\20 has the same 32 bit expansion bus as the KiX\30 and uses the same expansion boards. This includes the Ultra fast Multi Graphics Adaptor video board. The 32 bit MGA video board adds high end graphics to the low cost KiX\20. This combination provides the end user with high performance workstation capabilities at home computer prices. *If you are looking for an OSK machine that provides super fast video and low cost, then this is it.*

The KiX\20 specs:

►25 Mhz 68020 CPU
►1, 2, 4, 8 or 16 Meg of RAM on the motherboard using SIMM memory.
►On board SCSI controller (same as KiX\30)
►Dual density floppy controller. Supports 360K to 1.4 Meg floppies.
►8K Battery Backed Static RAM.
►4 on board full serial ports (same as KiX\30)
►Battery backed Real Time Clock
►Sound port
►1 Parallel printer port
►Full user manual with schematics
►Autoboot from floppy or hard disk included.
►4 Built in I/O expansion bus.
►1 KiX\30 32 bit Expansion bus connector
►4 layer PCB
►Small 8.7 in by 8.7 in. size drops in any PC case

includes...

►PROFESSIONAL OS9/68000
►MICROWARE K&R C COMPILER
►MICROWARE BASIC!
►MICROWARE MANUALS
►Run as a terminal system now, then add a video board later.

The bus runs at full CPU speed. The bus is not a limiting factor like the slow AT/ISA bus used on most PCs available today. (The AT/ISA bus is limited to 8Mhz and is only 16 bits.)

The performance gain of this system over current systems is dramatic.

In today's economy, cost is a major factor. This system is designed to start at only 699.95. That includes Professional OS9/68000, Microware K&R C compiler, Microware BASIC, deguggers and tons of utilities.

You can start out with a terminal based system and add the MGA video board later.

The introductory price for the KiX\20 is:

Only $699.95!

And that includes...
OS9/68000, C and BASIC!
Order today!

FRANK HOGG LABORATORY, INC.
204 Windmere Road Syracuse, NY 13205 *Fax:* 315/469-8537 *Telephone* 315/469-7364

KiX Systems

KiX\20 SALE

Terminal Systems

KiX\20 MB w/OSK	699.95
Case/PS/Floppy/Cables	249.95
Price seperately	949.90
Save	-50.00
ONLY	**899.95**

Video Systems

KiX\20 MB w/OSK	699.95
MGA Fast 32 Bit Video	450.00
G-Windows GUI	275.00
Case/PS/Floppy/Cables	249.95
Price seperately	1674.90
Save	-274.95
ONLY	**1399.95**

KiX\30 SALE

KiX\30 MB w/OSK	*1599.95
Case/PS/Floppy/Cables	249.95
Price seperately	1849.90
Save	-50.00
ONLY	**1799.95**

* 16 MHz version
Add $100/200 for 25 MHz /33 MHz

KiX\30 MB w/OSK	*1599.95
MGA Fast 32 Bit Video	450.00
G-Windows GUI	275.00
Case/PS/Floppy/Cables	249.95
Price seperately	2574.90
Save	-274.95
ONLY	**2299.95**

* 16 MHz version
Add $100/200 for 25 MHz /33 MHz

Hard Drives

Add a Hard Drive to any of the above systems. All hard drives come pre loaded with the latest software ready to use.

Quantum 52 Meg	149.95
Quantum 170 Meg	299.95
Maxtor 245 Meg	375.00
Maxtor 350 Meg	525.00
Quantum 500-+ Meg	Call

Note: All systems require some assembly. No soldering. Takes about 1 hour. All components are pre tested and guaranteed for 1 year. All are shipped with Zero K RAM. Call for latest prices on SIMM DRAM. Uses standard Mac (8 bit) or PC (9 Bit) SIMMs.

Software

35% OFF W/SYSTEM

DynaStar 140.00
Word Processor with the easy to use and familiar user interface. The standby used by many OS9 users.

VED Word processor 60.00

Fast BackUp 50.00
Backups by file list using compression.

Sculptor v1.14 200.00
Older version at greatly reduced cost.

KEEPER Business Software 299.95

G-Windows

MM/1 VERSION NOW SHIPPING!

Designed Specifically for OS-9 Users

The *G-Windows* windowing software goes far beyond anything ever offered for the OS-9 operating system. Its modular structure, multi-tasking 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.

G-Windows pricing

G-Windows for KiX	275.00*
G-Windows for MM/1	200.00
G-Windows for OS9000	275.00

Includes: Window manager and required modules. Desktop manager and related files. Fonts and image modules. Basic set of gadgets used by many utilities. GIF viewer. Manual: "Using G-WINDOWS/Desktop Manager" Easy setup information.

*KiX versions are packed with the MGA video board at reduced cost.

G-Windows Development

Develop software for G-Windows.
OS9/68000 Dev System 300.00

FRANK HOGG LABORATORY, INC.
204 Windmere Road Syracuse, NY 13205 *Fax:* 315/469-8537 *Telephone* 315/469-7364

with data, right? Before trying it, I thought I'd post a message in Delphi forum... after all, I really didn't have a clue! Stephen Carville solved the problem for me, and I was glad I asked. It turns out the _mexec (execution offset) is where you want to store data, NOT the _midata (data offset)! Here's Stephen's sample source code (my thanks to him for permission to print it here):

Listing #2: "datamod.c"

```
#include <module.h>
main()
{
  char *dataptr;  /* Pointer to whatever data-type to be stored */
    mod_exec mp;

    mp=_mkdata_module(modname, datasize,attrevs,perms);
        dataptr=(char *)mp+mp->_mexec;  /* data pointer */
}
```

Notice that what Stephen is doing is taking mp (which is a pointer to the module header) and adding to it the execution offset, since an offset is not an absolute memory address, but rather is a relative pointer. Stephen also mentions to make sure you don't mess with the value returned by _mkdata_module() ('mp' in the source code above), as it will be needed to unlink the module when the program is finished using it. OS-9 will not automatically deallocate a data module when the creating process terminates.

K-WINDOWS STEREO BELL

A few programmers have mentioned how much they like the stereo "bell" sound I use in Write-Right! (my commercial word processor for K-Windows, available through Sub-Etha Software). It's actually a very simple routine to set-up. Since it's such a simple sound, the frequency for playback doesn't need to be very high, so the memory the sample takes up can be very low (specifically, 1k)!

The idea behind the sample is I want a tone that evenly decreases in volume. The tone's frequency is not of concern, since it can be varied by the K-Window s _ss_play() setstat call. The volume of a tone is determined by the value difference from one byte to the next being sent to the computer's speaker. So, if I first send the byte value $80 to the speaker, and then send a byte value of $00 to the speaker, there is a large difference and the sound-wave has greater amplitude (i.e. volume). If I slowly make the difference between subsequent bytes shrink, the tone will taper off and sound like a chime or bell. The data bytes could look like this:

```
$00 $80 $00 $7f $00 $7e $00 $7d ... $00 $01 $00 $00
 LOUD                     QUIET
```

Because the K-Windows _ss_play() function is a stereo call, in actuality each byte needs to be duplicated, since the first byte goes to one speaker, the second byte to the other speaker. So, the data would actually be something like:

$00 $00 $80 $80 $00 $00 $7f $7f ... etc.

Ok, enough theory... here's the source code: (Please note that the source requires the 'cgfx.l' library for the _osk() function call.)

Listing #3: "playbell.c"

```
/* Playbell routine by Joel Mathew Hegberg */
#include <Stdio.h>
#include <MACHINE/regs.h>

#define PATH 1
#define SNDMEM 1024

extern char *malloc();

REGISTERS reg;

main(argc,argv)
int argc;
char*argv[];
{
  int length,t;
  char *sound_data,*tp,e;

  sound_data=malloc(SNDMEM);
  if (sound_data==NULL)
  {
    fprintf(stderr,"\nPlayBell: Cannot allocate %d bytes of memory. \n\n",SND
    MEM);
       exit(0);
    }
    e=0x80;
  for(t=0,tp=sound_data;t<SNDMEM; t+=8,e--)
    {
       *(tp++)=0;
       *(tp++)=0;
       *(tp++)=e;
       *(tp++)=e;
       *(tp++)=0;
       *(tp++)=0;
       *(tp++)=e;
       *(tp++)=e;
    }
    reg.d[0]=PATH;
      reg.d[1]=0x0097; /* $97 is K-Windows _ss_play setstat code */
    reg.d[2]=SNDMEM; /* Number of bytes of sound data to play */
    reg.d[3]=1000; /* Playback at 1000 Hz */
     reg.d[4]=0; /* Must be zero */
      reg.d[5]=0; /* No signal to send when done */
     reg.d[6]=0; /* Must be zero */
      reg.a[0]=sound_data; /* Address of sound data */
      _osk(0x8e,&reg);
      free(sound_data); /* Release the memory back to the system */
  }
```

Of course, if this is used in an application program, such as Write-Right!, you don't want to be reprogramming the sound data every time you want to ring the bell. Rather, you allocate sound data memory, initialize the sound data, and then whenever you need to ring the bell, simply call the _ss_play() K-Windows function. Before exiting the program, you should release the memory back to the system, using the free() function.

See you next issue!

Joel Mathew Hegberg
932 N. 12th Street
Dekalb, IL 60115
JOELHEGBERG on Delphi
joelhegberg@delphi.com on Internet
J.HEGBERG on GEnie

Programming in "C"

P.J. Ponzo

FOR WHILE and other good stuff.

```
main() {
int i
i=1;
while (i<11); {
printf("\n The square of %d is
%d",i,i*i);
i=i+1;
}
}
```

This is the opening for main().

main() {

This is the closing for main().

}

```
int i
i=1;
```

Here we declare i to be an integer variable, and define it to be (initially) the integer 1. Find the error here!

int i should be **int i;**

WE FORGOT THE SEMI-COLON!

