January/February 199[illegible] Vol 4 [illegible] $4.00US (North America)

# the world of 68' micros

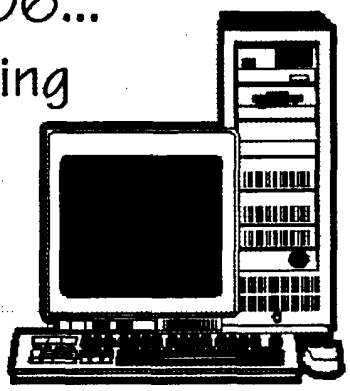
*Support for Motorola based computer systems and microcontrollers [illegible]stem*

IBM®

# LICENCES OS-9!!!

## What does Big Blue want?

Building the AT306...
as easy as following
the instructions!



## CONTENTS



**Does anyone remember Dennis Kitsz of Green Mountain Micro, Color Computer Magazine, and Undercolor notoriety? He's baaacccckkkk....** *(see page 4)*

**Build the fastest CoCo in the world...**

**A 4MHz accelerator circuit for any 6809 or 6309 system...**

**No special software! Compatible with all existing progams!!!**

*POSTMASTER:*
If undeliverable return to:
FARNA Systems PB
Box 321
WR, GA 31099

*If your address is incorrect, send me a postcard!*

*the world of 68' micros*

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

**Editor:**
*Francis (Frank) G. Swygert*

**Subscriptions:**
US/Mexico: $24 per year
Canada: $30 per year
Overseas: $50 per year (airmail)
Back and single issues are cover price. Canada add $0.50 per issue ($1.00 max) additional shipping, overseas add $2.00 per issue. Bulk/newsstand orders available.

**microdisk:**
Companion to magazine. Contains program listings, shareware, and freeware as mentioned in magazine text. Three issues per year.
US/Mexico: $20 per year ($8 single)
Canada: $25 per year ($10 single)
Overseas: $40 per year ($15 single)

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

**Contributions:**
Any and all contributions welcome. Submission constitutes warranty on part of the author that the work is original unless otherwise specified. Publisher reserves the right to edit or reject material without explanation. Editing will be limited to corrections and fitting available space. Authors retain copyright. Submission gives publisher first publication rights and right to reprint in any form with credit given author.

**General Information:**
Current publication frequency is bi-monthly. Frequency and prices subject to change without notice. All opinions expressed herein are those of the individual authors, not necessarily of the publisher. No warranty as to the suitability or operation of any software or hardware modifications is given nor implied under any circumstances. Use of any information in this publication is entirely at the discretion and responsibility of the reader.

**All trademarks/names property of their respective owners**

**The publisher is available via e-mail at: dsrtfox@delphi.com**



# A message from the editor...

You gain a few, you lose a few. That's how it goes with subscribers. Over the last three issues, there have been only a few loses. I guess that means we are getting down to the real core of CoCo and OS-9 users out there. Luckily, for the few who did leave, a few new subscribers came in.

I'm sure there are many who decide they can get whatever support they need over the Internet. They will miss the ads and articles and programs sent in by readers though. Some people ask me why I don't put the magazine on "the web" (meaning, of course, the Internet World Wide Web). I could. I have web page space. I have the capability of putting together pages that could be read with a simple text browser. It takes time and money to put together a publication though. I don't get "paid" much, but I do get a token sum for my work. I wouldn't be able to do it if it didn't at least pay for itself... I couldn't justify taking that much time out of my week for something that was basically losing money.

Charging for Internet content is a dilemma facing a lot of companies right now, both large and small. If I could charge for access, it would save me a tremendous amount of time and effort. No printing, keeping an address database, stamping, stapling, etc. All would be unnecessary! And I could probably attract a few more readers and charge less.

But there would still be those out there who don't have easy and inexpensive Internet access. I want to reach those people. And a magazine is handy... you can take it anywhere and read it! You can even read it when your phone connection or computer is down! Now isn't that convenient?

I never did like "disk based magazines". They are easy to copy and pass around for one thing, and darned inconvenient for another. I like to sit and relax when I read... in my recliner or on the couch... NOT at my desk over a keyboard! I spend ENOUGH time there as it is!!

Sure, you can print out the articles and read them later. But it isn't the same as getting that awaited item in the mail every other month.

I'll be printing for another year, that you can count on. If the recent trend is accurate, I'll be printing for another year after that. So you should have two more years of 68' micros easily. If I find a way to increase circulation without alienating my loyal readers, you will have even more years to come. I am always open to suggestions!

On this note, I had considered adding support for other orphaned computers. I got no response from the Atari ST crowd. I did get some interest from the Atari 8 bit (6502 based) crowd. I told them if they could get me 20 confirmed subscriptions, I'd add support. there wasn't a whole lot of response... but there is one fellow over there who wants to give it anther shot (attempting to get enough interested people).

I have reasoned why I didn't get much response from the Atari ST folks. I sent inquiries over the Internet. Now many of the Internet folks think they have all the support they could use. Why do they need a magazine? And many are just plain cheap... they want support, but they want it free! Well, you get what you pay for sometimes. There is a lot out on the 'net, but not everything. Otherwise there would be no computer magazines at all.

I would like to personally thank the following people for renewing their subscriptions:

| | |
|---|---|
| W.R. Hamblen | Earl M. White |
| Rodney Hamilton | Ray Shuter |
| Vern Larson | David Keefe |
| Art Boos | L.T. Day |
| J.G. Owen | Patrick Tully |
| Ted Willi | E.J. Haas |
| Monk Repair | |

I'd also like to thank these people for their new subscriptions:

| | |
|---|---|
| Paul Shubel | Don E. Peters |

I haven't gone through the mailing list yet, but I'm sure there are one or two whose subscriptions expired with the last issue, and a few whose will expire with this one. Please check you mailing labels! I don't want to miss any of you over the next couple months! I always send one extra issue with a reminder hand written on the label, but that is the last warning before your subscription lapses. Maybe I should print an "obituary" for those who have not renewed subscriptions... no, if they choose to forego this vehicle of CoCo/OS-9 assistance, that is their business. Mine is keeping it going for those who still desire it...

Many of you are now using Intel based PC compatibles as well as CoCos and OS-9/68K based machines. Would some PC news and maybe a help column be welcome in these pages? I don't mean to become a PC magazine, but to support hardware issues for those running CoCo/Dragon emulators and cross development systems for 68xx/68xxx systems. I would like some feedback from subscribers on this. Please let me know if this is a desirable feature, or if you would rather the couple pages be devoted to strictly CoCo and OS-9 issues.

**Thank you all for your continued support. Please have a great 1997!!**

# Messages from our readers...

### Microware taken to task...

I have to comment on Boisy Pitre's "educated opinion" about Microware's OS-9. Everyone who reads your magazine knows that Microware has done a lousy job of supporting the OS-9 hobbyist. The last version of OS-9 for the CoCo (Level Two) came out in 1986! Ten years of NO product support speaks loudly about how completely Microware abandoned the OS-9 hobby market.

If Microware has benefited so much from the eight or so programmers hired from the CoCo/OS-9 underground, just think of the pool of talent that might have developed over the years if Microware had actively supported the hobbyist market. Instead of eight qualified OS-9 programmers, there might easily be hundreds! Microware could have easily written off any losses incurred by a small OS-9 consumer division, while still focussing its efforts on the OS-9 industrial controls market.

One final point: people don't love Ford Motor Company because "it has targeted and is acquiring a potentially huge market". People like Ford because they can climb into their Mustang and go for a drive. Likewise, I don't see how any OS-9 hobbyist would get a kick out of knowing that "OS-9 will be on devices that we could have only dreamed of 8 or 10 years ago". If you can't get behind the keyboard and work with it, OS-9 becomes about as exciting as a toaster or set-top box.

Ted Willi
Box 447
Athens, GA 30603

*Ted, all that was pretty well said! I have to mention, however, that Microware has nothing to do with a version of OS-9 once it is licensed for a particular platform. Tandy, of course, has the license for the CoCo and is therefore solely responsible for any developments.*

*Discounting the CoCo, Microware would have benefited from supporting a hobby market. There is no need for me to repeat the benefits, you've done a terrific job already! I doubt it would have been profitable for them to directly support a hobby market, but indirect support could still be accomplished. Something like special licensing arrangements for "hobby" versions.*

*There are problems associated with hobby support. Things like not cutting into industrial markets. What would prevent an industrial user from purchasing hobby machines as cheap controllers? Wouldn't work for everyone due to harsh environments, but in some instances it would. Or an industrial user could buy hobby machines and repackage them. Seems trivial to you and I, but it could mean a lot of lost dollars to an industrial supplier who has invested much more money than the hobbyist in development tools and licenses.*

*Still, I think most of these problems could be overcome in one way or another. Microware would probably benefit more than they would lose with more public interest in, or at least knowledge of, their operating system.*

### Running DECB under OS-9...

I really enjoy your magazine. Seems like I don't spend as much time on CoCo's any more as I have gotten into PCs also. But I still work with my CoCos from time to time.

Now I have a question. I'm in the process of setting up a CoCo hard disk system. Is there a way to run DECB while under OS-9 Level II?

Ron Shively
820 Garden Street
Beatrice, NE 68310

*The short answer is yes... and no. FARNA currently carries a product written by Alan Dekok called "Rusty". This allows you to store DECB programs on an OS-9 hard drive and launch them. OS-9 is put into a "suspended state" after the DECB program starts executing. So if the DECB program expects to find data on a floppy, it will still have to be there. It can't store data on the hard drive. When the DECB program is exited, OS-9 reboots. This is very similar to Windows 95, which does basically the same thing when it starts a DOS program in DOS mode. See the FARNA ad for details.*

### Long live your CoCo...

I found the recent article on using the EZ-135 removable hard drive to be most useful and very timely as my old 20MB drive expired. How about an article on making the CoCo last a little longer, such as cleaning expansion port contacts as well as all the controllers/ROM paks. I found that older/cheaper disks are developing errors. Personally, I am copying all of my old software to new, "better" disks. I also found a need to clean all the ports– joystick, cassette, etc., with a good contact cleaner.

I have enjoyed every issue of "68' micros". Keep up the good work.

Vern F. Larson
598 Riverton Avenue
Winipeg, MN R2L 0P1
CANADA

*Vern, you made some good points! I'll add some to that now. The contact cleaner or tuner cleaner can be purchased at an Radio Shack or other electronics supply store. Either works equally well and doesn't need to be wiped off. WD-40 should probably be avoided, as it leaves a protective film that could cause problems later. Alcohol works equally well, but is hard to get into the contacts. Now this will sound strange, but you can take your entire computer and throw it in the dishwasher. Use a little mineral spirits instead of soap. I worked at a computer store in Okinawa years ago, and we did this on a regular basis. Put the machine in upside down so it will drain thoroughly and DO NOT use the drying function of the washer... gets hot enough to warp plastic such as used on computer cases! Give the computer plenty of time to dry before plugging it back in... I suggest 12 hours or so. Works wonders on dirty keyboards! A little soap won't hurt, but go light on it. If the keys are really dirty, scrub them with a toothbrush and some glass cleaner BEFORE throwing in the dishwasher. Hosing off with a water hose works in the absence of a dishwasher. Getting circuit boards wet is okay, just don't let them soak in water... don't stick it in a bucket overnight!*

*Some people have brought up the subject of hard water. The mineral deposits could cause shorts in rare cases. If you have hard water problems, be cautious! I have never heard of a problem with this, but it is a possibility.*

*If you are experiencing "sparklies" or intermittent problems, cleaning all the connectors may alleviate the problem. Don't forget to remove the GIME chip and clean the contacts on it and in the socket also! Just be careful when removing and inserting the GIME... and make sure it is put back in the same way it came out.*

*Another idea is to get a 6309 or 6809 processor chip, and a spare CoCo. There are currently plenty 6809 and 6309 processors on supplier shelves, but they are no longer being manufactured (according to an Hitachi rep). In a couple years they may be hard to come by. With the exception of the GIME, all the chips from a CoCo 2 will work in a CoCo 3, or they are common chips. The GIME rarely give problems or blows. The DAC chip (SC77526) that controls the joystick ports is another story, especially if you are using the joystick ports for input other than with joysticks. The SALT (SC77527) chip is also custom for the CoCo 2 and 3. It controls 5V power, the cassette port, and the RS-232 port. The custom keyboard controllers in the CoCo 2 & 3 are nothing but slightly modified 6822 PIAs. A standard 6822 or 6821 can be used in their place (the 6822 will handle higher loads than the 6821). The custom chips have resistance built into them on the keyboard input lines. This is to help in rare occasions where there is some keybounce. I have talked to many people who have replaced the keyboard controllers with 6821 and 22 PIAs directly with no noticed keybounce or other ill effects. PIAs and other 68xx series chips will continue to be made for some time. They are used with eight bit 68xx based embedded controllers.*

# IBM Licences OS-9!!!

Frank Swygert

## What does Big Blue want with OS-9? What does it mean for us (if anything)?

**FOR IMMEDIATE RELEASE**
**DATED MATERIAL**

Editorial Contact: Sara Payne/Deb Fry
(515) 223-8000
Microware Systems Corporation

Susan Radd/Howard Riefs
(312) 240-2749
Edelman Public Relations
(for Microware)

### Microware's OS-9 Licensed by IBM for OEM Technology

DES MOINES, Iowa, November 18, 1996 Microware Systems Corporation (Nasdaq: MWAR) announced today that IBM has licensed the OS-9 real-time operating system from Microware for potential use in technology aimed at original equipment manufacturers (OEMs) for production of a variety of wireless and wired Internet- and Intranet-access devices.

IBM will use Microware's OS-9 operating system in technology demonstrations this week in the IBM booth (1814L) and the PowerPC Pavilion at COMDEX Fall 1996 in Las Vegas.

"Being chosen as one of the potential operating systems in IBM's OEM network computer technology demonstrates that Microware has a proven and reliable operating system in OS-9", said Ken Kaplan, president and CEO of Microware. Terms were not disclosed.

### OS-9 System Software

OS-9 is a full-featured, multi-tasking real-time operating system. Its architecture is scalable and modular, allowing it to fit a wide range of designs from small handheld devices to large networked systems.

Microware's software is being targeted to emerging, high-volume consumer electronics markets. The company has developed software strategies for the Internet and wireless communications by incorporating Java support, web browser technology, networking, graphics, power management, user interface and communications capabilities.

### Microware

Microware (Nasdaq: MWAR) develops, markets and supports sophisticated real-time operating system software and development tools for advanced consumer, communications, and industrial products, including devices for the emerging digital television and wireless communication markets.

Microware is the first real-time embedded operating system provider to receive ISO 9001 certification for its software development quality assurance methodologies. Microware was founded in 1977, is headquartered in Des Moines, Iowa, and has offices in San Jose, California; Tokyo, Japan; Burnham, England; Aix-en-Provence, France and Munich, Germany; as well as other domestic and international sales offices and authorized distributors.

Microware, OS-9 and DAVID are registered trademarks of Microware Systems Corporation. All other marks are trademarks or registered trademarks of their respective holders.

Note: The statements in this release concerning the outlook for the referenced project are forward-looking statements that involve risks and uncertainties, including successful development of and customer acceptance of the products and services discussed.

***and another note of interest...***

On October 17th, Microware and Corel announced that Corel has chosen OS-9 for their Java products. Microware is getting Corel Office and some other Corel software for porting to OS-9. Corel purchased Word Perfect Corporation some time before this. Corel Office is basically the same as Perfect Office, and includes the popular Word Perfect word processing software as well as spreadsheet and presentation software.

### What does all this mean to OS-9 users?

Well, for the casual or "desktop" user, it probably means nothing. OS-9 is basically being used by IBM as it has been for years now... in embedded products. In this application OS-9 has always excelled due to its small size, hardware flexibility, and overall versatility.

For the industrial user and developer, their will definitely be a big difference. First, this will mean that there will be a bigger demand for OS-9 programming. IBM and Corel will probably hire some of their own programmers, but much of the work may be contracted out. With two big companies needing programming, there should be a steady demand for some time to come.

With the Corel Office Suite ported to OS-9, developers will finally have the software needed to support their work on the same system software is developed on. Word Perfect has been on the leading edge of word processing technology for some time now. This tool will be terrific for creating documentation for developing software. Anyone who knows much about PCs at all knows that Corel Draw has been the leading graphics creation software for a number of years now. The two combined (Word Perfect and Corel Draw... or a derivative thereof) should make it much easier for small developers to create their own documentation. They could even run the program up to a certain point in one window then switch to another to document the events, all on the same machine! If nothing else, it should save money on hardware.

When will all this transpire? It is to early to tell. Unlike Microsoft, Microware isn't in the habit of announcing a date and then pushing it forward. When a product is ready, Microware will send a press release out, and I'll forward that to you. Just remember, the Corel Office suite is a big, complicated package and porting it will involve some programming challenges. The IBM products are still in the planning stages. When they are released OS-9 will be virtually (if not totally) transparent to the user and may not be mentioned at all.

**OS/9> ??**

From: *Dennis Bathory-Kitsz*

**Hi folks! I've been hiding out in Vermont, but since it's the 10th anniversary of my company Green Mountain Micro's demise, I thought it might be time to put in an appearance here.**

**About 150 copies of 'Learning the 6809' (book only) remain, which I'd be happy to offer at $10 postpaid to anyone interested. If at least 10 people also want the original tapes, I'd be pleased to make up a set of those as well.**

**One of these days I'll tell my own tale ... amusing indeed...**

Dennis Bathory-Kitsz
RD 2 Box 2770
Cox Brook Road
Northfield, Vermont 05663

<bathory@maltedmedia.com>
Malted/Media:
http://www.maltedmedia.com/

# The AT306 Computer... Assembly!

## What you get in a box, and how to put it together.

Frank Swygert

As most readers know, FARNA Systems started advertising AT306 based computers in the May/June 96 issue. It normally takes several months for an advertisement to have much effect, and this was no exception. In November of 1996 I received a phone call asking if I sold motherboards and video cards as well as complete systems. He was quoted a price, and his order arrived shortly thereafter. About a week later another order arrived, this time for a complete system.

A few calls determined a good source for the Trident 8900D video cards and other components. Another call to Carl Kreider confirmed that a couple AT306 motherboards would be ready to ship in a week or two. With all parts on order, I awaited their arrival.

In about two weeks, everything arrived via UPS on my doorstep. The first thing I did was inspect the packages. All components for the complete system were in order. The next step was to inspect the package from Kreider Electronics. Contained in each of the two packages was:

AT306 motherboard
AT306 manual (37 pages)
Microware OS-9 manual (Using OS-9)
MGR Font Samples (110 pages)
Software

The first thing I did after checking package contents was to place a video card in one of the motherboard packages and get it off to New York. Then it was time to begin assembly of the full system.

First, I'd like to give some details on the AT306 package as received from the manufacturer. The 37 page manual seems small, but it is very complete. I could not have done a better writing job myself! It starts with a single page titled "Welcome to the AT306". This page contains miscellaneous bits of data that didn't fit in other sections or needed to be brought to the assembler's immediate attention. From there, the manual follows a logical order:

1. Introduction - explaining some design considerations of the AT306. Also points out some details needed for assembly, don't skip!
2. Getting the AT306 Running - installation in case and cables
3. A Look Inside the AT306 - memory map, details of peripherals, interrupts, and switch settings
4. The Keyboard/Video Interface - explanation of keyboard defaults and video modes
5. Serial Ports - baud rates and settings
6. Floppy Disks - descriptor definitions, support for CoCo, Atari ST, PC, MM/1, Mizar, and Microware standard disks, can boot from any
7. Hard Disks - IDE and SCSI supported, must use 512 byte sectors, which limits partitions to 256M max (larger clusters can be used to support larger drives, but performance suffers)
8. Software - description of AT306 specific utilities
9. Support - FTP site for latest software revisions and mailing list address
10. Appendices - schematics, connector pinouts, and other information

One must first read the manual! The first page ("Welcome to the AT306") and the Introduction are both important. Read these before mounting in the case continues. Then follow the directions as laid out in the manual.

The directions assume some familiarity with installing PC motherboards. This may be a slight hindrance if this is the first time one has installed a motherboard in a PC case. A thorough examination of the case and mounting hardware should be all that is necessary for the first timer. Do note that if a tower or minitower case is used, the panel that the motherboard mounts on is removable. Remove this panel, mount the motherboard then proceed, reinstalling the panel just before installing any cards.

Test fit the motherboard in the case (or on the tower panel). Use as many screws and standoffs as possible. I was able to use only two screws, one at the end of the last expansion slot and one in the center along the front edge of the board.

After mounting the motherboard, memory was installed. There are four SIMM sockets on the board, allowing from 512K to 64M of RAM. One must note, however, that all four sockets must contain the same size SIMM. This is due to a limitation of the onboard 68306 memory controller. Using a separate, more flexible memory control circuit on the motherboard would have raised the total price of the board. The sockets are arranged in banks of two. Only the first bank need be filled. Memory arrangement can therefore be as follows:

| | |
|---|---|
| 512K (2x256K) | 1M (4x256K) |
| 2M (2x1M) | 4M (4x1M) |
| 8M (2x4M) | 16M (4x4M) |
| 32M (2x16M) | 64M (4x16M) |

Common 30 pin SIMMs are used. Parity SIMMs are not required, but can be used. The AT306 simply ignores the parity bits. Note that IBM once had 2M SIMMs made, but these have 72 pins.

After installing the memory, the DIP switch should be set. There are eight positions on a single DIP switch array. From the manufacturer, they are set to allow booting from the floppy drive with a terminal connected to /term (port /t0 or /t1). The /term ports are part of the 68306. The baud rate is set by switches 2 and 3 to 2400, 9600, 19200, or 38400. The /term ports only have RTS and CTS signals. They can handshake but cannot detect a carrier loss. The other tow ports (/t2 and /t2) are part of the I/O controller and have the full complement of modem signals. The default setting for /term is 9600 baud. Switch 4 can be set to boot from an installed video card and keyboard instead of a terminal.

The floppy and hard drives were installed next. Cables were connected to the drives next, then to the appropriate motherboard connectors. The power connectors are usually marked PS8 and PS9 (or P8 and P9). These go to the like marked motherboard connectors. If the connectors from the power supply are not marked, remember that the black wires are always toward the center with both connectors edge to edge as connected to the motherboard. If using a tower case, connect the cables to the board after positioning the panel. Then swing the panel in place and screw it in.

With everything installed, the computer is ready to be booted and tested. Kreider does a great job by testing each board with a terminal before it is shipped out, so there should be no problems if everything is connected properly.

A null modem cable will have to be constructed if one wishes to use a terminal or another computer for display and keyboard input. Details of the onboard connectors are given in the AT306 manual appendix. Any terminal emulation software can be used on a CoCo or PC for booting.

Once booted from a terminal, follow the directions in the OS-9 manual to create a new boot disk with all desired modules. If a complete system was purchased from FARNA, it should boot from the hard drive as soon as it is powered on. A floppy boot disk is also supplied. A FARNA motherboard kit comes with a Trident 8900D 1M video card and a boot disk to utilize the card and keyboard.

At the time this was written, I was still waiting on a boot disk from Kreider Electronics. They had a minor problem with the "D" revision of the Trident chipset. Basically, some of the registers were changed by the manufacturer between the B, C, and D revisions. The B and C revision cards use the same driver, but the newest D revision required some adjustment. Carl assures me the disk will be here soon and the adjustments minor. In fact, I expect to have the disk well before the next issue, perhaps before this is read!

In conclusion, the board seems to be very well put together and tested. The manual is very clear and concise. The only thing missing was which SCSI cards are supported (Future Domain 1680 & Adaptec AAH15xx). I found the MGR fonts to be a waste of paper, but others may think otherwise.

# FARNA Systems

Your most complete source for Color Computer and OS-9 information!

Post Office Box 321
Warner Robins, GA 31099
Phone: 912-328-7859
E-mail: dsrtfox@delphi.com

**ADD $3 S&H, $4 CANADA, $10 OVERSEAS**

## BOOKS:

**Mastering OS-9 - $30.00**
Completely steps one through learning all aspects of OS-9 on the Color Computer. Easy to follow instructions and tutorials. With a disk full of added utilities and software!

**Tandy's Little Wonder - $25.00**
History, tech info, hacks, schematics, repairs... almost EVERYTHING available for the Color Computer! A MUST HAVE for ALL CoCo afficiandos, both new and old!!!

**Quick Reference Guides**
Handy little books contain the most referenced info in easy to find format. Size makes them unobtrusive on your desk. Command syntax, error codes, system calls, etc.
CoCo OS-9 Level II : $5.00
OS-9/68000 : $7.00

**Complete Disto Schematic set: $15**
Complete set of all Disto product schematics. Great to have... needed for repairs!

**"A Full Turn of the Screw": $20**
Lots of CoCo info, projects, and tutorials.. by Tony DiStefano

**"Inside Disto's 2 Meg Kit" : $10**
Schematics and explanation of how the 2 meg CoCo 3 upgrade works.

## SOFTWARE:

CoCo Family Recorder: Best genealogy record keeper EVER for the CoCo! Requires CoCo3, two drives (40 track for OS-9) and 80 cols.
DECB: $15.00 OS-9: $20.00

**DigiTech Pro: $10.00**
Add sounds to your BASIC and M/L programs! Very easy to use. Requires user to make a simple cable for sound input through a joystick port. Requires CoCo3, DECB, 512K.

ADOS: Most respected enhancement for DECB! Double sided drives, 40/80 tracks, fast formats, many extra and enhanced commands!
Original (CoCo 1/2/3) : $10.00
ADOS 3 (CoCo 3 only) : $20.00
Extended ADOS 3 (CoCo 3 only, requires ADOS 3, support for 512K-2MB, RAM drives, 40/80 track drives mixed) : $30.00
ADOS 3/EADOS 3 Combo: $40.00

**Pixel Blaster - $12.00**
High speed graphics tools for CoCo 3 OS-9 Level II. Easily speed up performance of your graphics programs! Designed especially for game programmmers!

**Patch OS-9 - $7.00**
Latest versions of all popular utils and new commands with complete documentation. Auto-installer requires 2 40T DS drives (one may be larger).

## NEW ITEMS!!!

FARNA Systems is pleased to announce that we are now distributors of the following, formerly from Northern Exposure!
Note: If you never received your order from NX, send a copy of yourcancelled check along with $5 to cover S&H and I'll fill the order!

**NitrOS-9 : $35.00**
A complete rewrite of OS-9 Level II that takes advantage of all features of Hitachi's 6309 processor. Easy install script! 6309 required.

**TuneUp : $20.00**
If you don't have a 6309, you can still take advantage of some of the Nitro software technology! Many OS-9 Level II modules rewritten for improved speed with the stock 6809!