Now that we're debugging our program, let's change these lines so that the declaration and the initialization of i are together:

```
main() {
int i=1;
while (i<11); {
printf("\n The square of %d is
%d",i,i*i);
i=i+1;
}
}
```

Here's something new...it says to execute certain statements again and again as long as i is less than 11 (or while i<11). Execute what statements?

```
while (i<11); {
}
```

All the stuff between these curly brackets! ...and this stuff says to printf:

The square of   is

value of i   value of i*i (square of i)

goes in here.   goes in here.

THIS PROGRAM IS HARD TO READ! Change it!

```
main()              /* sexy program */
{                   /* start main() */
   int i=1;         /* declare i=1  */
   while (i<11); {  /* while i<11   */
      printf("\n The square of %d
is %d",i,i*i);      /* print i, i*i */
      i=i+1;        /* increment i  */
   }                /* end of while */
}                   /* end of main  */
```

Here's the same program again... but nicer to read! Anything between /* and */ is a comment and is ignored by the C-compiler (it's for human consumption only), so we've added a comment to every line. NOW we can see what the program does by reading only the comments! Indenting the various parts makes for easier reading (again for human consumption ...the compiler doesn't care).

The start and end of main() are easy to spot. Although different programmers use different formats, WE will always start and end main() with { and } in the first column. Well ...sometimes we will start with: **main() {**

```
{
}
   while (i<11); { /* while i<11   */
   }               /* end of while */
```

...and we start a while loop with **while (....) {** and end it with } placed directly below the w in while. And we will always (?) indent (by 4 spaces) these inside loops.

```
main()              /* sexy program */
{                   /* start main() */
   int i=1;         /* declare i=1  */
   while (i<11); {  /* while i<11   */
      printf("\n The square of %d
is %d",i,i*i);      /* print i, i*i */
      i=i+1;        /* increment i  */
   }                /* end of while */
}                   /* end of main  */
```

Alas, this program won't even compile! Whereas most C statements end in a SEMI-COLON, the **while (...)** does NOT. We must delete the ; after a while.

```
while (i<11);~W { /* while i<11   */
   while (i<11) { /* while i<11   */
main()              /* sexy program */
{                   /* start main() */
   int i=1;         /* declare i=1  */
   while (i<11) {   /* while i<11   */
      printf("\n The square of %d
is %d",i,i*i);      /* print i, i*i */
      i=i+1;        /* increment i  */
   }                /* end of while */
}                   /* end of main  */
```

The construction:

```
i=1;
while (i<11) {
    some statements;
    i=i+1;
}
```

occurs so often (in any language) that a slick mechanism exists for handling this loop:

```
for (i=1; i<11;
i=i+1) {
    some statements;
}
```

```
main()              /* sexy program */
{                   /* start main() */
   int i;           /* declare i    */
   for (i=1; i<11; i=i+1) {
                    /* the for loop */
      printf("\n The square of %d
is %d",i,i*i);      /* print i, i*i */
   }                /* end of for   */
}                   /* end of main  */
```

Note that the for loop automatically initializes i to 1, then does the printf() again and again, each time incrementing i, until i has the value 11 (..then the program exits from this loop after printf-ing for the last time with i=10). The value of i, after the exit from the loop, is 11.

Just to check it all out, we leave our word processor after saving this source code under the name program2.c, then type:

```
cc program2
```

Then, assuming it compiles without errors, we finish with:

```
link program2
```

Then, since this compile/link procedure will generate an executable file called program2.exe, we type:

program2

Now the executable program will load from disk, then execute, to give:

```
The square of 1 is 1
The square of 2 is 4
The square of 3 is 9
The square of 4 is 16
The square of 5 is 25
The square of 6 is 36
The square of 7 is 49
The square of 8 is 64
The square of 9 is 81
The square of 10 is 100
```

Wonderful!

```
1   i=5;
2   while (i<5) {
3      some statements;
4   }
```

In this piece of code, the while loop will be executed only as long as i<5. Since we set i=5 in Line 1, the loop would be bypassed.

The condition, in a while loop, is checked at the beginning of the loop! Usually this is what we want .... but, sometimes it is NOT:

```
while (sam>100)   {
```

---

some statements which calculate some numbers and use these to compute the value of sam.

```
}
```

In this piece of code the value of sam is not even known until we go through the while loop... so we want to check the while-condition at the END of the loop !!! Now DO this for a WHILE

```
while (sam>100)   {
```

---

some statements which calculate some numbers and use these to compute the value of sam.

```
}
```

We replace the above construction by a DO-WHILE:

```
do {
```

---

some statements which calculate some numbers and use these to compute the value of sam.

```
} while (sam>100);
```

...and (magic) the while-condition is checked at the end of the loop!

```
1   double x=1.0, y, e;
                /* double precision ! */
2   do {      /* start of the do-loop*/
3     y=2.0*sin(x); /* calculate y */
4      e=fabs(y-x);
                /* calculate error  */
5      x=y;        /* change x to y */
6    }  while (e>.0000005);
                /* end condition   */
7      printf("x-2sin(x)=%f when x=%f",e,x);
```

This program calculates the root of the equation (x-2*sin(x)=0) by starting with x=1.0 (line 1), then repeatedly replacing x by y in line 5 (where y is calculated as 2.0*sin(x) in line 3).

While the error (the floating point absolute value of y-x, calculated in line 4) exceeds .0000005, we repeat the loop.

Finally, when the while-condition (in line 6) is false (i.e. when e is LESS THAN OR EQUAL to .0000005), we print: x-2sin(x)=0.000000 even when x=1.895494 correct to 6 decimal places! It's nice to check the error e after we go thru' the loop!

A REVIEW

```
while (something is true ) {
   do these statements;
}

for (initialize variables;repeat,if this is true;do this at end of loop) {
   do these statements;
}
```

NOTE: If there is only one statement to perform, in either a while or a for loop, then we don't need the { and }:

```
for (i=0; i<11; i=i+1)
  NO OPENING {
   printf("\n The square of %d is %d",i,i*i);
  or CLOSING }
do {
   do these statements;
} while (something is true);
```

NOTE: The while which occurs at the end of a DO loop needs a semi-colon!

We can also invoke a function (like getchar()) while inside....

```
char key;
while ( (key = getchar()) != 'e' )
     printf(" You pressed %c \n",key);
printf("\n  THAT'S THE eND");
```

...where we wait for a single keypress (that's what getchar() does!), and assign the key to the char variable key via key=getchar(), and so long as key is not equal to the letter 'e', we printf() the key (as a %character) and then a \newline .

NOTE: scanf("%c",&key) would require your pressing the enter key after each of the letters a, b, etc. ... so we used getchar()!

This program would give (if you pressed a then b then c etc.):

```
a You pressed a
b You pressed b
c You pressed c
d You pressed d
e  etc., etc. ...
```

P.J.Ponzo
Dept. of Applied Math
Univ. of Waterloo
Ontario N2L 3G1

<268'm>

CoCo-C

A complete Disk BASIC based C development package for the Color Computer NOT requiring the OS-9 Operating System. Contains:

Text Editor -
C Compiler
Assembler
Library/Linker -
Command Coordinator
CoCo 1,2, & 3 versions included.

$59.95

+ $4 S&H (check, money order, or COD)

Infinitum Technology
P.O. Box 356
Saddle River, NJ 07458
Phone 914-356-7688

Basic09 In Easy Steps

Chris Dekker

Differences between Disk Extended Color Basic, IBM Basic, and Basic09

Since It seems to be very hard for today's editors to get good Basic09 articles for their magazines, I have been asked to fill (some of) the gap. Among the issues that popped up were conversion of DECB and MS-DOS basic programs into Basic09. This is a wide field that could cover a number of articles by itself. However, I didn't think it is wise to jump into the subject without providing at least some background information on the differences between the languages. Starting up Basic09 from scratch will also be discussed.

I will start with DECB as most readers here are (at least somewhat) familiar with it. If you want to do a program conversion you have to understand that DECB is not just a Basic language. The code also functions as editor, operating system and graphics interface.

Basic09, on the other hand, is just a programming language. Commands that are part of the operating system's functions are dealt with by OS-9. Since it also had to be portable to computers with different graphics capabilities, the graphics interface is contained in separate modules. On the CoCo these modules are called gfx (low/medium resolution screens) and gfx2 (high resolution screens).

If you run into a function that you can not perform under Basic09 because it is part of OS-9's territory, you can issue a request to OS-9 using a utility called syscall. This utility, along with the gfx modules, can be accessed from Basic09 by simple RUN statements.

If you think there is more to it than that, you are right. You will have to contend with parameter passing, program optimization, etc. We will get to those things later on. For now all you have to keep in mind is that this is the basic triangle from which you build your Basic09 programs.

As for the structure of MSDOS basic: it, looks a lot like DECB. Presumably this is because both were developed by Microsoft. The reason they got away with this is that at a certain level the hardware of all PC compatibles looks very much alike. This allows for across the board use of the same graphics modes etc. Unfortunately for PC users this backward (in more ways than one) compatibility comes at a price: the computer's performance.

I have a PC compatible with a 25 Mhz 486SX processor in it. By all accounts this machine should run circles around the CoCo. In fact it will do so but not under the Quick Basic included in my version of MSDOS. Programs that are mostly calculating jobs run about 10x as fast as in Basic09, for graphics programs this slows down to a factor 5.

For all practical purposes, if you convert a MS-DOS basic program to Basic09 you can expect it to run about as fast as it would on an AT-class machine 80286/12MHz). There are a number of reasons for that most of which I won't go into here: they deal with the specifics of the microprocessors themselves, the way the machines address their memory, etc.

There is one important software reason, however, that I will go into. MS-DOS basic, like DECB, is an interpreter type language while Basic09 is a compiler type language. What it boils down to is that interpreters translate a given program line by line. They translate the first line execute it, translate the next line, execute it, etc.

A compiler like Basic09, on the other hand, first compiles the entire program into what is called intermediate code (I-code). Once it is done with that it can start executing the program with much less overhead so the program executes faster. Another advantage of I-code is that it uses relatively little space in memory which means you can get a lot done even with limited memory available.

One disadvantage (if you want to call it that way) is that compiled programs can not be altered or alter themselves. If you want to make changes you will have to do that in the source code and recompile that.

Fortunately, Basic09 implements the various functions of a compiler in a user friendly way. All the user really has to do is remember the difference between SAVE and PACK.

For the more curious among us, here is how Basic09 (roughly) works. If you decide to enter some (source) code into Basic09 you must do so through it's editor (the one part that could be more user friendly). You can enter the editor by typing e or edit plus a program name at the B: prompt. When you type in a line of code Basic09 checks it for typing errors, misplaced operators, etc. and translates it into I-code.

When you are done entering program code you type q<enter> at the editor's E: prompt. At this point Basic09 starts checking various aspects of the program. If there are problems with loop structures, complex datastructures or variable typing, Basic09 will report those problems using a code number telling you what problem to look for as well as the address of the offending code. Note that these addresses are the numbers shown in the leftmost column of Basic09 program listings. An explanation of the code numbers (what they mean) can be found in the OS-9 manual (Basic09 section: appendix A).

Assuming you have resolved all the problems in your program, you can now run your program. You do this by typing RUN plus the name of the program. If you didn't get all errors out Basic09 will display an error 51 message. This is labeled as a compiler line error and means there are still one or more lines in your program that have not passed the preliminary tests.

Of course the fact that your program runs doesn't mean that the code is bug free. The computer for one is happy if it can just run the code and couldn't care less whether it's results make sense or not. It is also possible that there are array pointers that end up pointing outside the intended array (reported as error 55), etc. Unfortunately Basic09 can not check for these things in advance so you will have to test your program for such errors yourself.

Since Basic09 is what we call a multi-pass compiler it is not quite finished with your program yet. At this point in the development stage Basic09 holds all line numbers and comment lines in memory as part of the program code. To save this code to disk you must use the SAVE command. If you need the code to make changes to it later on you can load it back into Basic09 using the load command and edit it with the edit command.

However, if you are convinced your program is ready to run on itself all the comments and line numbers in it become obsolete. You can purge them using the PACK command. PACK saves the program to disk in a form that can not be disassembled (at least not by Basic09) and consequently you can no longer change the code. On the upside: your program now runs faster and takes up less space.

Just keep in mind that you must ALWAYS keep a SAVEd version of your program around in case you want to make changes to the code. You don't have to worry that PACK will overwrite the source code file: Basic09 uses the file attributes to prevent that. The other thing I want to mention here is that Basic09 looks for source code files in the data directory (CHD) and places PACKed files in the execution (CHX) directory. Next time I will show you how to get Basic09 to run.

<268'm>

OS9> B

Support for OS-9 Level II: Shell+

Robert Gault

GetNW - Using Shell Variables and Subshell Modules

Two of the more intriguing aspects of the latest OS-9 Shell replacement, Shell+ 2.1, are shell variables and shellsub modules. I intend to present some examples of how these features and a few others can be used.

When I run OS-9 on my CoCo 3 system, I generally have three 80 column text screens and an occasional graphics screen active. I require a fast foolproof method of distinguishing one screen from another without being able to see the Shell+ prompt. My solution was to assign a different border color to each new screen. Further I wanted a short program that could automatically create the desired windows so that I could start new screens with unique border colors at will.

If you give this some thought, you will see the problem is not trivial. The following parameters must be kept track of:

1) Number of active screens.
2) Names of active screens; i.e.. /Wn
3) Border colors used.
4) Name of next available window.
5) Make the Display command accept variable data.

In this and the next issue, I will present several programs and shellscripts that achieve the goal presented above. I'll start with a method for finding the name of the next available window.

OS-9 Level II incorporates the generic window name /W which is supposed to access the next available window. Most of the time this works but there is no mechanism for the user to retrieve the information. Worse still, each time /W is used, a new window is accessed. What is needed is a method for trapping the name a single pass with /W yields and saving it for future use.

There is a program in the public domain, getNW, which addresses this issue but it reports the full name /Wn. This is not suitable for my purposes so my versions, which report only the number, are given below in the listing section.

Note that my versions of getNW send output to the standard OS-9 output path so that redirection is possible. Version 1 uses the I$GetStt function SS.DevNm, is very short, and seems bullet proof. It is not!

Those familiar with OS-9 know that Level II windows can be converted into VDG type windows using Tmode or Xmode. Bit 7 of the type parameter is set 0 for VDG and 1 for Window screens. However, version #1 of getNW can not recognize a VDG type screen. There is some sense to this in that it prevents sending windowing functions to a VDG screen, but there are occasions when other types of messages need to be sent. The main problem is, that on conversion back to a true Window by setting type=$80, /W and therefore getNW #1 still does not recognize the altered window!

I have found that the reason for this is that neither I$Close nor F$Exit issue an implied F$Unlink when closing paths to devices. Once a window is modified to VDG, /W will only find that window on two conditions: 1) reconversion to true Window, 2) reduction of link count to 0. However, unless you specifically use F$Unlink or Unlink, the link count will never be zero. There is a definite OS-9 bug here that needs to be addressed but that's another story.

GetNW #2 is a different approach which is not affected by the VDG problem. The technique is to read the system active device table for windows in use. Surprisingly this version, even though considerably longer, is noticeably faster in operation.

Use the version of your choice by entering getnw. The response will be the number of the "next" available window. Next time we'll see what can be done with getnw.

Listing 1

```
* Get next available window name via a getstat
* call and send ans to standard output.
* R.Gault Sept. 1993
* Reports ONLY the number of the next
* window and assumes that all window names
* are of the form w#

      nam  getNW
      ttl  Program Module
      ifp1
      use  /dd/defs/os9defs
      endc
TyLg   set  Prgrm+Objct
AtRv   set  ReEnt+Rev
Rev    set  2
     mod  Eom,Name,TyLg,AtRv,Start,Size
NAME   equ  *
      FCS  /getNW/
      FCB  1

cr     equ  $0d

* data starts here
wname  rmb  32
stack  rmb  $100
Size   equ .

gwin     fcc "/w"
      fcb cr
Start  lda #1 read  open path to /W; link count
+1
      leax gwin,pcr
      OS9 I$Open
      bcs error
      ldb #SS.DevNm  link counts to /W & /
W# +1 on get device name
      leax wname,u
      OS9 I$GetStt
      bcs error
      pshs x
      OS9 I$Close   close path
* The next few lines are here because I feel that
* when paths to OS-9 devices are closed, the
* link counts should decrease. That they won't
* on there own I feel is an OS-9 bug.
      clra        set any module type
      OS9 F$UnLoad   decrease link count;
regX already points to W#
      leax gwin+1,pcr  point to W; it needs a -
2 link change
      OS9 F$UnLoad
      leax gwin+1,pcr  repeat
      OS9 F$UnLoad
      ldx ,s       recover pointer to W# name
nmloop  lda ,x+      find last byte of name
      bpl nmloop
      anda #$7f    remove FCS format
      sta -1,x     and tell name about it
      lda #cr      add a CR for a "clean" report
      sta ,x
      puls x       recover pointer to W# name
      leax 1,x     skip past W; assumes all
windows have W# format
      lda #1       standard output
      ldy #10      just generous
      OS9 I$WritLn  write the window name
through the CR
      bcs error
quit    clrb        enter here; show no errors
error   OS9 F$Exit   enter here; keep error
indicated by regB

      EMOD
EOM    EQU  *
```

Listing 2

```
* Get next available window name using
* device table. Not sensitive to VDG/non-
* VDG type windows
* R. Gault Sept. 1993
      nam  getNW
      ttl  Program Module
      ifp1
      use  /dd/defs/os9defs
      endc
TyLg   set  Prgrm+Objct
AtRv   set  ReEnt+Rev
Rev    set  $01
     mod  Eom,Name,TyLg,AtRv,Start,Size
```