**Thexder OS-9**
**Shanghai OS-9 : $25.00 each**
Transfers your ROM Pack game code to an OS-9 disk! Please send manual or ROM Pack to verify ownership of original.

**Rusty : $20.00**
Launch DECB programs from OS-9! Allows loading of some programs from hard drive!

## FARNA Systems AT306 Based Computers

Complete computer systems based on the AT306 board from Kreider Electronics. Systems are completely setup and ready to go. Just add a VGA monitor (or we can supply that too)!

**Both systems include:**
16 bit PC/AT I/O bus with five slots
MC68306 CPU at 16.67MHz
4 30 pin SIMM sockets
IDE Hard Drive Interface
1.4MB Floppy Drive
Two 16 byte fast serial ports (up to 115K baud)
Bi-directional parallel printer port
Real-time clock
PC/AT keyboard
Desktop Case and Power Supply
(mini-tower case optional, no cost!)
BASIC (resembles Microsoft BASIC)
MGR Graphical Windowing Environment
with full documentation
"Personal" OS-9/68000 Vr 3.0
(Industrial with RBF)
Drivers for Tseng W32i
and Trident 8900 VGA cards
Drivers for Future Domain 1680
and Adaptec AAH15xx SCSI cards

***Many other utilities and tools***

**FARNA-11123 Includes:**
**2MB RAM**
**300MB Hard Drive (was 200!)***
**Trident 8900 1MB Video Card**
**$960.75**

**FARNA-11225 Includes:**
**2MB RAM**
**500MB Hard Drive***
**Tseng W32i 1MB Video Card**
**$1114.47**

***This is the SMALLEST amount of formatted space available.***
***Prices fluctuate - we get you the largest drive possible for the money allotted!***

**HACKERS MINI KIT (FARNA-11100): Includes AT306 board, OS-9 and drivers, util software, assembly instructions/tips, T8900 1MB video card. Add your own case, keyboard, drives, and monitor! *ONLY $500!***

**Call for a quote on different configurations and components.**
**Warranty is 90 days for labor & setup, components limited to manufacturers warranty.**

**Microware Programmers Package -**
**Licensed copies of Microware C compiler, Assembler, Debugger, Basic, and many other tools!**
***With system purchase:* $65.00 *Without system:* $85.00**

# Detecting 680X0 Processors

## Which model processor is in the box?

Simon N. Goodwin

How does one tell what type of 68K processor is in a particular machine? I sorted this out on the Amiga a couple of years ago, and the code seems to work on lots of 68040/68060 systems including Commodore, GVP, Cyberstorm, Warp Engine etc.

I don't know of any way to check for these CPUs without fiddling with the exception vectors. There are easier ways to check for 68020 versus 68000, such as setting a scale factor and seeing if it's ignored, or this trick:

```
* Test routine to determine CPU type - distinguishes 68020+ from earlier models; S N Goodwin. See 68XXX Programmer's Reference Manual, page 4-128
*
 SiftCPU movem.l a7,-(a7)
* Don't let this line get 'optimised'
      cmpa.l  (a7)+,a7
*Was stored value pre-decremented?
      bne.s   MC020
*Z unless this is a 68020 or later
*
* Warn that this is not a 32 bit capable processor
*
      moveq   #10,d0
      rts
*
* MOVEM gives the same result as this on a 68000/68008/68010/68012
*
*       move.l  a7,-(a7)
*       cmpa.l  (a7)+,a7
*
* But 32 bit CPUs (68020/68030/68040/68060/CPU32) MOVEM differently
*
MC020   moveq   #0,d0
      rts

      end
```

The following code does not look explicitly for a memory management unit, but it does not seem necessary to set up a full translation table to determine this on a 68040. I've not got a 68060EC or LC to try this routine on, but it sifts 68040XC and 68040EC perfectly. Unfortunately the 68060 PCR does not distinguish EC and LC chips, probably because Motorola wants to ship LCs with EC labels if the demand justifies it...

Contrary to popular belief, it IS possible to tell whether or not a 68040 has an on-chip MMU with a simple test. This code has been extensively used on QXL cards which work with both XC and EC 040s and checked on an Amiga 4000 with XC and EC processors in the CBM 3640 processor card:

```
      pflusha                 Opcode $F518
```

This yields a Line F exception if the 68EC040 is fitted yet works fine on 68LC040 and above. Note that the encoding $F518 *must* be used, which requires a 68040/68060-aware assembler. 68020 and 030 assemblers use a *different* opcode for PFLUSHA ($F0002400) which is rejected by the 68040 and above. If in doubt, use DC.W $F518 (Source: PROCTYPE_X, part of the PD C68 compiler package for Qdos). Any equivalent for the 68EC030 would be most gratefully received!

Returning to the main part of this article, PROCESSOR_ASM (following), there are debug lines to change the palette as it works; you'll need to alter the palette address and values but might otherwise find these useful.

Recently I've been in contact with Johan Klockars (johan@rand.thn.htu.se) who's written one of the Qdos emulators for ST and is currently porting it from 68030 to 68040, making considerable use of Amiga Qdos source; so he might be able to advise on ST-specific hardware problems associated with the 68040/060, Copyback cache etc.

```
*/beginfile PROCESSOR_asm
; PROCESSOR_asm - QDOS processor specific routines - last modified 26/11/96
; QDOS-Amiga sources by Rainer Kowallik
; latest changes by Mark J Swift
; 68040/68060 changes by SNG, 16/8/94
;*/beginoverlay
;
; This is an extract from the processor dependent routines for Amiga Qdos (a 680XX operating system based on Sinclair's Qdos for QL (Quantum Leap) 68008 machines, now available on Amiga,
ST, PC, Mac and Unix under various hard and soft emulators.
; The code is freely distributable and seems to work fine. It uses a freeware 68000 assembler so 68020+ opcodes are written as word constants, with comments to indicate what's going on. Hope this helps!
; Simon N Goodwin, January 1997
;  check attn flags for 68000/008 processor. Returns Z if so create attention flags (check processor type)
;  bit 0 - at least 68010
;  bit 1 - at least 68020
;  bit 2 - at least 68030
;  bit 3 - at least 68040
;  bit 4 - at least 68881 (possibly 68882)
;  bit 5 - 68882 present (or emulation)
;  bit 6 - 68040 or 68060 on-chip FPU enabled
;  bit 7 - at least 68060
; Note: If the MC68060 FPU has been turned off in software since the last reset (which enables it) bit 6 is zero, as if the FPU was not present and we were using an EC or LC processor.
ATTN_FLGS:
      movem.l d1/a0-a3,-(a7)
      dc.w    $2078,$0010
; move.l  $10.w,a0
      dc.w    $2478,$002C
; move.l  $2C.w,a2
      lea     LF80C18(pc),a1
      dc.w    $21C9,$0010
; move.l  a1,$10.w
   ; (ILLEGAL INSTRUCTION vector)
      dc.w    $21C9,$002C
; move.l  a1,$2C.w
   ; (F-LINE EMULATION vector)
      move.l  a7,a1
      moveq   #$0,d0
; initialise attn flags
      move.b  #0,161(a6)
; ditto for QDOS
      dc.w    $4E7B,$0801
; movec d0,vbr
      bset    #$0,d0
; vbr present - 68010 at least
;               | Clear Data cache
;               || Freeze Data cache
;               ||| Enable Data cache (off)
;         |||| Clear Instruction cache
;         ||||| Enable Instruction cache
;
      move.l  #%0000101000001001,d1
      dc.w    $4E7B,$1002
; movec d1,cacr
      dc.w    $4E7A,$1002
; movec cacr,d1
      bset    #$1,d0
; cacr present - 68020 at least
      move.b  #$20,161(a6)
; store for QDOS
      btst    #$9,d1
      beq.s   LF80BB4
; no data cache - not a 68030
      bset    #$2,d0
; definitely a 68030
      move.b  #$30,161(a6)
; store for QDOS
 LF80BB4:
      btst    #$0,d1
      bne.s   LF80BDC
; cacr present and (EI) bit operative - not a 68040
      or.w    #$C,d0
; 68040 at least!
      move.b  #$40,161(a6)
; store for QDOS
      dc.w    $F4D8
; CINVA ic/dc - inavalidate caches
; This code set up the TTR/ACU registers to stop data cacheing of memory in the first 16 Mb. This is not needed under Amiga Qdos as the cache is explicitly cleared when necessary after DMA.
; SNG  move.l #$C040,d1
; Serialize 0-16 Mb
; moved: dc.w  $4E7B,$1006
; movec d1,(006)
      moveq   #0,d1
; MMU off
      dc.w    $4E7B,$1003
; movec d1,TCR
      move.l  #$00FFC000,d1
; ACU/TTU off
      dc.w    $4E7B,$1007
; movec d1,(007)
      dc.w    $4E7B,$1004
; movec d1,(004)
      dc.w    $4E7B,$1005
; movec d1,(005)
      dc.w    $4E7B,$1006
```

```
; movec d1,(006) - moved by SNG
      move.l #$0000C000,d1
; Write Through to Chip memory
      dc.w   $4E7B,$1006
; movec d1,(006) DTT0
      move.l #$00FFC020,d1
; Copyback on all memory
      dc.w   $4E7B,$1007
; movec d1,(007) DTT1
      move.l #$80008000,d1
; Both caches on, for 040+
      bra.s  SetCACR
 LF80BDC:
;     Enable Instruction cache =040 (1=ON)
;           | Write Allocation (1=ON)
;           || Clear Data cache
;           ||| Enable Data cache (1=ON)
;           |||| Clear Instruction cache
;           ||||| Enable Instruction cache
;         ||||||
      move.l #%1010100100001001,d1
 SetCACR dc.w   $4E7B,$1002
; movec  d1,cacr
      btst   #$3,d0
      beq.s  LF80BF6
; skip if not a 68040 or later
; 68060 initialisation added here
; Note this tests the new PCR register to find out if
we're on an 060. If this fails, the routine returns after
checking for a 68040 FPU. Otherwise it turns on the
060 accelerators and reports the presence of an enabled
FPU.
      lea    NoPCR(pc),a3
; Continuation if PCR absent
      dc.w   $21CB,$0010
; move.l a3,$10.w
; (ILLEGAL INSTRUCTION vector)
      dc.w   $4E7A,$1808
; 68060 PCR to D1 or illegal
      swap   d1
      cmp.w  #$0431,d1
      beq.s  No60FPU
; No FPU!
;     cmp.w  #$0430,d1
; The only other possibility
;  bne.s  Unknown at the time of writing...
      btst   #17,d1
; Test swapped DFP flag bit
      bne.s  No60FPU
; If FPU is disabled, leave off
      bset   #6,d0
; Note 68060 FPU is available
 No60FPU bset   #7,d0
; Set 060 bit in ATTN flags
      move.b #$60,161(a6)
; Tell Qdos we're on an 060
      swap   d1
      bset   #0,d1
; Ensure SOEP is on
      dc.w   $4E7B,$1808
; Store updated PCR
; Enable store buffer, code, data and (cleared) branch
cache
      move.l #$A0C08000,d1
; EDC+ESB+EBC+CABC+EIC
      dc.w   $4E7B,$1002
; movec  d1,cacr
      bra.s  LF80C18
; The old code to check for the 040 FPU always failed
because the 68040 does not support cpSAVE and
cpRESTORE! Substitute FNOP
;     dc.w   $F327
; cpSAVE
;     dc.w   $F35F
; cpRESTORE
 TryFPU  dc.w   $F280,0
; FNOP sifts out ECs and LCs
      bset   #$6,d0
; 68040 FPU present
      bra.s  LF80C18
 NoPCR  movea.l a1,a7
; Tidy stacked exception frame
      bra.s  TryFPU
; Test for an off-CPU co-processor
 LF80BF6:
      moveq  #$0,d1
      dc.w   $F201,$9000
; FMOVE D1 to FPU
      dc.w   $F201,$B000
; FMOVE FPU to D1
      tst.l  d1
; A most unlikely case
      bne.s  LF80C18
; Something strange has happened
      bset   #$4,d0
; 68881 at least
      dc.w   $F327
; cpSAVE -(A7)
      cmpi.b #$18,$1(a7)
; Check for 68882 frame
      beq.s  LF80C16
      bset   #$5,d0
; 68882 at least
 LF80C16:
      dc.w   $F35F
; cpRESTORE (A7)+
 LF80C18:
      move.l a1,a7
; clean-up stack
      dc.w   $21C8,$0010
; move.l a0,$10.w
      dc.w   $21CA,$002C
; move.l a2,$2C.w
      movem.l (a7)+,d1/a0-a3
      rts
 ; clear data and instruction caches
 CLRALL:
      movem.l d0-d1/d4,-(a7)
      move.l #$808,d1
      bra.s  L0000C3E
; clear data caches
 CLRDATA:
      movem.l d0-d1/d4,-(a7)
      moveq  #$8,d1
      rol.l  #8,d1
; clear cache(s). d1=$800 - clear data cache d1=$808
- clear data and instruction caches and on 68040/060 -
update memory from caches
 L0000C3E:
      trap   #0
      move.w sr,-(a7)
      beq.s  CLRC00
      bclr   #5,(a7)
 CLRC00:
      ori.w  #$0700,sr
; interrupts off
      exg    d1,d4
      bsr    GET_ATTN
      exg    d1,d4
      btst   #$1,d4
; branch if '020 or more
      bne.s  L0000C48
      ifd    ShoCach
      move.l d7,-(a7)
      move.w #$8000,d7
 WAITBLU:
      move.w #$000F,$DFF180
      move.w #0,$DFF180
      dbra   d7,WAITBLU
      move.l (a7)+,d7
      endif
      bra.s  CLRCACHEX
; ...otherwise exit
 L0000C48:
      btst   #$3,d4
      bne.s  L0000C68      ; '040 ?
; No explicit test for the 060 is needed as long as the
040 bit remains set by tests on a 68060.
;     btst   #$7,d4
; Check for '060
;     bne.s  L0000C68

      and.l  #$808,d1
      ori    #$700,sr
      dc.w   $4E7A,$0002    ; movec cacr,d0
      or.l   d1,d0
      dc.w   $4E7B,$0002    ; movec d0,cacr

      ifd    ShoCach
      move.l d7,-(a7)
      move.w #$8000,d7
 WAITRED:
      move.w #$0F00,$DFF180
      move.w #0,$DFF180
      dbra   d7,WAITRED
      move.l (a7)+,d7
      endif
      bra.s  CLRCACHEX
 L0000C68:
      btst   #$3,d1
      bne.s  L0000C74
      dc.w   $F478
; CPUSHA dc ('040 only)
; update memory from cache
      bra.s  L0000C76
 L0000C74:
;     dc.w   $F4F8
; CPUSHA ic/dc ('040 only)
; update memory from caches
      dc.w   $F478
      dc.w   $F498
; CINVA ic ('040 only)
; invalidate instruction cache
 L0000C76:
      ifd    ShoCach
      move.l d7,-(a7)
      move.w #$8000,d7
 WAITGRN:
      move.w #$00F0,$DFF180
      move.w #0,$DFF180
      dbra   d7,WAITGRN
      move.l (a7)+,d7
      endif
 CLRCACHEX:
      move.w (a7)+,sr
      movem.l (a7)+,d0-d1/d4
;     tst.l  d0
      rts
 ;*/endoverlay
 ;*/endfile
```