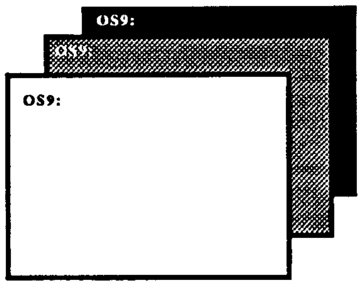
```
* data starts here
dpadr   rmb  2    pointer to direct page
datptr  rmb  2    pointer to dat image
wlnptr rmb 2  working pointer into line buffer
linptr  rmb  2    pointer to start of line buffer
iotptr  rmb  2    pointer to i/o device table
counter rmb  1
actwnd  rmb  1    total active windows
number rmb 1  used to convert ascii # to value
linbuf  rmb 31  storage for window numbers
in use
datimg  rmb  64   copy of DAT image
iotbuf rmb $100 primary use; i/o device table
mainbuf rmb 512  buffer for system process
descriptor; will have DAT image
stack   rmb  $100
SIZE    equ  .
NAME     equ  *
     FCS  /getNW/
     FCB  $02
Start  stu  dpadr     save direct page pointer
     leax linbuf,u   point to line buffer
     stx  linptr
     stx  wlnptr     save current item pointer
     leax  datimg,u
    stx  datptr   save pointer to what will be
DAT image
       clr   actwnd      clear active window
counter
     lda   #$01      process id#: 1=system
   leax mainbuf,u pointer to 512 byte buffer
area
     os9  F$GPrDsc   get process descriptor
      bcs   error
     ldb   #64       size of DAT image
      leax  P$DATImg,X     point to DAT
image in descriptor
     ldy   datptr     point to copy space
cloop   lda   ,x+      copy DAT contents; free
buffer
     sta   ,y+
     decb
     bne   cloop
main   leau iotbuf,u  point to buffer; i/o table
     stu   iotptr     save pointer
     ldd   datptr    get pointer to DAT image
     ldx   #D.DevTbl  request bytes starting
at device table pointer
     ldy   #$0002     copy 2 bytes
     os9   F$CpyMem   get pointer to device
table
     blo   error
     ldx   ,u        read address of device table
    ldy  #$100    bytes to copy; room for 28
entries
     ldd   datptr    point to DAT image
     os9   F$CpyMem    copy the i/o table
     blo   error
   ldb  #28     initialize counter; max entries
we left room for
     stb   counter    store value in counter
     ldu   dpadr      point to DP
     ldx   iotptr     initialize regX
```

```
mainlp  bsr  readdsc    get device descriptor
if present and
* obtain desired info from descriptor ie. name
     dec   counter     update counter
     beq   main2
      ldx   iotptr     update table pointer by
DEVSIZ; entry size
     leax  DEVSIZ,X
     stx   iotptr     save corrected pointer
     bra   mainlp
main2   lbsr  sort
     lbsr  sndans     send answer to standard
output
finish  clrb
error   os9   F$Exit
* get and read device descriptor; print required
data
readdsc ldx  V$DESC,X  get pointer to device
descriptor module
     bne   getinf     if there is one then ....
nxtdsc  rts
getinf  pshs  u
    leau mainbuf,u point to device descriptor
buffer
     ldd   datptr     point to DAT image
     ldy   #200      bytes to copy; just in case
     os9   F$CpyMem  get device descriptor
     puls  u
     lblo  error
    leax mainbuf,u  point to device descriptor
     ldd   M$Name,x    get offset to Device
NAME
     leax  d,x        point to device name
     lda   ,x+        get first letter
     anda  #^$20      make upper case
        cmpa  #'W          is this a window
descriptor?
     bne   nxtdsc     no?; then try next device
     clr   number
gloop   lda   ,x+        get window number
     bmi   lsn
     bsr   cnvnum     convert number byte
     bra   gloop
lsn     anda  #$7f    compensate for fcs format
     bsr   cnvnum     convert number byte
     lda   number
    bra   storen     store in line buffer; return
from sub.
* convert ascii # to binary value: read left to
right; each time
* ans = ans*10+digit
cnvnum  suba  #'0       convert to binary
     pshs  a
     lda   number     get total
     ldb   #10
     mul            *10
     addb  ,s+       add in last digit
     stb   number     save new total
     rts
* send window number to standard path #1;
first convert to ASCII
sndans  ldx   linptr    point to buffer
     ldy   #2       minimum value of bytes to
```

```
send to output
     cmpa  #9
     bls   onedg
     leay  1,y       increase character count
#>=10
     ldb   #-1
asci    incb
     suba  #10      divide by 10
     bhs   asci
    adda  #10     overshot so correct reg.A
     addb  #'0
     stb   ,x+
onedg   adda  #'0
     sta   ,x+       store the first available
window #
     lda   #$0d      CR
     sta   ,x
     lda   #1       standard path
     ldx   linptr    point to buffer
     os9   I$WritLn
     lblo  error
     rts
* update line buffer with new window number
storen   pshs  x
     ldx   wlnptr    get pointer
     sta   ,x+     put result in memory
     inc   actwnd    count number of active
windows
     stx   wlnptr    save pointer
     puls  x,pc

* find highest # window in table; table not
sequenced
sort    lda   #1      first possible window is 1
     tst   actwnd
   beq  nowind   trap unique case; no active
windows
sortlp  ldb   actwnd
    ldx  linptr  point to list of active windows
sloop2  cmpa  ,x+     is this window active?
     beq   badwnd   yes?, then go
     decb        decrease counter
     bne   sloop2   try next window
nowind  rts     exit with reg.A=first available
window #
badwnd  inca       update window value
   cmpa  #30    trap upper limit; you never
know!
    lbls  sortlp   if not at limit keep looking
    ldb  #E$TblFul no available windows
     coma
     lbra  error
     EMOD
EOM     EQU  *
```

< 268'm >



Beginner's Showcase

Programmers of all skill levels will appear here. The emphasis is on short, easy to type in programs that illustrate programming techniques. Typing in examples is a great learning tool! If you have a short program or subroutine in any language, drop us a line! Any program/subroutine printed may be used by anyone within their programs, even commercial programs, as long as credit is given the author and magazine within the code (REM statements) and documentation.

Editor's note: *This article was prepared in response to Joel Sherman's letter in the 15 DEC issue. Since it addresses some beginning C issues, I felt this an appropriate place for it.*

A Squib on C and Standards

Henry Harwell

Many people- often with competing interests- have had a hand in the evolution of what we now know as the "C" programming language. After the initial development of C in the early seventies by Dennis Ritchie at Bell Telephone Labs, Bell published a programmers manual by Brian Kernighan and Dennis Ritchie, The C Programming Language. It soon became a cult classic, undoubtedly circulated in mimeo and photo copied form prior to official publication in 1978. The book was reprinted in 1991/92 by Prentice-Hall with updates to include ANSI standards. This edition is currently available by special order from most major book stores. There is very little difference between basic ANSI and original K&R C. Hold onto any bona fide first editions (1978); they are rare and have appreciated in value!

The second edition of The C Programming Language by Kernighan and Ritchie accommodates growing recognition of C and, bane to all programmers, standardization. the American National Standards Institute (ANSI) began drafting a so-called standard C in 1983- finishing in 1988. This is where ANSI C comes in, taking what was very streamlined code known best to Berkley and Bell Labs insiders and generalizing it. The major differences I have found between "pure" (K&R) C and ANSI C is in the use of function headers as per the attached example (see sidebar).

Microware C on the CoCo is faithful to the original K&R specifications. In fact, the Microware User's Guide (UG) refers the user to K&R as "the language reference manual" (see pages 1-2 in the UG, where several departures from "standard" pre-ANSI C are noted). While as a user I am profoundly respectful of the work done in developing the C compiler for Tandy's 6809 based machine, the Microware UG itself offers little incentive for learning the C language. At best, it is, and was in 1983, a deft cop-out from writing a substantial introduction to the compiler. Perhaps in Microware's defense it should be noted that their UG was never meant for the general novice, but for seasoned programmers- even those who already knew C, i.e., those who merely needed a map of the specific implementation of C on the MC6809 under OS-9. That may account for how truly unreadable the UG often is.

True, official documentation varies from skimpy to unwieldy (Microsoft's eight or nine books on FoxPro2 come to mind...). So "our" UG at least avoids Ecclesiastes' condemnation that "a big book is a big evil." Hence the Microware/Radio Shack User's Guide must be the lesser evil! At the very least, I would have preferred a traditionally bound UG- one that didn't start loosing pages after periodic use,,, but we digress!

Here are a couple of "picky points" that might help:

1. To print floating point numbers or doubles, you MUST invoke the library function pffinit() somewhere prior to the call for printf(); Similarly, invoke (call) pflinit() for longs. See the UG pages 1-7.

2. I haven't proven this, but a strobe of keyboard input MAY FAIL unless you do an ungetc/getc combination ant then proceed with manipulating your data. My C teacher, who has been around a long time, thinks this is just wacko. I have suspected that it was because I was NOT using a no-halt disk controller. I'll test this theory soon with a Disto Super Controller II in no-halt mode.

3. Another point to test concerns the use of void to signal no return value from main. My teacher starts just about every program with it, and Level II OS-9 requires the ubiquitous semi-colon for void when it precedes main.

```
K&R C                  Microware K&R C
void                   void;
main()                  main()
{                      {
 int hotdogs;          int hotdogs;
/* statements */        /* statements */
}                      }
```

I believe I have tried calling void as "main(void)..." with indeterminate results. Check the UG pages 1-2 for official differences with K&R (notably not supporting bit fields and only supporting "=+" and "=*", not the older forms).

I should note how much I benefited from the technical advice of Zak Sessions (owner of Color Systems) on Delphi. I might never have got the C compiler started had it not been for his assistance. I would recommend a measure of relief to anyone learning C by contacting and staying in touch with a few gurus before giving up on the Microware C compiler. Delphi, StGNet, or FIDO bulletin boards are excellent sources for easy and fast help.

The irony of this discussion is, insofar as "present C standards" are concerned, that today's C is rapidly phasing into C++. I am told that to much familiarity with "standard" C will actually hinder learning this newer object oriented dialect. *< 268'm >*

Functions, Anyone?

In the Kernighan and Ritchie (K&R) C format, parameter types must be declared outside of the function header (parentheses). In ANSI C, you may declare parameter types inside the header. For example, suppose you are calculating the number of drumsticks available from a number of turkeys:

```
K&R C (OS-9 Level II)               ANSI C
void;                               void megasticks (int birds, drumsticks);
megasticks (birds, drumsticks)      {
{                                   /* calculate drumsticks here */
int birds, drumsticks;              }
/* calculate drumsticks here */
}
```

That's about it. As can be seen, the ANSI revision permits tighter code. Of course, any good ANSI C compiler will be downward compatible and accept the function and parameter declarations in the K&R format.

Reviews

MM/1 68340 Review Update

Zack C. Sessions

I would like to inform you of some changes with the 68340 Accelerator upgrade for the MM/1 which have come about since the review published in the December 15, 1993 issue:

1. It has been found that another PAL needs replacing. It is not required for all users, but Kevin Pease "recommends" that it be installed. A new PAL was shipped out to all owners of current 68340 Accelerator upgrades. It is my impression that this second PAL is now being shipped with all upgrades.

2. Edition 51 of WindIO was also distributed. It fixes the Palette bug which caused changing colors in one window to affect all other windows.

3. I have now tested the /t5 port and it works just fine, at least on my system. A standard serial paddle board is required.

4. Brian White's speed setting utility, setspeed, is available for downloading from all major sources. (Includes source in C.)

5. It seems that my particular disk which came with my upgrade was one of only a few which actually came with a new shell. Kevin Pease told me that he had not intended on distributing any new shell with the upgrade and was surprised it was even there!

6. Still no sound driver... yet!

Historically BREWED

F.G. Swygert

This is the newsletter for the Historical Computer Society, a group that promotes the preservation, restoration, and use of old computers. The title refers to one of the first home computer organizations- the "Home Brewed Computer Club" of California's silicon valley (Steve Jobs was a member).

I found the newsletter to be very interesting. There were some general articles on computing as well as an article on the history of the Apple. How many of us know why the Apple I (yes, there WAS an Apple I, but only a few hundred were built) was so important in the history of computers? One is worth around $15,000 today!

Many of you. like myself, enjoy older computers, not just our beloved CoCo and 68K machines. Here one will find or be able to inquire about information on any computer ever made! Several members have letters printed with lists of their collections. One person has 49 systems! CoCos and other 680x and 680x0 systems, including some workstations, are among those listed.

Those interested in a sample copy should send $2.50 to:

Historical Computer Society
10928 Ted Williams Place
El Paso, TX 79934

Subscriptions are $25 yearly. Home Brewed is published bi-monthly.

Smash! Breakout Type Game for OS-9 Level II (CoCo)

F.G. Swygert

Smash! is modeled after the popular Steve Bjork game for DECB, Bash. Both are "breakout" type games where the player tries to knock colored tiles out at the top of the screen with a ball and paddle. The paddle moves horizontally across the screen via a joystick. Either joystick can be used by a single player (defaults to right joystick).

The game is shipped on a single sided 35 track disk, so any Color Computer 3 user should be able to get it up and running. Documentation consists of five pages of 8.5"x11" text. It is very clear and easily understood. Only one thing isn't clearly mentioned... how easy it actually is to start the game!

I'm working near the deadline right now, so had very little time to work with a program. All you really need to do to start playing is backup the distribution disk, put the backup disk in a drive, then type **chd /dx; chx /dx/cmds; smash** (where /dx is the drive number). You will be greeted first with a "loading levels" message, followed after a few seconds by the start-up screen. The start-up screen asks for number of players (1 or 2), quit, and password for level. The first two are pretty self-explanatory. The password is to prevent cheating! You can start at any level... provided you have already played to it and wrote down the password shown while that level is loading! So if you don't want to start at level one all the time, write down the eight letter password for each level as it is reached!

If you wait 15 seconds after the start-up screen, the high scores will be displayed. Wait 15 more seconds and the demo mode will automatically kick in. The demo selects levels at random, with a cute little quip about the level being demoed.

There are several additional features of this particular version worth mentioning: make your own levels with a text processor; run multiple games or other programs in other windows; auto-pause when you switch to another window; easily customized to allow running from ANY directory you choose rather than the defaults of CMDS and DD/SYS (customization requires Basic09, great for hard disk users!); easy ALT and CTRL key sequences to change defaults, start demo mode, select special features, etc.; all memory/windows released upon quitting game.

The game will run on a 128K CoCo 3, but requires a total of 96K and two free (not initialized) window descriptors. There is a technical note stating that "VRN", a public domain animation enhancement utility, is necessary for smooth animation. I don't have VRN on my system, and thought the animation very good.

What else can I say? This is an EXCELLENT, ADDICTIVE game that runs from OS-9. If you are a "Bash" or "Arkanoid" fan, and you use OS-9 a lot, then you need to order this program. Then you won't have to quit OS-9 to play, and you can play while OS-9 is churning on something else in the background (though this will affect play somewhat... VRN may be necessary in this situation). And you can modify levels to suit your own skills.

Smash! retails for $25, available from:

Northern Xposure
7 Greenboro Cres
Ottawa, ON K1T 1W6
Canada

Full source code, including the sprite library, is available for $30 (over 150K of source code, 10K documentation). Smash! was written by Alan DeKok.

< 268'm >

Disk Check

Robert Gault

A graphical floppy disk speed timer for the CoCo 3

Recently I had some problems formatting a floppy disk under OS-9. The same drive worked just fine under Disk Basic with DSKINI. I was starting to get all excited about my modified OS-9 kernal messing up the Format function. Before I went overboard, I had the thought that perhaps OS-9 is more sensitive to floppy disk rotational speed than Disk Basic.

It had been some time since I checked my disk timing, so I located a timing program and tried it; problem solved. Then I thought this program really looks nice, funny I had forgotten that I wrote it, it's good enough to publish. So here it is.

The code pretty much speaks for itself. What is interesting is that the program has a graphics format and is independent of CPU clock speed. Sorry, but it is for the CoCo 3 only.

When you type it in, be careful of line numbers. The LI variables must point to the correct data lines for the checksum routine to work correctly.

An equation in line 370 needs some explanation. Its present short state is intended to require the minimum of floating point operations. Here is a more meaningful version:

X=320+30*(SP-300)

Explanation:

index location=center of screen
+ 30 pixels per delta RPM.
SP=RPM; 300rpm is target value

Editor's note: Assembly language source code for the program will be found on this issue's microdisk. It was not printed due to space limitations. If you have Tandy's EDTASM and are trying to learn assembler, you might want to get a copy.

```
10 ' FLOPPY DRIVE RPM CHECKER
FOR  COCO3 BY ROBERT GAULT
20 ON ERR GOTO430:ON BRK GOTO
400
30MESSAGE=10:' THIS MUST MATCH
THE NUMBER/3 OF DATA ITE MS US
ED FOR THE GRAPH
40GOSUB450:DEFUSR0=&HE00: DEF
USR1=&HE30
50CLOCK=USR1(0):'ML. ROUTI NE
RETURNS CPU CLOCK SPEED 1 OR 2
MHZ
60CM=60*CLOCK*894886.3: 'CYCLES/
MIN.
70POKE&HFF40,0:HSCREEN0: CLS
80 T$="FLOPPY DISK TIMER":RW=2:
GOSUB420
90T$="by":RW=3:GOSUB420
100 T$="Robert
Gault":RW=4:GOSUB420
110T$="Select drive for testing (0 - 3) or
hit any other key to exit.": RW =12: GOSU
B 420
120 A$=INKEY$:IFA$=""THEN 120
130A=INSTR(1,"0123",A$):IFA<1 OR A
>4 THEN400 ELSE DR=A-1
140  D=VAL("&H"+MID$("292A6 96A
", DR *2+1,2)):' USE FOR DOUBLE SI
DED DRIVES
150  'D=VAL("&H"+MID$("292A2C86
" , DR*2+1,2)):' USE FOR SINGLE SID
ED DRIVES
160RESTORE:HSCREEN4: PALETTE0,
63:PALETTE1,0: POKE&HFF9 A,63170
T$ ="FLOPPY  DRIVE  TIMER": RW=1:
GOSUB410
180 T$="DRIVE #"+STR$(DR):RW=3:
GOSUB410
190HLINE(20,80)-(620,80),PSET200FOR
I=0TO600 STEP30: HLINE (20+I,80)-(20+
I,75),PSET:NEXT
210FORI=0TO600STEP150:HLINE(20+
I,80)-(20+I,73),PSET:NEXT
220FORI=0TO600STEP300:HLINE(20+
I,80)-(20+I,70),PSET:NEXT
230HLINE(185,55)-(455,55),PSET:HPRI
NT (23,6),"-1.5%":HPRINT (52,6),"+
1.5%"
240HLINE(185,55)-(185,59),PSET: HLIN
E(320,55)-(320,59),PSET: HLINE(455,55
)-(455,59),PSET
250 FORI=1 TO MESSAGE:READ CL, R
W, T$:HPRINT(CL,RW),T$:NEXT
260DATA2,11,"2",2,12,"9",2,13,"0",40,11,
"3",40,12,"0",40,13,"0"
270DATA77,11,"3",77,12,"1",77,13,"0",39,
6,"RPM"
280 T$="Target value is 300 RPM; +-1.5
% is acceptable.":RW=18:GO SUB410
290 T$="Hit any key to return to dri ve sel
ection.":RW=23:GOSUB410
300 POKE &HFF40,D
310 FOR I=1 TO 800:NEXTI:' WAIT FO
R MOTOR TO REACH FULL SPEED
320 FOR J=1 TO 10 STEP0:' ENDL ESS
LOOP
330EXEC&HE02
340HL=USR0(0):'ML. CYCLES PER
REV.
350TC=((HL)*13):' NUMBER OF CYCL
ES/REV
360 SP=CM/TC:' NUMBER OF RE VS./
MINUTE
370HLINE(X,69)-(X+1,60), PRESET, BF:
X=30*SP-8680:HLINE(X,69)-(X+1, 60),
PSET,BF
380 A$=INKEY$:IFA$<>""THEN70
390NEXT
400CLS:POKE&HFF40,0:END
410T=LEN(T$)/2:HPRINT(40-T,RW),T$:
RETURN
420T=LEN(T$)/2:LOCATE40-T,RW: PR
INT T$;:RETURN
430 HSCREEN0:CLS:PRINT"DRIVE N
OT  READY":PRINT:PRINT"HIT  ANY
KEY"
440EXEC&HADFB:GOTO70
450FORI=1TOMESSAGE:READ A,B,
C$: NEXT
510LI=550:START=&HE00: FINISH=
&H E2E: GOSUB519
512LI=600:START=&HE30: FINISH=
&HE50
519 FORM=START TO FINISH STEP10:
SUM=0
520FOR I=0TO9:READ A$:VA=VAL
("&H"+A$):SUM=SUM+VA:POKE
M+I,VA:NEXT:READ CHK: IFSUM<>
CHK  THEN  PRINT"ERROR  IN  LINE"
LI:END
530LI=LI+10:NEXT
540RETURN
550DATA 1A, 50, 9E, 8A, 31, 84, CC, D0,
2 , B7, 1180
560DATA FF, 48, F5, FF, 48, 26, FB, 31,
21, 27, 1309
570DATA 13, F5, FF, 48, 27, F7, 30, 1 ,
F5, FF, 1426
580DATA 48, 26, F9, 30, 1 , F5, FF, 48,
27, F9, 1268
590DATA 1C, AF, 1F, 10, 7E, B4, F4, 00,
00, 00, 800
595 REM  Second ml. data packet starts
here; CPU clock timer.
600DATA 1A, 50, 9E, 8A, 13, B6, FF, 02,
B6, FF, 1297
610DATA 03, 2B, F8, 30, 01, B6, FF, 03,
2A, F9, 1074
620DATA 1C, AF, CC, 00, 01, 8C, 06, C1,
25, 01, 785
630DATA 5C, 7E, B4, F4, 00, 00, 00, 00,
00, 00, 642
640This program will work correctly at
either fast or slow CPU clock speed.
```