Simon N Goodwin
simon@studio.woden.com
AKA simon@silicon.studio.co.uk

# Operating System Nine

Alan Dekok

## Make a new OS-9 boot disk in just a few minutes!

This article is number three of a series of articles on the inner workings of OS-9 Level II and NitrOS-9. It is also the final installment of the series. Next issue... *Rick's back!!!*

**Creating OS-9 Boot Disks**

Do you still dread the thought of creating a new boot disk every time someone comes out with a new patch for OS-9? Fear no more.

First, we will show you a technique for creating a new boot disk painlessly in just a few minutes. There is a little setup involved, but the end result makes it worthwhile. In the second part of this article, we will cover what modules you NEED to have in a boot file.

To simplify creating boot disks, set up a disk with the following basic structure:

```
CMDS/  Any commands you need to
    create bootdisks.
BOOTMODS/  Boot modules.
BOOTMODS/LISTS  Lists of modules,
    and scripts to generate a boot disk.
SYS/  System files, such as fonts,
    patterns etc...
```

Then copy any boot modules you commonly use to that disk, placing them in the BOOTMODS directory. Now copy any needed commands to the CMDS directory. Leave the LISTS directory empty for now. One of the NitrOS-9 bootfile disks, for example, is set up as follows. The CMDS directory shows the minimum set of commands you will need:

```
cc3go   * CC3Go file
CMDS/    shell
    * shell with some merged commands
grfdrv    * graphics driver
format    * disk formatting command
rename    * used by os9gen
os9gen    * creates a bootable disk
BOOTMODS/os9p2.122n
    * second part of kernel
os9p3.122b * prints long error messages
init.dd   * init table
rbf.120   * Random Block File manager
cc3disk.115 * Tandy floppy disk driver
dd.d0.360 * Default Device (same as d0)
d0.d0.360    * 360k floppy drive
d1.d1.720    * 720k floppy drive
rammer.120  * KD RamDisk driver
md          * memory device descriptor
r0           * ramdisk descriptor
ioman.121   * Input/Output manager
cc3io.joy.120  * CC3IO for joysticks
scf.120
   * Sequential Character File manager
sacia.nompi.115
   * RS232 pak driver for Y-cable
sacia.mpi.115
   * RS232 pak driver for MPI
t2           * serial port
printer      * bit-banger printer driver
p            * printer descriptor
pipeman.122j  * pipe manager
piper        * pipe driver
pipe         * pipe descriptor
windint.122c    * window driver
term         * window descriptor
w       * default window descriptor
w1-w15* additional window descriptors
clock.soft.60.122n    * software clock

BOOTMODS/LISTS/
SYS/  stdfonts    * standard fonts
      stdpats     * standard patterns
      errmsg      * error messages
```

That's most of the work done — getting everything you need in one place.

One key to creating bootdisks easily and quickly is to avoid doing as much of the work as possible yourself. To do that, let your computer do all the work. Just follow these three steps, A, B, and C:

A. Create a file called 'BOOTMODS/LISTS/boot.tandy.d0' using your favourite text editor, such as VED from Bob van der Poel. Put in the following entries, substituting the names you use for your modules:

```
../os9p2.122m
../init.dd
../ioman.121
../rbf.122
../cc3disk.121
../dd.d0.360
../d0.d0.360
../d1.d1.720
../scf.120
../printer.116
../p
../cc3io.joy.120
../windint.122c
../term
../w
../w1
../w2
../w3
../w4
../clock.soft.60.122n
```

This is the list of boot modules that you will later pass to os9gen to create the OS9Boot file on a new disk. The '../' indicates that the files are located in the directory above the LISTS directory.

If you have a lot of modules, you can even create sub-directories, such as RBF/ for disk drivers and descriptors, SCF/ for serial and printer modules, and WIN/ for all your window descriptors. Special hardware modules such as Burke and Burke hard drives can also go in their own directories.

Now create a second text file called 'BOOTMODS/LISTS/make.tandy' on the boot modules disk, such as the following which creates a new boot disk in driver /D0. When running the script, ensure that the disk in /D0 doesn't contain information you want to save!

```
echo Formatting /d0...
* Format the disk in /d0
format /d0 r "OS-9 Bootdisk"
echo
echo Creating OS9Boot file...
* Use OS9Gen to create
os9gen /d0 <boot.tandy.d0
* the kernel track and
echo
* the OS9Boot file
echo Creating CMDS directory...
* Create a CMDS directory
makdir /d0/CMDS
echo Copying command files...
* And copy any needed
copy .../cmds/grfdrv.122m /d0/cmds/grfdrv
* files
copy .../cmds/shell /d0/cmds/shell
echo Creating SYS directory...
* Create the SYS directory
makdir /d0/SYS
echo Copying CC3Go file...
* Another needed file
copy .../CC3Go /d0/CC3Go
echo Done.
```

That's the last of the setup work. The short-term payoff comes next.

B. To see all your hard work in action, put a blank disk in drive 0 and the boot modules disk in drive 1, and type in the following:

```
chd /d1/bootmods/lists
chx /d1/cmds
make.tandy
```

At this point, your work is over, and the make.tandy script will begin to run. First, the disk in drive 0 will be formatted. Next, OS9Gen will copy REL, BOOT, and OS9P1 from memory to track 34 of /d0. OS9Gen will then read the boot.tandy.d0 file and put those modules onto /d0 into a contiguous file called TEMPBOOT. Once OS9Gen has done this, it will put the TEMPBOOT file's location on LSN0, and use RENAME to rename TEMPBOOT to OS9Boot.

At this point, the disk is ready to go, except for the commands, and the CC3Go file. The rest of the script takes care of this for you.

After the script has finished running, you now have a bootable disk in /d0! Simply press reset, and the computer will boot from your new bootdisk. Press reset twice, and type in

DOS, and once again the computer will boot. Once things were set up, you have to admit this second step was pretty easy.

C. Now comes the fun part, and you should be able to handle the details yourself. Simply by adding new modules or commands to the disk, and modifying the boot.tandy.d0 or make.tandy files, you can easily create your own custom boot disks.

**Required Modules in an OS-9 Boot Disk**

Having REAL problems creating an OS-9 boot disk? Here's a minimal OS-9 Boot disk set up. If you make a boot disk using JUST what's listed here, you're pretty much guaranteed that either it will boot, or you will easily be able to track down the problem. From the point that you are able to create your first working boot disk, it is a simple matter to create the second!

Kernel track: REL, BOOT, OS9p1.

The hardest things here are picking the correct BOOT module, and getting the kernel track written to the disk, but fortunately OS9Gen will take care of that for you on a simple system.

OS9Boot file:
OS9P2 - second half of the kernel
Init - use the stock Tandy one
RBF
CC3Disk - or whatever driver is used by DD
DD
VDGInt - or WindInt, but then you have to set up GrfDrv
Term
SCF
CC3IO - for the keyboard

Surprisingly, that's all you need! All of the other modules in the OS9Boot file just make your life easier. Add them in if you want, too.

Remember however you get the OS9Boot file, you MUST have LSN0 updated! Use OS9Gen, ezgen, or whatever you're used to.

in /DD:
CC3Go - use the stock Tandy one
CMDS - commands directory

Note: No 'startup' file: it's not necessary.

in /DD/CMDS
Shell - remember: attr Shell e pe
GrfDrv - required if Term is a 80 or 40-column (non-VDG) window - remember: attr GrfDrv e pe

Whatever commands you're interested in using. But DON'T FORGET SHELL and GRFDRV !!!!

At this point, once you have created a boot disk, add one set (eg. /t2, aciapak, SCF) of modules at a time until you have the boot disk you've always wanted. Remember though — OS9Gen does do some checking. If, for example, you try to add in a descriptor that refers to a driver that isn't present, OS9Gen won't allow it to be included in the OS9Boot file. Same thing for a driver — if its manager isn't present, it won't be loaded.

Editor: A Note on Patch OS-9

Between CoNect and FARNA Systems, I believe about 200 copies of Patch OS-9 were sold. If you have a copy, the installer routine does just what Alan has described. A "bootmaker" disk is created from your original Tandy disk. The bootmaker contains all the original Tandy drivers, new drivers, and patched modules that come with Patch OS-9. All one has to do is edit one of the included bootlists and run OS9Gen to create a new boot disk!

If you don't have Patch OS-9 and would like to order, it is still available from FARNA Systems. Do note that the installation program requires the use of the original Tandy distribution disks. You cannot use a previously patched boot disk to create a "bootmaker" disk! The reason for this is that it would be nearly impossible for the install program to sort through and detect all the possible patches. What it does is check for original modules and copies them to a Tandy directory, then installs patches and copies the newly patched modules to another directory. All the latest, most compatible patches are included with Patch OS-9. If there are other, special patches you need, install them on the bootmaker disk AFTER Patch OS-9 has been run using your original disk (or rather BACKUPS of your originals!).

**Differences Between Patch OS-9, Nitro, Tuneup, and Powerboost**

***PatchOS-9*** is a collection of the most common, popular, and compatible patches freely available. All can be found in the various archives scattered over FIDO, public BBSes, and the Internet. These have been thoroughly tested and will all work with each other in the same system. There is also an installation program that does most of the work for you.

***Nitro*** (NitrOS-9) is a complete rewrite of most OS-9 modules to take full advantage of the Hitachi 6309 CPU. It is not compatible with most available patches. 6809 (standard OS-9) add-on modules and commands should work with no problems, though some may not. Most of the features of all the popular patches has been incorporated into Nitro, making the patch problem a moot point. System speedup is between 25-50%.

***Tuneup*** was written by the authors of Nitro. It incorporates many of the discoveries implemented in Nitro for 6809 CPU owners. Some OS-9 modules are completely rewritten to provide a noticeable speed improvement. Overall system speed may not be affected, but some operations gain up to 10% in speed.

***PowerBoost*** was written by Chris Burke. Just the OS-9 kernal was rewritten to take advantage of the 6309. Since external modules and commands are the same as those on a 6809 system, most patches can be applied with no compatibility problems. If there is a special patch that you have devised or use, it may not work with Nitro, but will with PowerBoost. Overall system speedup will only be about 10%.

Note that PowerBoost is not currently available, though a used copy may turn up. All the other products are currently available from FARNA Systems.

**Recommendations?**

If compatibility along with added functionality is your primary concern, stick with old faithful – Patch OS-9. For the most speed, nothing beats Nitro! Tuneup and Powerboost are both in the middle. If you are using a 6309, you may want to try finding a copy of Powerboost for the best compromise of compatibility and speed. If you want more speed but don't want to install a socket [illegible] 6309, Tuneup is the only way to go.

# MORE COCO SPEED!

John Kowalski (Sockmaster)

## Double the speed of your 6809 or 6309 to 4MHz!!!!

***The 4Mhz CoCo3 Project***

This project is actually a pretty old idea. The idea of pushing more speed out of the CoCo3 is almost as old as the CoCo3 itself. Soon after the introduction of the CoCo3, it was many people's impression that it was nice and all, but it just wasn't enough. Sure you could do 2Mhz, but so could the CoCo2 - to some extent. With the increasing speed of *other* computers, the CoCo3 was starting to lag behind in CPU power.

To clarify a few things, this project isn't really 4Mhz. Nor is a stock CoCo3 really 2Mhz. This little bit of mislabelling actually started at the very beginning with the CoCo1 - which was actually 0.89Mhz. Since it was close enough to 1Mhz, most people just called it as such because it was easier to say. When the 'fast' pokes were discovered, people simply carried the tradition bysaying it ran at 2Mhz because it was twice as fast.