Special Report: Is there an MM/1 in the future? *David Graham*

Editor's Note: *Don't construe the inclusion of this article as favoritism toward BlackHawk Enterprises or the MM/1. I thought it best to include this article as a news item to inform our many subscribers who have an interest in the MM/1 of what is currently happening with the computer and what the manufacturer plans for the future, especially in light of recent advertising. Other items to large for* Micro News *on any computer will be given special attention as thought necessary.*

In November of 1993, Paul K. Ward of Interactive Media Systems, Inc. and BlackHawk Enterprises reached an agreement allowing BlackHawk Enterprises, Inc., full rights to manufacture and sell the MM/1 and it's peripheral boards, as well as several other assets of IMS. Communications with software vendors had gone well, resulting in several licensing agreements, though software sales remain weak. BlackHawk had already developed sourcing for MM/1 serial paddle boards and MIDI boards earlier in 1992, and sales have been encouraging. A deal with Kreider Electronics allowing BlackHawk and it's resellers, formerly the IMS representative net, to sell the 68340 accelerator card has gone well, leading the company to feel optimistic about the chances of an MM/1 comeback.

The first action on being asked to take over production of the MM/1 was to examine the past record of the machine and the company that produced it. While the majority of MM/1 owners feel that the machine is great (it resells for up to 80% of original retail cost on the rare occasions on is sold), Interactive Media Systems, Inc. had suffered some performance problems. Paul was quite candid about the problems and the reasons they exist. Poor delivery times and lost orders were the principal problems. It was apparent that some changes in management techniques were in order. The delivery problems centered around three areas.

1. The use of a Just In Time (JIT) delivery system for a custom manufactured product was the first problem. It was compounded by difficulties in programming PAL chips for the I/O board. Shipping these chips back and forth across the country for programming added to assembly time, and cut into a slim profit margin.
2. Shipping errors by the manufacturing subcontractor was the second problem. Several shipments were sent out without PAL chips at all.
3. The third problem was the lack of contact between those who sold the products, and those who shipped them.

Because of these 3 primary problems, a 4th problem entered the picture - financing. As IMS's funds ran out, some orders went unfilled, and activity ground to a halt. This is when Paul began to look for someone to buy IMS. He was not able to find a buyer.

Paul's attempt to sell IMS stemmed from several goals. First, the feeling was that the MM/1 should go on. It is a good machine, and a worthy successor to the Color Computer it was designed to replace. Secondly, Paul wanted all of his outstanding orders to be filled. Finally, he wanted to receive some value for his stockholders. I sympathized with all those ideals. But I could not afford them. As the others did, I turned down the offer to buy IMS. Instead , I offered to buy the corporate assets. To secure the note, I offered stock in BlackHawk Enterprises, Inc., and incorporated my business on November 17th, 1993. This arrangement allowed both of us some benefits.

First, I gained an existing machine and operating system licenses to sell. Next, I received a listing of a substantial amount, though not all, MM/1 owners. In exchange, I gave Paul and IMS a note secured by a non-controlling interest in my new corporation, and the right to buy MM/1 products at the same discount as my other authorized resellers.

Thus, Paul realized all of his goals for selling IMS, though the delivery on his second goal will take quite a bit longer than it would with an outright sale. Of course, with limited capital, I could not afford the liabilities such a deal would have required. So, that brings us to the present, with the MM/1 under the stewardship of a new company.

On contacting the manufacturing subcontractor, I found that 50 Extended systems was in inventory at their plant. Those systems are complete, minus EPROMS and PAL chips. So, I arranged for someone to program those chips, and announced a sale.

Because I was unable to get a loan to buy these systems in advance without showing the bank some proof of interest, I am offering a substantial discount to those who are willing to send a $10 deposit and wait up to 60 days after I arrange for a loan to buy these MM/1 systems (see ad on page 3!). Profit from these sales will be used to fund a standing inventory of MM/1 systems, allowing us to finance ourselves with minimal interest expense, and keep prices down. We are working on our first catalog, and continuing to sell the 68340 boards manufactured by Kreider Electronics.

I'm looking forward to hearing from the OS-9 community about what YOU think about the MM/1. Please, write today! Our address is :

BlackHawk Enterprises, Inc.
PO Box 10552
Enid, OK 73706-0552
(405) 234-2347

I can be reached via Internet as nimitz@delphi.com, and on DELPHI as NIMITZ. I access that account almost nightly, but I reserve the right to reply by regular mail as necessary to control costs. The editor has offered me an MM/1 support column. In the next issue, I'll address other issues of importance to MM/1 owners.

Micro News

Epson America has developed the smallest motherboard in the world. An entire 386SX PC motherboard has been reduced to a single chip! The package is about the same size as a PCMIA card or credit card calculator - 3.4"x2.1"x.22" (85.6x54.5.5mm). It features a 386SL and support chips, VGA controller, floppy controller, keyboard controller, ROM (BIOS), and 1-4MB or RAM. The card uses a proprietary 236 pin connector, the Epson All-In-One System Interface (EASI), to connect to ISA peripherals and I/O ports. Epson plans to license the technology to system manufacturers. A PC could easily be produced on a simple double sided board with standard connectors only, and maybe an expansion slot or two. Now if Motorola would do something like this with a 680x0 chip, anyone could design a simple double sided board and build there own OS-9 compatible computer!

Elitegroup Computer Systems, Inc., has come up with a neat way to improve CPU performance. They have developed a hardware/software combination (dubbed "PowerShift") that speeds the CPU clock up for a short period. The system clock is replaced with a voltage controlled oscillator. Software control speeds up the oscillator by as much as 22% for short periods (such as boot-up, calculating a spreadsheet, displaying a large graphics file, etc.). Chip makers point out one problem with the technology- they won't guarantee the CPUs to run over the tested clock rate. Speeding up the CPU above the tested rate usually means a ffaster heat build-up, which is why it shouldn't be used for extended periods. This method could be used on any motherboard design. It is possible that the CoCo clock could be replaced with a voltage controlled oscillator and switched slightly higher, but one would lose the video signal while the speed-up option was in use, since the CoCo clock controls video as well as CPU speed.

Apple started production of PowerPC based Macs early in January. Developers with pre-production machines have been very impressed with the units, even though portions of the Mac operating system (System 7) is running in 68040 emulation mode. A graphics application running on one of the pre-production models and a Pentium based machine (ports of the same program) ran over 50% faster on the PowerPC machine. The first three systems, scheduled to be introduced in March, are targeted for mid and high range applications. The lowest price models (around $2000) will sport a 60MHz MPC601, 8MB or RAM, and a 160MB hard drive. Monitors and keyboards are priced separately on all Macs. PC emulators should run as fast as mid level 486 systems (486SX/33MHz) on the PwerPC Macs, and Apple hopes to win over some customers with 286 and 386 systems waiting to upgrade. This way they get a decent 486 and a Mac for less than the price of both, and they share resources and take up less desk space. When I open my printing shop in the next year or so, I want a MPC601 Mac in my office...

More news from Apple... they have developed a larger Newton MessagePad. This new model, which should be available around April, is 8.5x11x1 inches. It is targeted toward data collection and other corporate uses where the larger size is more practical. An upgraded Newton is also in the works. It should have several features missing on the current model and be about $100 lower. It won't, however, be available any time soon.

Speaking of the Newton, the Department of Defense has purchased a couple hundred to be tested in the medical field. I suppose they could easily replace clipboards and paper files as used in some hospitals today. The larger model would certainly be more practical in my opinion. Just plug the serial port into a master station, up or download the day's data, and be off! Heck, the head nurse could even code in the data that each nurse should have, and when they connect their MessagePad and key in their code, the transfer could be done automatically!

Everyone who keeps up with computers in general knows that the Intel 486 and Pentium CPUs run very hot. They usually come with a heat sink attached to the CPU (my 486/50MHz did!). One can also buy clip-on heat sinks with small fans attached. Someone was making a special heat sink with a selenium fluid in it to get rid of higher heat loads than a standard aluminum heat sink. Well, now an air conditioner has been developed! The system uses a sealed evaporator tray that clips to the CPU like a heat sink. A special fluid fills the tray. Heat from the CPU evaporates the fluid, taking away heat in the process. The vapors flow (via convection) into a condenser mounted away from the CPU, where they cool and return to a fluid. The fluid returns to the sealed tray by gravity flow. The fluid is a special low boiling point concoction made by 3M. I wonder if the fluid is non-conducting though? Sure would be nice... what if you computer sprang a leak?

MicroSoft has announced that a version of Windows NT will be available for the PowerPC late in 1994. IBM will be porting 16 bit Windows 3.1. Windows on the PowerPC 601 (60 MHz) is said to run about twice as fast as on an Intel 486DX/33MHz. And this in partial emulation! Windows NT from the original writer should sing along really fast!

Apple Computer's software subsidiary, Claris Corp., has announced that ClarisWorks, an integrated software package, will be ready when the PowerPC based Macs appear. A smart install procedure will determine if the software is running on a 680x0 or PowerPC based Mac and install the appropriate code segments. Insignia Solutions Inc. announced that it's SoftWindows emulation package will also be available for the new machines, allowing System 7 and MicroSoft Windows (and the Mac version of OS-9) to all run on the same platform. Metrowerk Inc. of Canada have also announced a PowerPC product- a C/C++/Pascal compiler package that compiles code for System 7 running on the 680x0 and PowerPC platforms. The high end edition includes a cross compiler that generates PowerPC code from a 680x0 platform.

Neil Brookins' Bible Accordance program is ready for the CoCo 3! His phone numbers are:

work: 215-233-8517 (has an answer machine)
home: 215-542-2348

Call his work number first, if he's not there just leave your message and he will call you back. Neil also has an OSK version, from which the CoCo OS-9 Level II version was developed.

Frank Hogg Laboratory announced the availability of a fully supported version of G-Windows for the MM/1. The version being offered is the current version, with a free upgrade to version #51 when it becomes available (should be shipping by the time this gets out). 640X400 and 768X420 resolutions are supported. G-Windows for the MM/1 is $200 The Development Package is $290

Frank Hogg Laboratory, Inc.
204 Windmere Road
Syracuse NY 13205
315-469-7364

< 268'm >

If you have new soft or hardware products, let us know! We will gladly print a free blurb for you here in MicroNews whether you advertise or not (though we will be happy to have your ad also).

THE SWAP SHOP

Buy, sell, trade hard/soft ware!
If you have something you no longer need, need something you can't find, or just have some extra "stuff" lying around, this is the place to find or get rid of it!

Ads are free. **Only those vendors with at least a 1/4 page ad may place classifieds without special permission.** For these, short ads will be free as space permits. PLEASE drop us a post card or E-mail if/when an item sells or is found!

FOR SALE

Multi-Pak, 26-3124 (small), upgraded for CC3- $70; OS-9 Level II- $35; Multi-Vue-$10; Koronis Rift-$10; Rogue-$10. Shipping included to US. **John Gar, RR1 Box 141, Newel, SD 57760**

Replacement 68B09E for all CoCos. $7 each includes S&H. Timothy D. Boos Sr., Rt. 1 Box 155, Peculiar, MO 64078

SALE OR TRADE

CoCo3, 512K, CM-8 RGB monitor, FD-501 disk drive (2 drives), Ken-Ton SCSI hard drive interface, Y-cable, RGB-DOS in ROM, 40MB hard drive (factory RGB-DOS setup for DECB), lots of software (send SASE for listing). Will take best offer. Alan Clarey, 1012 Borrette Lane, Napa, CA 94558

26-3124 MPI - $45; Tandy RS-232 Pak - $30; 2 Touch Pads - $20 (or trade for pair of deluxe joysticks); **V-Term - $15; VIP Writer - $15;** Prices include S&H in the US or Canada. George Demers, 603-726-3638

I have approx. 24 never used Tandy CoCo3 programs for sale or trade for CC3 soft/hardware. Send SASE for list. John R. Mott, Jr., Box 26246, Phoenix, AZ 85068-6246

WANTED

Tandy OS-9 C Compiler with documentation. Contact Joe Charbonneau 527 Jarvis St., Windsor, Ontario N8P 1C8 (Canada); Phone 519-735-8630

Used CoCo hard/software. Will buy by piece or entire collections. Rick Ulland, 449 South 90th St., West Allis, WI 53214

DISTO Products

Quality hardware for your Color Computer!

2MB Upgrade (no RAM) - $99.95
Mini Disk Controller - $70
Super Controller I - $100
Super Controller II - $130
MPROM Burner - $50
3-N-1 (parallel, RS-232 RTC) - $75
SASI/SCSI Hard Disk Adapter - $75
Full Turn of the Screw - book with all Tony DiStefano's Rainbow articles, January 1983 to July 1989 - **$20**
Complete Schematic Set - all DISTO product schematics except 2MB upgrade - **$20**

Include $2 S&H for book or schematics. Hardware S&H is $4.50 for one item, $6.50 two or more. Certified check or International Money Order only!

DISTO
1710 DePatie,
St. Laurent, QC H4L 4A8
CANADA

Bob van der Poel Software

Great stuff for your OS-9 Level II system!

Ved Text Editor	$24.95
Vprint Text Formatter	$29.95
OS-9 character Set Editor	$19.95
OS-9 Disk Mailing List (DML9)	$24.95
Basic09 Subroutine Package	$24.95
Cribbage	$19.95
Ultra Label Maker	$19.95
Magazine Index System	$19.95
RMA Assembler Library	$19.95
Stock Manager	$24.95
OS-9 Public Domain Disk	$9.95

(see DML9 review in this issue!)

All our programs are in stock for immediate shipping. Please include check or money order with your order. Sorry, no credit cards; but will ship COD to US and Canada (we add a small additional charge to cover the post office COD fee). Mention this ad and get FREE SHIPPING (normally 5% or $2 minimum)! All orders are shipped via first class mail, usually the day received. Write or call for free DECB or OS-9/6800 catalogue.

P.O. Box 355	**P.O. Box 57**
Porthill, ID	**Wynndel, BC**
US 83853	**Canada V0B 2N0**

Telephone (604) 866-5772

for all your CoCo hardware needs, connect with

CoNect

449 South 90th Street
Milwaukee, WI 53214
414-258-2989 (after 5pm EST)

Mini RS-232 Port: Don't let the name fool you! This is a full featured serial port, supporting the signals needed for flow control as well as the basic 4. Jumper blocks allow readdressing or swapping DSR/DCD. No custom cables or hardware widgets needed here! Y cable users will need to add $9.95 for a power supply. **$49.95**

XPander: Don't you think the CoCo would be a lot nicer without all that mess hanging off the right side? Of course it would! Our XPander allows mounting two SCS decoded devices (like a floppy and hard drive controller) inside your CoCo. Built-in no-slot RS-232 port is similar to our "Mini" described above. The external cartridge connector is still present, and can be configured to run games or as an additional hardware slot. Kit includes new lower case shell and 12V power supply. Board only is great for use in a PC case!
Kit: $124.95 Board Only: $99.95

Hitachification: CoNect will install a Hitachi 63B09E CPU and a socket into your CoCo. Machine MUST be in working condition! The 68B09E will be returned unharmed. 90 day limited warranty. Chip and installation only **$29.95**

REPAIRS: We can repair most damaged CoCos, even those with bad traces where a 68B09 was removed. Costs vary with damage. Bad 68B09 sockets repaired for only $40! **Inquire BEFORE sending your computer.**

New Lower Prices! from ColorSystems

Variations of Solitaire

Includes FIVE variations; Pyramid, Klondike, Spider, Poker and Canfield. Complete documentation shows how to create your own games boot disk using the special menu program which is included.

CoCo 3 Version $29.95
MM/1 Version $39.95

WPShel

A Word Processing Oriented Point and Click Shell for all your word processing needs. Requires WindInt from your Multi-Vue disk. Does not include Editor, Formatter, or Spelling Checker.

CoCo 3 Only! $20.00

We accept Personal Checks or Money Orders drawn from US Banks or International Postal Money Orders.
NC residents please add 6% Sales Tax.
Call or write for FREE catalog!
Please add $3 per item for shipping outside of the Continental United States.

NEW!

Using AWK With OS-9

A description of the AWK Programming Language with an emphasis on GNU AWK for OSK. Includes the latest version of GNU AWK.

OSK ONLY! Just $19.95

OS-9 Game Pack

Includes FIVE complete games; Othello, Yahtzee, Minefield, Knights Bridge, and Battleship. Includes special menu and step by step instructions on creating your own games boot disk.

CoCo 3 Version $29.95
MM/1 Version $39.95

All CoCo 3 Programs require at least 256K of memory.

Coming SOON! Indexed Files for OS-9 Level 2, OS-9/68000, and OS-9000!

ColorSystems
P.O. Box 540
Castle Hayne, NC 28429
(916) 675-1706

Quality OS-9 Software for the Color Computer 3 and the MM/1 from IMS

Quality CoCo Software and Publications from FARNA *Systems...*

ALL NEW! -- **DigiTech Professional Sound Recorder**
Record ANY sound on disk with your CoCo! Easily add sound effects/music to BASIC or M/L programs. Sound input through joystick port with easy to make cable (cable instrucs. included, or order separately). **$20** cable- **$6** w/software, **$8** later

OS-9 Quick Reference Guide II & PatchOS9 Disk Set
New, revised edition of the popular QRG now has Basic09 error codes added to it. Also contains commands, system calls, error codes, everything needed in a handy desktop size. The PatchOS-9 disk set includes the most popular and useful patches and replacement commands for CoCo OS-9 Level II.
QRGII/PatchOS9 set-**$13.50**; QRGII- **$8.50**; PatchOS9 - **$7.50**

ADOS - the premier Disk Basic enhancement! Adds support for double sided drives, 40/80 tracks, faster formatting, many more items. Extended ADOS3 has built in RAM disk, even more!
ADOS (CC 1 & 2) - **$15**; ADOS 3 - **$25**; Ext ADOS 3 - **$30** (ADOS3 required); ADOS3/Ext ADOS3 Combo - **$50**

CoCo Family Recorder - absolutely the *BEST* for Genealogy!
Prints many different reports. Tracks up to 550 persons, any number of generations. Requires 80 column monitor and two disk drives. No special order for information to be in, easy to use. **$20**

Programming the 6809 - Zaks & Labiak, teaches m/l program-

FARNA *Systems*
Box 321, Warner Robins, GA 31099-0321
Phone 912-328-7859; E-mail dsrtfox@delphi.com
add $2.50 S&H, $4 Canada, $7 Overseas

INTRODUCING: CoCoTop,
Desktop Program for OS-9 Level II.