In actuality, the CoCo3 is really 1.789Mhz, and a '4Mhz' CoCo3 would be twice that - 3.579Mhz. But there are other reasons why I convince myself that I can get away with calling it 4Mhz. I'll get into those later.

***How to speed up your CoCo3***

There are quite a number of ways you can increase the clock rate of your CoCo3. We'll start with the simplest methods and work our way up to the more complex hardware modifications.

The easiest modification you can do is to simply replace the main clock crystal on the motherboard with another frequency, as explained in the last issue. The CoCo3 has a single clock crystal on the board, 28.636363Mhz. This crystal gets divided and provides the timing for every single piece of hardware on the machine. If you change it, EVERYTHING changes speed, including the rate of the video display. Because of this, the usefullness of accelerating your CoCo this way is a bit limited. See the last issue for details.

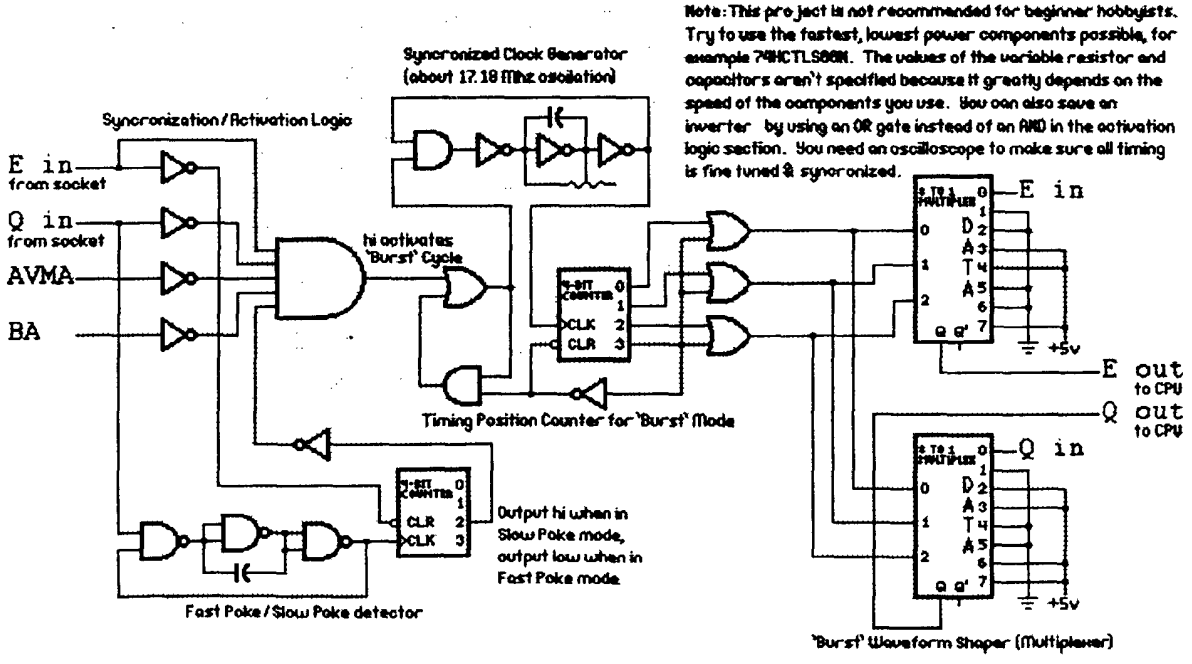
All that aside, there are more problems, especially if you're planning on going beyond 2Mhz. If you have an RS-232 Pak, it's operation starts failing around this point, and there is no solution that I can see unless there is a faster version of the 6551 chip that you can buy and plug into your RS-232 Pak. Printer and Cassette port operation also changes rate accordingly. The Cassette routines will no longer be able to read your old tapes, and there is no easy solution - but then, it probably doesn't matter. Not too many people use the cassette port. The Printer port is the same, but it's easy to poke new values into the baud rate selector, so this turns out to be a very minor thing.

The next easiest thing you can do is simply toss out the 6809 and replace it with a Hitachi 6309. While this doesn't actually increase the clock speed of the CPU, it's a common way of increasing performance, so I've included it here.

Without doing anything else, the CPU performs at exactly the same speed as a 6809, but once you kick it into 6309 Native mode, things start looking up. All old programs automatically get an average speed increase of about 12% (everyone gets a different number, but this is my measurement). When you start using programs that were specifically written for the 6309, things get drastically better. Some operations can be many times faster, but on average, I'd say it's somewhere near 40% faster.

The performance increases of the 6309 are due to a number of things, but the two important ones are that the CPU executes most instructions in fewer clock cycles than a 6809. The other big reason is that is has a bunch of new registers, functions, and instructions that the 6809 does not. You will need either Nitro, PowerBoost, or Chris Dekker's patches to get a speed increase from OS-9, Robert Gault's patches for DECB.

## Sock Master's 4Mhz CoCo3 Project (Logic Diagram)



One good thing about the 6309 is that, even if you do get to try one of the other methods written below, if you combine the acceleration hardware along with a 6309, you can increase your performance even more! That, by the way, is my excuse for getting away with calling my 4Mhzaccelerator project '4Mhz'. Even though my accelerator is really 3.579Mhz, the extra performance increase of using it along with a 6309 make it not so much of a lie to call it 4Mhz.

For real speed increases, what you need to do is run the CPU at a different rate while still running the rest of the CoCo hardware at it's original speed. There are some bottlenecks to this concept. The biggest enemy is actually the infamous GIME chip itself. The GIME controls all access to memory, and since it's a big chunk of circuitry all built into a single package, it becomes impossible to tinker with it. The conclusion quickly becomes that even though the CPU can run at whatever rate you want it to, all memory access STILL has to happen at the old rate of 1.789Mhz.

This next idea is that the clock rate of the CPU be generated by a seperate circuit that is independent of the main clock circuit. While I was thinking about the 4Mhz Project, I had a friend of mine also thinking about it. Both of us came to completely different concepts.

This is his idea: The new clock circuit for the CPU can run at whatever rate you want. He had ideas of making it selectable to any speed, going as fast as 8Mhz. The circuit attempts to keep the CPU running at this rate as much as possible, but every time the CPU needs to access memory, it halts the clock generator and waits for syncronization with the old 1.789Mhz bus. Once syncronized with a bus cycle, the new clock generator proceeds to execute a single 1.789Mhz cycle. Thus memory is accessed, the rest of the CoCo is happy, and then the CPU tries to resume it's idea of running at 8 (or whatever) MHz.

There turns out to be a hitch. In order for true 8Mhz operation to happen, the CPU has to never access memory! In most operations, the CPU accesses memory most of the time. True, there are plenty of 'non memory' cycles, but even single cycles that don't use memory don't help. For any single cycle that doesn't use memory, the CPU DOES execute at 8Mhz, but for the next cycle that does use memory, the circuitry has to halt the clock until it reaches a point ofsyncronization with the bus. The reality of this is that it has just spent 7/8ths of a bus clock cycle waiting for that syncronization point - hence no speed increase whatsoever. Ah, but what about cases where there are two or more CPU cycles in a row that don't use memory? In those cases, the CPU runs happily along at 8Mhz, until the next cycle that needs memory comes along.

There are a number of cases where there are two or more 'non-memory' cycles in a row,most notably, the MUL instruction which only reads memory twice, and then runs the next 9 cycles internally. With this circuit, a MUL will execute in 4, possibly 3 (if you clock the CPU at just over 8Mhz) bus cycles, instead of the stock 11 cycles. That's a speed increase of 366%! This is a best case scenario though. On average, this circuit would only increase performance by a fairly small percentage.

Surely, there must be some other way, you say? And yes, there are plenty of other tricks you can use. That was my friend's idea, here is what my idea at the time was (we came up with these ideas independantly without knowledge of the drawbacks of each other's idea). My figuring was that the best way to boost performance would be to try to sneak extra CPU cycles by the GIME without the GIME ever noticing what had happened. This concept doesn't actually run the CPU at a seperate clock speed. The plan was that some extra circuitry could analyse the signals coming out of the CPU and then decide when it could get away with a sneaky 'burst' cycle without the rest of the motherboard noticing. The circuit is designed to run the CPU at the same old speed it usually ran at (1.789Mhz), except that the circuit knows when the CPU will not be using memory (note the future tense) in the next CPU cycle. Every time this happens, the new clock circuit takes over.

This circuit waits for a point near the end of the 1.789Mhz bus cycle, then bypasses the normal clock signals. It starts feeding this new signal to the CPU instead. The new signal is effectively pasted in the place of the old one. The general gist of it is that the last bit of the normal clock pulse gets cut short, after a small amount of time a very fast extra clock pulse is generated (somewhere's around 6 or 7Mhz) and completed, then it leaves a bit of blank time for the CPU to settle and then restores the normal clock operation after shaving a little bit of time off the beginning of the next 1.789MHz bus cycle. The effect? The CPU just executed two operations in one bus cycle and the GIME never saw it.The advantage of this method is that it works even when there is only one non-memory CPU cycle between two cycles that do use memory. This circuit can make the CPU run at 4Mhz even when it has to access memory half of the time. In reality, the CPU uses memory more often than it does not, so the circuit doesn't quite reach 4Mhz performance levels. But it is noticably more effective than the first method. After constructing this circuit, I realized I had other obstacles...

We are now in the realm of the theoretical. The next ideas have never been built but offer even better performance. The problem with the previous method is that the 6809 was designed in a way to improve performance. Yes, that is the drawback of the previous method. Motorola's design was that the CPU read memory while it's decoding the opcode, in the expectation that this value that was just read will be used in the future. It works in many cases, but for certain operations, the value that was just read from memory gets discarded because it was not needed. To make matters worse, when it completes executing that opcode, the CPU then decides to read the SAME memory location again to get the next opcode to execute. The CPU has just read one memory location twice in a row!

I was expecting better performance in my accelerator project, and now I figured out why it wasn't running as fast as it should have. Each of those 'twice in a row' memory reads take two CPU cycles, one effectively becoming a wasted bus cycle to my accelerator. The solution? A one-byte cache for the 4Mhz Project. Every memory location that gets read by the CPU gets stored in this cache. Every time the CPU tries to read the same memory location twice in a row, a new part of the circuit would decide that this value is already stored in the cache, and enable the 'burst' cycle to the CPU while putting this value on the data bus. This would effectively eliminate all slowdowns caused by the CPU's inefficient usage of the bus.

Even still, the project will not actually run the CPU at true double speed. In cases where the CPU actually needs to access memory many times in a row, the circuit has to shut down and let the CPU access the bus at 1.789Mhz. How could we solve

28456 S.R. 2, New Carlisle, IN 46552
219-654-7080 eves & ends MO, Check, COD; US Funds
HawkSoft Shipping included for US, Canada, & Mexico

## MM/1 Products (OS-9/68000)

**CDF 0.00** - CD-ROM File Manager! Unlock a wealth of files on CD with the MM/1! Read most text and some graphics from MS-DOS type CDs.

VCDP $50.00 - New Virtual CD Player allows you to play audio CDs on your MM/1! Graphical interface emulates a physical CD player. Requires SCSI interface and NEC CD-ROM drive.

KLOCK $20.00 - Optional Cuckoo on the hour and half hour!! Continuosly displays the digital time and date on the /term screen or on all open screens. Requires I/O board, I/O cable, audio cable, and speakers.

WAVES vr 1.5 $30.00 - Now supports 8SVX and WAV files. Allows you to save and play all or any part of a sound file. Merge files or split into pieces. Record, edit, and save files; change playback/record speed. Convert mono to stereo and vice-versa! Record and play requires I/O board, cable, and audio equipment.

MM/1 SOUND CABLE $10.00 - Connects MM/1 sound port to stereo equipment for recording and palyback.

GNOP $5.00 - Award winning version of PONG(tm) exclusively for the MM/1. You'll fo crazytrying to beat the clock and keep that @#$%& ball in line! Professional pongists everywhere swear by (at) it! Requires MM/1, mouse, and lots of patience.

## CoCo Products (DECB)

**HOME CON RO 20.00** - Put your old TRS-80 Color Computer Plug n' Power controller back on the job with your CoCo3! Control up to 256 modules, 99 events! Compatible with X-10 modules.

HI & LO RES JOYSTICK ADAPTER $27.00 - Tandy Hi-Res adapter or no adapter at the flick of a switch! No more plug and unplugging of the joystick!

KEYBOARD CABLE $25.00 - Five foot extender cable for CoCo 2 and 3. Custom lengths available.

MYDOS $15.00 - Customizable, EPROMable DECB enhancement. The commands and options Tandy left out! Supports double sided and 40 track drives, 6ms disk access, set CMP or RGB palettes on power-up, come up in any screen size, Speech and Sound Cartridge support, point and click mouse directory, and MORE OPTIONS than you can shake a stick at! Requires CoCo3 and DECB 2.1.

DOMINATION $18.00 - Multi-Player strategy game. Battle other players armies to take control of the planet. Play on a hi-res map. Become a Planet-Lord today! Requires CoCo3, disk drive, and joystick or mouse.

SMALL GRAFX ETC.

| | | |
|---|---|---|
| "Y" and "TRI" cables. Special 40 pin male/female end connectors, priced EACH CONNECTOR - | | $6.50 |
| Rainbow 40 wire ribbon cable, per foot - | | $1.00 |
| Hitachi 63B09E CPU and socket - | | $13.00 |
| 512K Upgrades, with RAM chips - | | $72.00 |
| MPI Upgrades for all small MPIs (satellite board) - | | $10.00 |
| Serial to Parallel Convertor with 64K buffer, cables, and external power supply - | **NOW ONLY** | **$28.00!!!** |
| Serial to Parallel Convertor (no buffer), cables, and external power supply - | **ONLY** | **$18.00!!!** |
| 2400 baud Hayes compatible external modems - | | $15.00 |
| Modem cable (4 pin to 25 pin) - | | $5.00 |

**ADD $3.00 S&H FOR FIRST ITEM, $1.00 EACH ADDITIONAL ITEM**

SERVICE, PARTS, & HARD TO FIND SOFTWARE WITH COMPLETE DOCUMENTATION AVAILABLE. INKS & REFILL KITS FOR CGP-220, CANON, & HP INK JET PRINTERS, RIBBONS & vr. 6 EPROM FOR CGP-220 PRINTER (BOLD MODE), CUSTOM COLOR PRINTING.

Terry Laraway
41 N.W. Doncee Drive
Bremerton, WA 98311
360-692-5374

NOTE NEW ZIP CODE AND PHONE NUMBER!

## The BlackHawk MM/1b

Based on the AT306 board from Kreider Electronics. Features built into the motherboard include:

- 16 bit PC/AT I/O bus with five slots
- MC68306 CPU at 16.67MHz
- 512K to 16MB of RAM with 30 pin SIMMs (4 sockets)
- IDE Hard Drive Interface (2 drives)
- 360K-1.44MB Floppy Drive Interface (2 drives)
- Two 16 byte fast serial ports (up to 115K baud)
- Bi-directional parallel printer port
- Real-time clock
- PC/AT keyboard interface
- Standard PC/AT power connector
- Baby AT size - fits standard PC case
- BASIC (resembles Microsoft BASIC)
- MGR Graphical Windowing Environment with full documentation
- "Personal" OS-9/68000 Vr 3.0 (Industrial with RBF)
- Drivers for Tseng W32i and Trident 8900 VGA cards
- Drivers for Future Domain 1680 and Adaptec AAH15xx SCSI cards
- OS-9/68000 Vr 2.4 with Microware C 3.2, Assembler, MW Basic (like Basic09), MW Debug, MW Programmers Toolkit
- UUCP from Bon Billson
- Ghostscript (software PostScript interpreter)
- Many other utilities and tools

Prices start at $400!
(motherboard,
Personal OSK, & MGR only)

BlackHawk
Enterprises, Inc.
756 Gause Blvd. 29
Slidell, LA 70458
E-mail: nimitz@stolY.com

that problem?

Well, there's also a way to cross that obstacle. It just so happens that the GIME chip actually accesses memory 16 bits at a time. That's how we can have those hires 16 color graphics while only needing to access memory twice as fast as on a CoCo2. The GIME manages to pull in twice as much data per memory access for video than the CoCo2 did.

There's a way to use this to your advantage. Every time the CPU reads something from the memory, the GIME actually reads a 16bit value, but only feeds the CPU whichever 8 bits it wants. If we put a 16 bit cache on the data lines going to the GIME from the memory, we can store the whole 16 bit value and then already have the next memory byte for the CPU when it wants to read it during the next bus cycle (90% of the time, the CPU reads memory sequentially, one byte after the next). Always having the second byte in advance, we can give the CPU closer to true double speed operation while still retaining the old 1.789Mhz bus speed. Since we're caching the full 16 bit value read from memory, it's also possible to speed up the times when the CPU reads the same memory location twice in a row, and even some cases where the CPU reads memory locations in reverse order (if any?). With this idea, we can eliminate the 8 bit CPU cache from the previous method, and use this 16 bit memory cache in it's place to get even better speed increases.

Sound good? There is one minor hitch that needs to be worked out, but I don't think it should be too hard. The cache should be smart enough to know when the CPU *isn't* actually reading memory. The problem would arise when the CPU tries to read a 16 bit value from some hardware registers (lets say we're reading the MMU bank values). Upon reading the first byte during one cycle, the cache will try to feed the CPU the next byte in the next CPU cycle - but, the CPU isn't trying to read from memory, so it will be fed the wrong value. One way of avoiding this would be to simply disable the cache from working when the CPU is reading from the $FFxx page of memory (which is where all the hardware registers are located).

Can we go even further? You must think this is getting ridiculous. Since we've already incorporated a cache into the CoCo, it might be a good idea to take better advantage of it. Why not throw in a whole 8K of cache? It gets a little tricky when you have an MMU. The CPU only sees 64k at a time, and if you start swapping different banks of memory in and out, the cache would get awfully confused.

One possibility would be something that might sound a little strange at first, but it works out quite nicely in the CoCo3's case. Sure, put in an 8K cache, but make it 8K by 14 bits. The lower 8 bits of the cache hold the 8 bit value from memory, the upper 6 bits hold the value of the MMU bank that this memory byte originated from. The address bits don't need to be stored because the address will actually be the index for the cache.

Starting to become interesting? This cache will always automatically retain the memory last used by the CPU, with no hardware design hassles. With something like this, Method 3 in the text becomes extremely appealing. It solves all the problems of often waiting for syncronisation with the bus, and gives us the possibility of running the CPU at TRUE 8Mhz with only minimal slowdowns in worst case scenarios!

Thinking ahead, it would also be a good idea to throw in the extra 2 bits of data into the cache so that it could properly handle 2Meg machines, making the cache an even 8K by 16 bits. 8 bits for the data, and 8 bits for the MMU bank value.

The limitation of this idea is that the CPU can still only write to memory at 1.789Mhz. All the speed increases are for memory reads only. But that's what the CPU does most of the time. So the CPU speed will be drastically increased.

More? No. I haven't gotten this far. But the next step would be to allow the cache to buffer memory writes as well. Hence removing the 2nd to last bottleneck from true accelerated CPU performance. The very last bottleneck is the 1.789Mhz bus speed limitation. And that is something to deal with in another time...

John Kowalski (Sock Master)http://www.axess.com/twilight/sock/

Editor's Note: The last bottleneck is the GIME itself, and probably will never be surpassed. The GIME is what makes the CoCo 3. It is a Graphics Interupt Memory Enhancement chip. It controls the CoCo 3's graphics, interupts and memory (as if you couldn't tell from the name!). It will NOT run at more than 2MHz. To do so would require a new chip, which would be very expensive. The only other way to get around this would be to mirror the circuits in "regular" chis. That would be costly and take a lot of circuit board room.

In fact, about the only way to accomplish recreating the GIME with standard components would be to design an entirely new motherboard. It could be done, but the ultimate speedup for the CoCo may just be to buy a 486 clone and Jeff Vavasour's emulator, and if you are into OS-9, to get a 68K based machine. A new CoCo motherboard would ultimately cost as much (or more) than the AT306 (around $450), and still suffer many of the CoCo's limitations.

No, I'm not suggesting you trash your CoCo. It is still the best "hacker's machine" around. It is inexpensive, robust, and easy to repair. The most common peripherals are easy to find. Programming is relatively easy to learn. Play with it, learn with it, and push it to its limits.

**Announcing Nitro Level III!**

**How many times have you been unable to load a driver due to not enough system RAM? How would you like to have up to 32K of system RAM available? How is this possible?**

**In effect, Nitro Level III turns the system into a Kernel only process, with 8K or RAM, and 2 IO processes (RBF and SCF), each with 16K of RAM. This is similar to Grfdrv having it's own 6 k memory area.**

**The kernel process contains the minimum modules to run an OS-9 system, and also the descriptors. The RBF/SCF processes contain the IO modules, and the IO buffers.**

**There are 2 big benefits here:**

**1 - Both RBF and SCF are not in system memory at the same time, so you save RAM.**

**2 - You don't have 16K of SCF or RBF modules, so everything up to 16K can be used as device data storage (sector buffers, etc.)**

**Level III works only with Nitro (all versions). It can be purchased from FARNA Systems alone ($20) or with the latest version of Nitro ($ for Nitro v2.00 and Level III). See the FARNA ad in this issue for ordering information.**

Chet Simpson

# Sprites! Little fairies running all over your computer screen...

<chets@pitnet.net>

Just what is a sprite? In the video game world, a sprite is a graphical object, that when placed onto the screen, seem to mix in with the background. If you look at some of the CoCo III games such as Warrior King or Super Pitfall, you will notice a lot of small graphical images all moving around the screen while the background remains unchanged. These are sprites. Before we can put (or 'blit' In technical terms), we need to know how sprites are stored in memory, what types of data are associated with them and other tidbits of information.

Sprites are defined by 3 variables. Width, height and transparency. Both with and height are evident, but what is transparency? This is the area around the sprite, that is unused and will not be copied to the screen and is usually an unused color slot.

The following is a text representation of an 8 by 8 pixel sprite, that has an unused area around it. The unused area is designated by a period (.) and the actual sprite is an asterisk (*).

```
........      Line 1
...**...      Line 2
..****..      Line 3
.******.      Line 4
.******.      Line 5
..****..      Line 6
...**...      Line 7
........      Line 8
```

On the 16 color screens of the CoCo, graphics are displayed as 2 pixels per byte. The high nibble (4 top bits) is the first pixel and the low nibble (4 bottom bits are the second pixel. Since the example sprite is 8 pixels wide, each line will only take 4 bytes. Total, this is only 32 bytes. On the CoCo's 320x200 16 color screen, each 320 pixel horizontal line only takes up 160 bytes. This is a crucial thing to remember when 'blitting' an image to the screen so that after each line of the sprite is copied, we can skip down to the starting position of the next line.

First lets create a routine that places the sprite onto the screen. The example assumes that certain registers have been set up ahead of time. See the end of this article for the complete source for the examples.

```
* BLIT:
*  register X = location on screen to put the sprite
*  register Y = number of bytes to skip after each line
*  register U = points to the sprite
*  SPR_W and SPR_H hold the width and height (respectively)
*  of the sprite
 BRA   LOOP1
* Start the blit
*      Start copying sprite to screen
LOOP0  LEAX  Y,X
* Adjust the screen down one line
LOOP1  LDB   SPR_W
* Get width of the sprite (In bytes)
LOOP2  LDA   ,U+
* Get a byte from the sprite
 STA   ,X+
* Place it onto the screen
 DECB
* Are we done with this line?
 BNE   LOOP2  * No
 DEC   SPR_H
* Are we done with the sprite?
 BNE   LOOP0  * No
 RTS          * return
```

Seems simple enough. But how do we do transparency? Well, since each byte represents 2 pixels, we can check both pixels (high and low nibbles) to see if they match out transparency color_..or we can create a 'mask'. I prefer using the 'masks' as it is tremendously quicker

In order to create a mask, we look at each pixel of the sprite. Areas that are unused are set to color 16 (hex $f) and values that are used are set to color 0 ($0). This will eliminate having to check each pixel to see if it is transparent.

Those values are very important as logical AND and logical OR operations are used to manipulate the screen data with the mask and sprite data. The way that masks are used is whenever we go to put a byte (2 pixels) onto the screen, we first get the contents of the screen. Then a logical and is done with the mask. This will erase the first, second or both pixels, discarding only the ones the will be replaced by sprite data. We then take the result of the logical and, and or it with the sprite data. Then the data is placed back onto the screen.

If only the first pixel (high nibble) is used by the sprite, the value of the mask is $0F. If the second pixel (low nibble) is used by the sprite, the value of the mask is $F0. If both are used the value is $FF. If the screen data is $AB, the mask data is $F0 and the sprite data is $0C the result (after the logical operations are done) will be $AC.

Here is what the mask would look like for the example sprite:

```
********      Line 1
***..***      Line 2
**....**      Line 3
*......*      Line 4
*......*      Line 5
**....**      Line 6
***..***      Line 7
********      Line 8
```

Notice that it looks (in a manner of speaking) 'inverted.' That is exactly what we want.

So now that we have a mask and we know what needs to be done with it, how do we do put the sprite and the mask together? Well, the best way that I have found for storing masks is right along with the sprite. This can get a little confusing, but is one of the most efficient ways of using masks. When storing both a mask and a sprite together in memory they are stored as mask byte, sprite byte, mask byte, sprite byte, and so one.

In order to use the previous example code to work with masks, we will need to change the following two lines:

```
LOOP2  LDA   ,U+
* Get a byte from the sprite
 STA   ,X+
* Place it onto the screen to:
LOOP2  LDA   ,X
* Get the current contents of the screen
```

```
ANDA  ,U+
* And it with the mask data
ORA   ,U+
* Or  it with the sprite data
STA   ,X+
* Store it to the screen
```

Ok, now that we have the basics for blitting sprites, lets put them to use. I will not be going into how the additional code works as it is primarily for example only and most of what it does will be explained in the next issue.

In the next couple of issues we will delve into very high-speed specialized routines. Some of them include FAST 32k block copies (good copy screen saving), 'compiled' sprites, and something special that plays 8k digital sound samples using less than 26% of the CPU time.

For now, play with the examples given. You might try optimizing them a bit to work with fixed sized sprites, say 16 by 16 pixels. If you are interested in creating sprites, I have written a program specifically for the task, and it does include automatically creating a mask for sprites. It is called Image Master and can be obtained (along with other programs I have written) on Delphi or from my WWW page.

Also, if you have Internet access, I would love to hear your comments and requests. Recently my email address has changed due to my ISP merging with another company. I can now be reached at:

e-mail: chets@pitnet.net
medialink@delphi.com
or: The CoCo Newsfront WWW Site:
http://www.pitnet.net/coco/coco_main.html

**Using the examples**

1) Type in Listing 1 and save it to disk.