Now it has become much easier to use OS-9 on your CoCo. Most of the functions that you carry out by typing command lines at the OS-9: prompt have been built into this program that sports some 60 functions in a text based point-and-click interface.

If you have to copy or delete a file, you no longer have to type a filename or pathlist but you highlight the name on screen with you mouse or joystick, press a button, and watch your CoCo do the work.

There are also a number of functions you will not find in standard OS-9: a "safe" format along with an "unformat" command that will let you recover data from a formatted disk. For files: undelete, move, a file finder, and overwrite capabilities for the copy command. For directories you will find sorting of contents and you can move a directory if you wish.

There are functions to track your system: configuration and changing device descriptors for disk and printer. Maps of how your computer uses it's memory and disk space. CoCoTop can set up windows for you and launch other programs. It even has a menu where you can link your favorite programs to it. Of course it also has a built in calculator and notepad.

For people who have problems with OS-9's disk structure the good news is that the program presents that directory structure as a series of point-and-click menus: keeping track of where you are and where you can safely go. It also displays the pathlists used so you can learn as you go along.

Requires 512K CoCo 3. There are two versions: 1.0 is the complete package. 1.1 is for those who already have TOOLS 3 since CoCoTop uses some of the TOOLS 3 utilities.

CoCoTop 1.0	$24.95
CoCoTop 1.1	$19.95
CoCoTop 1.1 + TOOLS3	$34.95
TOOLS3 1.1	$29.95
QUICKLETTER 2.0	$19.95
ACCOUNTING LEVEL II	$34.95
INVESTING LEVEL II	$24.95
LEVEL II GRAPHICS	$34.95

UPGRADES from earlier versions are only $5 (return original disk)

Christmas Sale: 20% off two packages, 40% off three or more!

Level II Graphics Demo: $5 (+$3 S&H). Full amount credited toward purchase of full working version.
RScopy/OScopy: Transfer between OS-9 and Disk BASIC with no special disk drivers or patches. $10 (+$3 S&H).

C. Dekker ... *User-friendly Level II Programs!*

RR #4 Centreville, NB
E0J 1H0, CANADA
Phone 506-276-4841

Shipping+handling: US/Canada $3.00 all others $5.
Prices in US dollars Send cheque or money order NO COD'S.
Call or write for Canadian dollar prices.

C. Dekker

This Issue's "microdisk"

OS-9 Content:

CHKCHILD.C	GetNW1.ASM
DATAMOD.C	GetNW2.ASM
PLAYBELL.C	Shellplus (also .DOC file)

Disk BASIC Content:

DISKTEST.BAS	PUZZLER.BAS
DISKTEST.BIN	WELL.PUZ
DISKTEST.ASM	NUMBERS.PUZ

*Note: Puzzler is a game for the CoCo 3. A picture is scrambled and the player must assemble it. The two .PUZ files go with PUZZLER.

"microdisk" is available for $40 per year , $21 for six months, or $6 per issue. Overseas must add $10/year, $5/six months, $1/single issue for air mail delivery. "microdisk" is NOT a stand-alone product, but compliments "the world of 68' micros" magazine.

Disk BASIC users, don't criticize the quality of programs and articles... DO SOMETHING ABOUT IT! See "Submitting Material", this page.

Submitting Material to 68' micros

FARNA Systems retains rights to print and distribute any and all contributions. The submitter retains rights to distribute, but not to print in another publication without consent of FARNA Systems unless other arrangements are made.

We accept program submissions in any programming language for DECB and OS-9 (6809 & 68000) of any type (games, utilities, etc.). Articles are accepted covering any aspect of Motorola 68xx and 68xxx processors. This includes microcontroller projects as well as alternate operating systems. If there are enough subscribers interested, we will begin accepting programs for alternate operating systems as well.

Submissions should be sent on disk in ASCII and executable formats. A printed listing should also be included if possible. A letter describing the program or article is also necessary. Submissions can be made to DSRTFOX on Delphi, or dsrtfox@delphi.com via Internet.

Media accepted: 5.25" disk in CoCo OS-9 (35/40T, SS/DS), IBM (DD/HD), or DECB (35/40T). 3.5" in IBM only (DD/HD)

Your Ad Should be here!

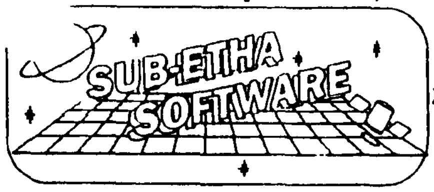
Advertising budget tight? Don't think you can afford to advertise? Well, you can't sell if you don't!
You can advertise in "the world of 68' micros" for as little as $10 per issue !
NOTE: These rates for first time advertisers only. Limited to three insertions. See inside front cover for regular rates.

MultiBoot by Terry Todd & Allen Huffman
Now have up to SIXTEEN bootfiles on your startup disk!
Hot off the assemblers and compilers is a great must-have utility which lets you have up to 16 bootfiles on one disk! No more boot disk floppy-swapping! MultiBoot will install itself to a cobbled boot disk and, upon typing "DOS", will greet you with a scrolling menu of available bootfiles!
OS-9 Req: CoCo3, OS-9 Level 2..........................$19.95

Towel by Allen C. Huffman
The first EthaWin program - a disk utility for OS-9.
A program no intergalactic hitchhiker should be without! Use a mouse or keyboard hot-keys to perform common file and disk commands from pull-down menus. Tag multiple files for Delete, Copy, Rename, etc., and even have point 'n click disk Backup, Cobbler, Dcheck and other commands. User menu lets you specify up to seven of your own commands to execute. Runs under the EthaWin interface on a high-speed text screen. All commands/colors configurable.
OS-9 Req: CoCo3, OS-9 Level 2..............................$19.95
OS/K Req: MM/1 or K-Windows Compatible............$24.95

1992 CoCoFest SIMULATOR by Allen C. Huffman
Graphics "adventure" based on the 1992 Atlanta CoCoFest
The next best thing to having been there! Digitized graphics of the event and a text command parser (ie, "get the box of disks") let you see all the vendors and even run into some famous faces of the CoCo Community. The show area, seminar room, and portions of the hotel are all represented. No true "goal", but you do have to figure some things out, like how to get into the show and how to buy items from vendors. Runs on a 640x192 hi-res graphics screen.
OS-9 Req: 512K CC3, OS-9 Lvl 2, 490K Disk Space...$9.95
OS/K Req: MM/1 or 100% K-Windows Compat.......$14.95

Worlds at War: *A complete wargame simulator package!*
Finally, this Canadian masterpiece is available. Icon editor lets you build full color game pieces. Map editor lets you put together a multi-screen playfield. Options such as pass, move, attack, status, cargo, search, and build make this game a real "blast".
OS-9 Req: 512K CoCo3, OS-9 Level 2..........................$SOON!



**P.O. Box 152442,
Lufkin, TX 75915**
Please include $2.50 S&H per order

ADVERTISER'S INDEX:

BlackHawk Enterprises	*3*
Bob van der Poel Software	*29*
C. Dekker	*30*
Color Systems	*30*
CoNect	*29*
Delmar Company	*BC*
DISTO	*29*
FARNA Systems	*30*
Frank Hogg Labs	*16,17*
Infinitum Technology	*20*
Jim Bennett	*15*
Mid Iowa & Country CoCo	*8*
OS-9 Underground	*3*
Robert Gault	*8*
Sub-Etha Software	*31*

Don't have a subscription yet?
WHAT ARE YOU WAITING FOR?!
Subscribe today!
the world of
68' micros
(Details inside front cover)

For superior OS-9 performance, the

SYSTEM V

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

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

G-WINDOWS benchmark performance index is 0.15 seconds faster with a standard VGA board than a 68030 running at 30 MHz with ARTC video board (85.90 seconds vs 86.05 seconds).

Or, for less demanding requirements, the

SYSTEM IV

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

Both computers provide flexible screen displays in the native mode with the optional VGA card.

Eight text modes are supported; 40 x 24, 80 x 25, 80 x 50, 100 x 40, 132 x 25, 132 x 28, 132 x 44, and 132 x 60. Foreground, background, and border colors are user selectable from up to 16 colors.

Eleven graphics modes are supported; 640 x 200 x 16, 320 x 200 x 256, 640 x 350 x 16, 640 x 350 x 256, 40 x 480 x 16, 640 x 400 x 256, 800 x 600 x 16, 640 x 480 x 256, 1024 x 768 x 16, 800 x 600 x 256, and 1024 x 768 x 256

Text and graphics modes may be selected by a utility provided, MODESET, by software using SetStt calls, or by termcap entries. In the text mode, the screen responds to standard VT100 control sequences. The full character set from Hex 20 through Hex FF is supported in the text modes up to and including 100 characters wide. The upper 128 characters follow the 'IBM Character Set 2' popular with many terminals and printers. These may be displayed on the screen by using the 'Alt' key and one or two other keys (software permitting).

Optional G-WINDOWS provides three screen resolutions: 640 x 480 x 256, 800 x 600 x 256, or 1024 x 768 x 256. You can have two full size 80 x 25 windows with room to spare. Or, a window as large as 122 x 44 using the large fonts or one over 180 x 70 using the small fonts

delmar co

PO Box 78 - 5238 Summit Bridge Road - Middletown, DE 19709
302-378-2555   FAX 302-378-2556