2) Using your editor/assembler type in Listing 2 and assemble it as EX12.BIN

3) Using your editor/assembler type in Listing 3 and assemble it as EX13.BIN

When all three programs have been created, run EX11. This will set the video display to 320x192 16 colors, set the palettes, and then load and run the examples. Whenever you hear a beep, the program is expecting you to kit any keep to continue.

## Listing 1: EX11.BAS

(BASIC loader)

```
100 CLEAR2500
110 PALETTE0,1:PALETTE1,9: HSCREEN2: HCLS
120 LOADM"EX12.BIN":GOSUB100
130 LOADM"EX13.BIN":GOSUB100
140 END
200 POKE65497,0:FORX=0TO15:HCOLOR X:
202 HLINE(X*16,0)-(X*16+16,191),HSET,BF
203 NEXTX
204 FOR I=0TO1000
205 POKE&HE00,RND(120):102POKE&HE01,
RND(184)
206 EXEC:NEXTI:POKE65496,0:SOUND10,5
207 I$=INKEY$:IF I$="" GOTO 103
204 RETURN
```

## Listing 2: EX12.ASM

(Blit without transparency)

```
 ORG   $E00
SCREEN EQU   $8000  * Start of screen
SPR_X  RMB   1      * X coordinates
SPR_Y  RMB   1      * Y coordinates
SPR_W  RMB   1      * Sprite width (In bytes)
SPR_H  RMB   1      * Sprite height

SPRITE FCB   $00,$00,$00,$00
 FCB   $00,$01,$10,$00
 FCB   $00,$10,$01,$00
 FCB   $01,$00,$00,$10
 FCB   $01,$00,$00,$10
 FCB   $00,$10,$01,$00
 FCB   $00,$01,$10,$00
 FCB   $00,$00,$00,$00

EXEC   LDD   $FFA4  * Get MMU registers
 LDX   $FFA6  *
 PSHS  D,X,CC  * Save with CC on the stack
 ORCC  #$50   * Disable interrupts
 LDD   #$7879 * Get MMU blocks for the
screen
 STD   $FFA4  * And set the MMU registers
 LDD   #$7A7B
 STD   $FFA6
 LDD   #$0408 * Get width and height of sprite
 STD   SPR_W  * Set it
 LDD   #NMSPR * Point to non masked sprite
 STD   SPRITE



 LDB   #160   * Get number of bytes per line
 SUBB  SPR_W
* Calculate  number of bytes to skip
 LDA   SPR_Y  * Number of lines down
 LDB   #160   * Number of bytes per screen
 MUL          * Multiply
 ADDD  #SCREEN * Add location of the screen
 TFR   D,X    * Send into X
 LDB   SPY_X  * Get X location (In bytes)
 ABX          * Add to X
 LDU   #SPRITE * Put the sprite here
 BRA   LOOP1  * Start the blit
* Start copying sprite to screen
LOOP0  LEAX   Y,X
* Adjust the screen down one line
LOOP1  LDB    SPR_W
* Get width of the sprite (In bytes)
LOOP2  LDA    ,U+
* Get a byte from the sprite
 STA   ,X+    * Place it onto the screen
 DECB         * Are we done with this line?
 BNE   LOOP2  * No
 DEC   SPR_H  * Are we done with the sprite?
 BNE   LOOP0  * No
 PULS  D,X,CC
* Get registers and turn the interrupts on
 STD   $FFA4
* And set the MMU registers back
 STX   $FFA6
 RTS          * return
 END   EXEC   * End of listing
```

## Listing 3: EX13.ASM

(Blit with transparency)

```
 ORG   $E00
SCREEN EQU   $8000  * Start of screen
SPR_X  RMB   1      * X coordinates
SPR_Y  RMB   1      * Y coordinates
SPR_W  RMB   1      * Sprite width (In bytes)
SPR_H  RMB   1      * Sprite height

SPRITE FCB
$FF,$00,$FF,$00,$FF,$00,$FF,$00
 FCB   $FF,$00,$F0,$01,$0F,$10,$FF,$00
 FCB   $FF,$00,$0F,$10,$F0,$01,$FF,$00
 FCB   $F0,$01,$FF,$00,$FF,$00,$0F,$10
 FCB   $F0,$01,$FF,$00,$FF,$00,$0F,$10
 FCB   $FF,$00,$0F,$10,$F0,$01,$FF,$00
 FCB   $FF,$00,$F0,$01,$0F,$10,$FF,$00
 FCB   $FF,$00,$FF,$00,$FF,$00,$FF,$00

EXEC   LDD   $FFA4  * Get MMU registers
 LDX   $FFA6  *
 PSHS  D,X,CC  * Save  with CC on the stack
 ORCC  #$50   * Disable interrupts
 LDD   #$7879
* Get MMU blocks for the screen
 STD   $FFA4  * And set the MMU registers
 LDD   #$7A7B
 STD   $FFA6
 LDD   #$0408 * Get width and height of sprite
 STD   SPR_W  * Set it
 LDD   #NMSPR * Point to non masked sprite
 STD   SPRITE

 LDB   #160   * Get number of bytes per line
 SUBB  SPR_W
* Calculate number of bytes to skip
 LDA   SPR_Y  * Number of lines down
 LDB   #160   * Number of bytes per screen
 MUL          * Multiply
 ADDD  #SCREEN * Add location of the screen
 TFR   D,X    * Send into X
 LDB   SPY_X  * Get X location (In bytes)
 ABX          * Add to X
 LDU   #SPRITE * Put the sprite here
 BRA   LOOP1  * Start the blit
* Start copying sprite to screen
LOOP0  LEAX   Y,X    * Adjust the screen
down one line
LOOP1  LDB    SPR_W  * Get width of the
sprite (In bytes)
LOOP2  LDA    ,X     * Get the current
contents of the screen
 ANDA  ,U+    * And it with the mask data
 ORA   ,U+    * Or  it with the sprite data
 STA   ,X+    * Store it to the screen
 DECB         * Are we done with this line?
 BNE   LOOP2  * No
 DEC   SPR_H  * Are we done with the sprite?
 BNE   LOOP0  * No
 PULS  D,X,CC
* Get registers and turn the interrupts on
 STD   $FFA4
* And set the MMU registers back
 STX   $FFA6
 RTS          * return
 END   EXEC   * End of listing
```

# DATREDER and DATEDTR

P.B. Blackwell

## Read and edit .DAT files without the creating program!

DATREDER is a short program designed to read DAT and TXT files directly from the disk. It has an option to either print to the screen ot to the printer. Most of the necessary instructions are found in the body of the program. A few additional precautions/instructions, however, cannot do harm.

If your system has trouble with the high speed poke [ POKE 65497,0 ], this should be edited out. Also, if your drives will not work at 6ms, these pokes [ POKE 55232,0 :POKE 55318,20 ] should be edited also. The high speed poke and the disk timing pokes are found in Line 100.

The printer options have been set up for the Radio Shack DMP-105. If you have a different printer, you will probably need to change these options. The printer options are found in Line 380 and Line 390.

DATEDTR allows you to read data from a disk file. As you read the data you have the option to (a) retain the odl data, as is; (b) change the name (or number) of the data (of course, you lose the old data); (c) add data to the file and; (d) delete data from the file.

With DATEDTR you can go to and remain in lower case mode throughout the program, except when setting up "file name/ext" at the beginning.

The same precautions/instructions above (DATREDER) applies to both programs, except that DATEDTR has no printer options.

If you press <Q> or <q> or <BREAK> or you experience an error, the programs will (1) preserve your old file; (2) save any additions you have made; (3) CLOSE your files; (4) clear the screen and (5) give you a DIRectory listing at the point that whichever occurred. I am not sure whether this works in case of a drastic ("FATAL") error. I tried to program in as much safety as I knew how. These programs were written on and for the CoCo 3, however with some editing and manipulating, that the novice can do, these should work equally well for the CoCO 1 and CoCo 2.

If anyone does not want to type in the programs, you may send me $1.00 plus shipping ($0.55), and I'll backup you a copy for the CoCo 3. Better yet — subscribe to FARNA Systems '68 micros disk. Mr. Swygert and the CoCo's need all the support they can get!

Happy CoCoing, gentle beings!!!

P.B. Blackwell
1408 McFadden St.
Paris TN 38242-3210

```
5 WIDTH 40:CLS 4:'DATREDER
10 PRINT"********** Pete Blackwell **********
15 PRINT"****** 1408 1/2 McFadden St."*********
20 PRINT"******* Paris  TN 38242-3210"*********
25 PRINT"********** JUNE 29, 1993
**********": PRINT:PRINT
30 PRINT"********** <press a key> **********
:PRINT
35 EXEC 44539
40 CLS 8
45 PRINT"!!!!!!!!!!! CAUTION !!!!!!!!!!!!!!!!!":
PRINT
50 PRINT"  If your system will not accept the
high speed poke and/or your drives will not
operate at 6ms; EDIT Line 100 for satisfactory
operation."
55 PRINT:PRINT"-------------------------
----------":PRINT:PRINT
60 PRINT"  This program should work, with
no problem, on the CoCo 1 and CoCo 2. You
will, however, have to EDIT Lines 5  & 100;
DELETE  Line 110 and EDIT the prompt and
instruction Lines to properly";
65 PRINT"  fit the 32 column
screen.":PRINT
70 PRINT:PRINT"********** <press a key>
****** *******"
75 EXEC 44539
100 POKE 65497,0:POKE 55232,0:POKE
55318,20
110 ON BRK GOTO 340:ON ERR GOTO 340
120 N=0:M$="L3T5O3V30GF":L$="L5T5O3V
30EDG"
130 PLAY L$+M$
140 CLS:PRINT"..... Disk File Reader .....":
PRINT
150 PLAY M$
160 INPUT"Name of file to be read ";X$
170 PLAY M$
180 INPUT"EXT (do not use / or .)";Y$
190 PLAY M$
200 INPUT"Drive # (do not use :)";Z$
210 Z=VAL(Z$):K$=X$+"/"+Y$+":"+Z$
220 PLAY L$:PRINT PRINT"Send to printer
<1>=no::  <2>=yes:?";
230 INPUT P : IF P=2 THEN 380
240 OPEN"I",#1,K$
250 PLAY L$:CLS 3:PRINT K$:PRINT
260 IF EOF(1)=-1 AND P=1 THEN 310 ELSE
IF EOF(1)=-1 AND P=2 THEN 440
270 INPUT #1,A$
280 N=N+1 : IF P=2 THEN 420
290 PLAY M$:PRINT:PRINT A$
300 GOTO 260
310 PLAY M$
320 PRINT:PRINT"Total data items:";N
330 CLOSE #1
340 PLAY L$:PRINT: PRINT"Read another?
<Y>::  End <N>::  DIR <D>::";
350 I$=INKEY$:IF I$="Y"THEN 120 ELSE IF
I$="N"THEN 360 ELSE IF I$="D" THEN 370
ELSE 350
360 PLAY M$+L$ : CLS 5 : END
370 CLS 8 :PLAY L$+M$ : DIR Z :GOTO 340
380 POKE 150,41
390 PRINT#-2,CHR$(27);CHR$(23)
400 PRINT#-2,TAB(15);CHR$(15);K$;CHR$
(14):PRINT#-2
410 GOTO 240
420 PRINT#-2,TAB(15);A$
430 GOTO 260
440 PRINT#-2: PRINT#-2,TAB(15)"Total data
items:";N
450 GOTO 330
```

```
5 WIDTH 40:CLS 4:'DATEDTR
10 PRINT"********** Pete Blackwell **********
15 PRINT"***** 1408 1/2 McFadden St. *** ****
20 PRINT"****** Paris  TN 38242-3210 ********
25 PRINT"********* JUNE 29, 1993 **********":
PRINT:PRINT
30 PRINT"********** <press a key> **********":
PRINT
35 EXEC 44539
40 CLS 8
45 PRINT"!!!!!!!!! CAUTION !!!!!!!!!!!!!":PRINT
50 PRINT"  If your system will not accept the
high speed poke and/or your drives will not
operate at 6ms; EDIT Line 100 for satisfactory
operation."
55 PRINT:PRINT"-------------------------
----------":PRINT:PRINT
60 PRINT"  This program should work, with no
problem, on the CoCo 1 and CoCo 2. You will,
however, have to EDIT Lines 5  & 100; DELETE
Line 110 and EDIT the prompt and instruction
Lines to properly";
65 PRINT"  fit the 32 column screen.":PRINT
70 PRINT:PRINT"******* <press a key> **********
75 EXEC 44539
100 POKE 65497,0:POKE 55232,0:POKE
55318,20
110 ON BRK GOTO 430:ON ERR GOTO 430
120 L$="L3T5O3V30GF":M$="L3T5O1V30B-E"
130 PLAY L$+M$+M$+L$:CLS 3
140 PRINT"...<DATEDTR>.. Change/Add/
Delete data":PRINT
150 PLAY L$
160 INPUT"Name of file (do not use /, ., or
:)...";F$
170 PLAY M$
180 INPUT"EXT.....";E$
190 PLAY L$
200 INPUT"Drive # .. (0-5)";D$
210 D=VAL(D$):K$=F$+"/"+E$+":"+D$
220 OPEN"I",#1,K$
230 OPEN"O",#2,"NEW/DAT"+":"+D$
240 IF EOF(1)=-1 THEN PRINT:PRINT"***** No
more data on disk *****":PRINT:GOTO 280
250 INPUT#1,A$
260 PLAY"T60L1C":CLS 3:PRINT K$:PRINT
270 PRINT"..... Data Item....";A$
280 PRINT:PRINT"Press <N>=no change::
<A>=add data::  <C>=change
data::<D>=delete data  <Q>=quit::"
290 I$=INKEY$:IF I$="N"OR I$="n"THEN 400
ELSEIF I$="A"OR I$="a"THEN 370
ELSEIF I$="C"OR I$="c"THEN 330
```

*listing continued on page 18*

# DECBIN.BAS

Richard Albers

## Convert decimal numbers to binary with the clever two liner!

**Decimal to Binary Convertor listing:**

```
1 PRINT:PRINT:INPUT"NUMBER";N:
IFN= INT (N) ANDN>0THEND=INT
((LOG(N+.001)/LOG(2)+4)/4)*4:C=INT
(2^D):PRINT"BINARY=";:FORX=1TOD:
C=C2: PRINTCHR$(49+(N<C));:N=N+C*
(N=>C):IFX=INT(X/4)*4THENPRINT" ";
:NEXT: GOTO1ELSENEXT:GOTO1
ELSE PRINT"+INTS>0 ONLY";:GOTO1
2 PRINT:INPUT"BINARY";B$:N=0:FOR
X=1TOLEN(B$):A$=MID$(B$,X,1):B=
INSTR ("01",A$): IFB THENN=N+N+B1:
NEXT:PRINT"DECIMAL ="N:GOTO2
ELSEIFA$<>" "THEN PRINT" "A$" IS
NOT BINARY":GOTO2 ELSE NEXT:
PRINT "DECIMAL="N:GOTO2
```

**Explanation of listing:**

Two PRINTs put some blank space on the screen. I didn't use a CLS because I often want to use the result of one conversion as input or inspiration for another. This way I don't need to write down any of the results. Also, because the program auto-repeats, I would have to read VERY fast before the repeat erased the screen!

Next I get a number in "N" to convert and test it for positive integer-hood. This small program can't handle fractionsor negative numbers, so I remind the user and start over.

Once a positive integer is in "N", the conversion begins. The function "INT(LOG(N+.1 )/LOG(2)" calculates the number of binary digits needed to contain the value of "N". "+4", "/4", "*4", "-4" forces "D" to be a multiple of four so I can print the binary result as nybbles; groups of four digits. That's much easier to read than a long string of only zeros and ones. It's also easy to convert nybbles to hexadecimal if I need to.

That "-4*(N<1)" is an example of an embedded function, similar to common "C" language practice. I could have used "IFN<1..." but that would take longer and prevent putting the whole program on one line. The "INT(LOG(...))"function returns a zero if "N" is zero, and I wanted to print four zeros, so "-4*(N<1)" changes the digit count. "(N<1)" can only return a zero or a minus one (try it yourself in immediate mode: "N=0:?(N<1)<enter>" (repeat with N=1, N=2)). If "N" is less than one (zero is the only possible value -- I tested for negative) then "(N<1)" will return a -1. -1 times -4 equals +4, which will be added to the zero from "INT(LOG(...))" and force 4 digits to be printed. If "N" is equal to or greater than one, "(N<1)" will return a zero; zero times -4 is zero, leaving the precomputed number of digits unchanged.

"C" is the value of the binary digit to be printed next. It is compared to "N" to determine whether to print a "1"or a "0".

The output begins with printing "BINARY=" without a newline. The binary number will be computed and printed left-to-right, or most significant digit to least significant digit. The hard way.

"FORX=1TOD" just counts the binary digits; pretty obvious. The next two statements are less obvious because of the embedded functions. Lets look at them the way BASIC will -- right to left.

"N<C" is the test that decides whether to print a zero or a one, the only possible digits in binary. If "N" is less than "C", print a zero. If "N" is equal to or greater than "C", print a one. "N<C" will return -1 if "N" really is less than "C" and a zero for all other conditions. "PRINTCHR$(49)" will print a "1" (try it) which should happen only if "N" is equal to or greater than "C". If "N<1" returns "-1" because "N" is less than "C", a minus one is added to 49, giving 48, and "PRINTCHR$(48)" prints a "0".

Now that this digit has been printed, it's time to adjust "N" and "C" in preparation for printing the next digit. We should subtract the value of "C" from "N" if we just printed a "1", but not if we just printed a "0". "N=N+C*(N=>C)" does just that. See if you can figure out how it does it.

"C=C/2" adjusts the value of "C". The value of a binary digit is half of the value of the digit to its left. "IFX=INT(X/4)*4" tests whether we have printed four digits; if so, it's time to print a space.

"NEXT:GOTO1ELSENEXT:GOTO1" closes both possible results of the "IF" and allows the program to exist on a single line. This construction should only be used near the end of a line, since a lot of program function must be repeated after the "ELSE" otherwise. In this program, that is only two commands. If you want to use this program as a subroutine, replace the "GOTO1"s with "RETURN"s.

Did you determine how "N=N+C*(N=>C" works? If you want to go back and try, I'll wait for you here... OK, here is my version: If "N" is equal to or greater than "C" (we just printed a "1"), "N=>C" will return a minus one. "C" times "-1" equals "-C", "N+(-C)" equals "N-C" and we d subtract "C" from "N" as we should. If "N" is less than "C" (we just printed a "0"), "N=>C" will return zero, "C" times zero is zero, and "N+0" leaves "N" unchanged. Did you get that?

Now try this on your own BASIC programs. You will reduce the number of variables and maybe speed up execution too. You should test each embedded function, either by typing in immediate mode, or by writing a short version of it, initializing your variables to critical boundary values.

Just be sure to test ALL possibilities! I have revised this program from time to time over several years, but hadn't realized until I started writing this article that I had never tried it with a negative input. That proved the need to modify it yet again, and I just managed to squeeze it into a single line by shortening the error message.

---

DATEDTR Listing

continued from page 17

```
ELSEIF I$="D"OR I$="d"THEN 300
ELSEIF I$="Q"OR I$="q"THEN 430
ELSE290
300 CLS 8:PRINT" Old data name: ";A$
310 PRINT:PRINT"!!!! Are You Sure? <Y/
N>";:PRINT
320 S$=INKEY$:IF S$=""THEN 320 ELSEIF
   S$="Y"OR S$="y"THEN 240 ELSEIF
   S$="N"OR S$="n"THEN CLS 3 :
    GOTO 260
330 CLS 6:PRINT"Old data name: ";A$
340 PRINT:PRINT"... Change data to: ";
350 INPUT X$
360 GOTO 410
370 CLS 8:PRINT:PRINT"New data to add:";
380 INPUT X$
390 GOTO 410
400 X$=A$
410 WRITE#2,X$
420 GOTO 240
430 CLOSE#1:CLOSE#2
440 KILL K$
450 RENAME"NEW/DAT"+":"+D$ TO K$
460 PLAY M$+L$+L$+M$ : CLS 5 : DIR
```

## RGBoost - $15.00

If you want to speed up DECB easily, install an Hitachi 6309 and get RGBoost. This patch for DECB uses the extra 6309 functions for up to a 15% gain in overall speed. It is compatible with all programs tested to date! Save an additional $5 by purchasing RGBoost along with one of my other products listed below!

**EDTASM6309 v2.02 - $35.00**
Patches Tandy's Disk EDTASM to support Hitachi 6309 codes! Supports all CoCo models, including stock 6809 models. CoCo 3 version uses 80 column screen, runs at 2MHz. YOU MUST HAVE A COPY OF DISK EDTASM. This is a PATCH ONLY! It will not work with "disk patched" cartridge EDTASM

**CC3FAX - $35.00**
Receive and print weather fascimile maps from shortwave! The US weather service sends them all the time! Requires 512K CoCo3 and shortwave receiver. Instructions for simple cable included.

**HRSDOS - $25.00**
Move programs and data between DECB and OS-9 disks! Supports RGB-DOS - move files easily between DECB and OS-9 partitions! No modifications to OS-9 modules required.

**DECB SmartWatch Drivers - $20.00**
Access your SmartWatch from DECB! Adds function to BASIC (DATE$) for accessing date and time. *Only $15.00 with any other purchase!*

Robert Gault
832 N. Renaud
Grosse Pointe Woods, MI 48236
313-881-0335
*Please add $4 S&H per order*

*Get more from your CoCo OS-9 Level II system with help from...*

## Chris Dekker

**BasicBoost :** 6309 port of Basic09's RunB module. Packed programs are 10% to 15% faster (varies with functions used)!

**ScreenBoost:** 6309 version of Level II screen drivers. Noticeably speeds up most functions! Adds support for up to 200 graphic lines and 28 by 128 column text screens, plus horizontal scrolling. New commands to manipulate graphic fonts and one that allows programs to move and resize the window in which they run.

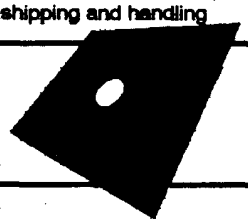
**$10 each or $16 for the pair**

**HSLINK:** Null-modem file transfers between a CoCo and any other computer. Uses CoCo printer port – no special hardware required! ASCII and Xmodem transfers up to 19,200 bps - 57,600 bps with 6309! Cables can be made on request (specify ends needed) or use your own null-modem cables.

**$14.95 or $24.95 with cable**

Prices are in US dollars. Call or write for Canadian dollar prices.
Add $3 US, $5 Canada for shipping and handling

**C. Dekker**
**RR #4**
**Centreville, NB E0J 1H0**
**CANADA** Phone 506-276-4841



*for all your CoCo hardware needs, connect with*

## CoNect

1629 South 61st Street
West Allis, WI 53214
(pulland@omnifest.uwm.edu)

That thing that Tandy calls a serial port on the CoCo has always been a problem. It was designed with minimal cost in mind, and never upgraded. Even Tandy tried to fix it with their RS-232 Pak, but even it was only half done! Our Fast 232 port uses a 16 byte buffer at alleviate missed characters at any speed and also has ALL RS-232 lines implemented. It is easy to set up with jumpers for different addresses. A daughterboard can be purchased to easily add a second fast serial port! And all this in a cartridge the size of a ROM Pak! 6809 and 6309 OS-9 drivers included. Completely supports up to 57,600 bps, limited support for 115,000 bps.

**Fast 232 - $79.95**
**Daughter Board - $45.00**

*Check with us for complete disk drive systems, misc. hardware items, hardware repairs, and hard to find new and used CoCo software!*

## ADVERTISER'S INDEX



## *What are you waiting for?*

**Get your friends to subscribe to the only magazine that still supports the Tandy Color Computer... *"the world of 68' micros"*!**
**The more people who want the support, the longer it will be here!**

# GET READY! IT IS ALMOST TIME FOR THE...

# Sixth Annual "Last" Chicago CoCoFest

The event of the year is to be held at the Holiday Inn, 345 West River Road, Elgin, IL. Call 847-695-5000 for reservations. If you are just shopping or browsing, come by any time Saturday, April 26 between 10am and 5 pm or Sunday, April 27 from 10 am until 3:30pm. Vendors may come as early as 5:30am on Saturday for booth setup. General admission is $5 for Glenside members, $10 all others, one day or both. Room rates are only $57.00 per night (ask for the CoCoFest rate!). Vendor booths are $35 each.

Please send fees to George Schneweiss, Treasurer, Glenside Color Computer Club, RR#2 Box 67, Forrest, IL 61741-9629. All monies should be received no later than March 6, 1997. Vendors should send $15 to reserve a booth. Additional booth spaces are $30, additional vendor passes $5 (maximum 4 per vendor). Balance of booth rental is due by April 10th, 1997. No refunds after April 10th. For more information contact Tony Podraza, 119 Adobe Circle, Carpentersville, IL 60110-1101. Phone 708-428-3576 or Eddie Kuns, phone 708-820-3943, e-mail eddiekuns@delphi.com.

Will this really be the "last" Chicago CoCoFest?
Just in case, YOU DON'T WANT TO MISS IT!!!

---

# NEW STUFF FROM FARNA SYSTEMS!!!

FARNA Systems is pleased to announce that we are now distributors of the following, formerly from Northern Exposure! Note: If you never received your order from NX, send a copy of yourcancelled check along with $5 to cover S&H and I'll fill the order! All other orders add $3 US/Mexico, $4 Canada, or $10 Overseas for S&H.

Nitro v2.00 : $35.00
A complete rewrite of OS-9 Level II that takes advantage of all features of Hitachi's 6309 processor. Easy install script! 6309 required.

Nitro Level III : $20.00
($45 if purchased with v2.00)
Just what you always needed... MORE SYSTEM RAM! Up to 32K (16K for RBF, 16K for SCF devices)!

TuneUp : $20.00
If you don't have a 6309, you can still take advantage of some of the Nitro software technology! Many OS-9 Level II modules rewritten for improved speed with the stock 6809!

Rusty : $20.00
Launch DECB programs from OS-9! Allows loading of some programs from hard drive! NOTE: DECB Programs will still use floppy drives if required for data.

Thexder OS-9 or
Shanghai OS-9 : $25.00 each
Transfers your ROM Pack game code to an OS-9 disk! Please send manual or ROM Pack to verify ownership of original program.

FARNA Systems
Post Office Box 321
Warner Robins, GA 31099
Phone: 912-328-7859
E-mail: dsrtfox@delphi.com