'68' MICRO JOURNAL

$2.95 USA

Australia	A $ 4.75	New Zealand	NZ $ 6.50
Singapore	S $ 9.45	Hong Kong	H $23.50
Malaysia	M $ 9.45	Sweden	30:-SEK

VOLUME VII ISSUE IV • Devoted to the 68XX User • April 1985

"Small Computers Doing Big Things"

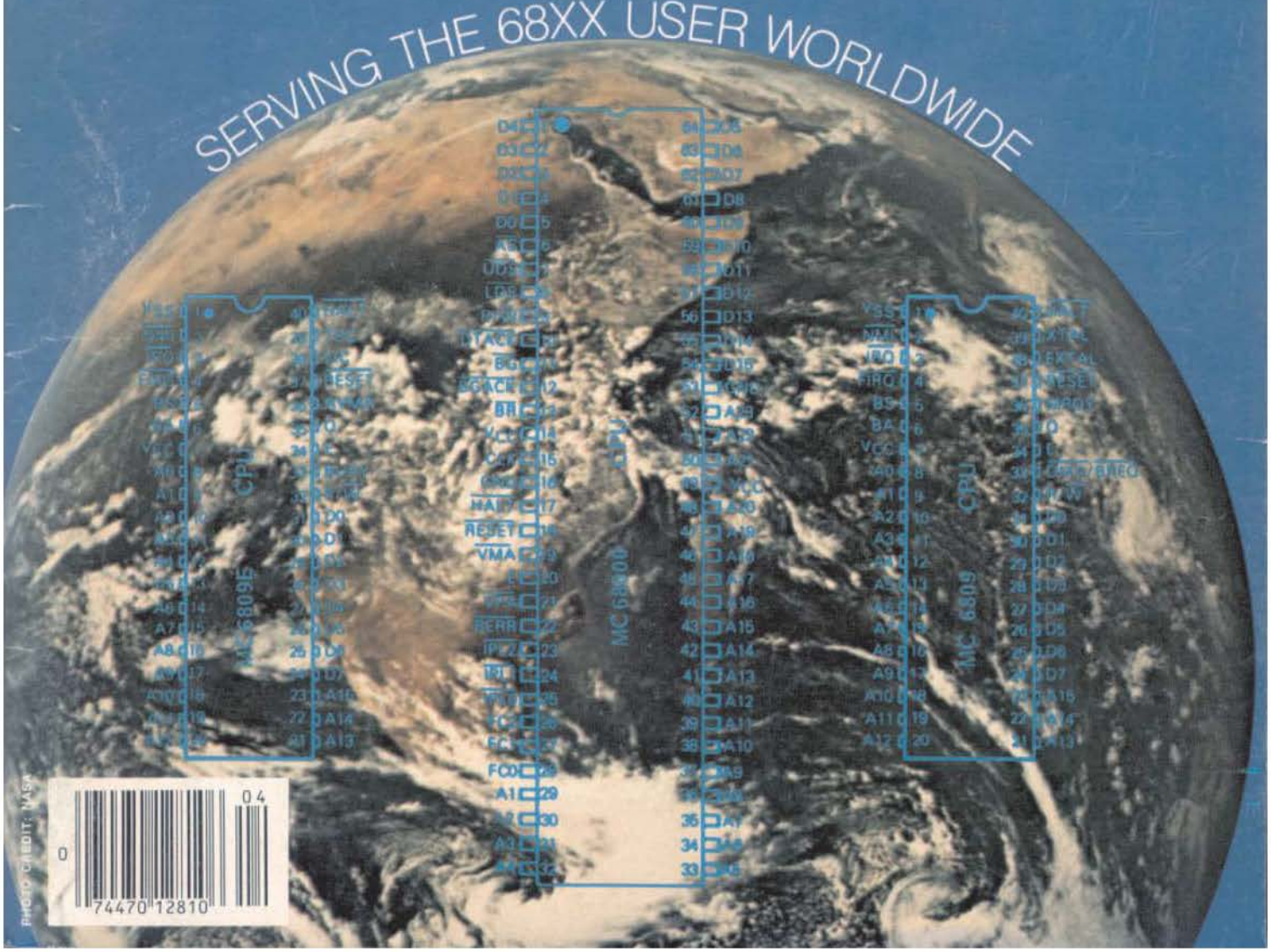


PHOTO CREDIT: NASA

0 74470 12810 04

WE DON'T PLAY GAMES

X-12+ A SERIOUS COMPUTER IN A DESKTOP PACKAGE

Multiprocessor Technology - Combination of 8, 16 and 32 bit types

1.0 Megabyte Memory - Insures no limitation on programs

"Winchester" Disk System - Fast response, large storage capacity

UniFlex* Operating System - The standard of comparison

Hardware Floating Point - Unmatched speed in a small system

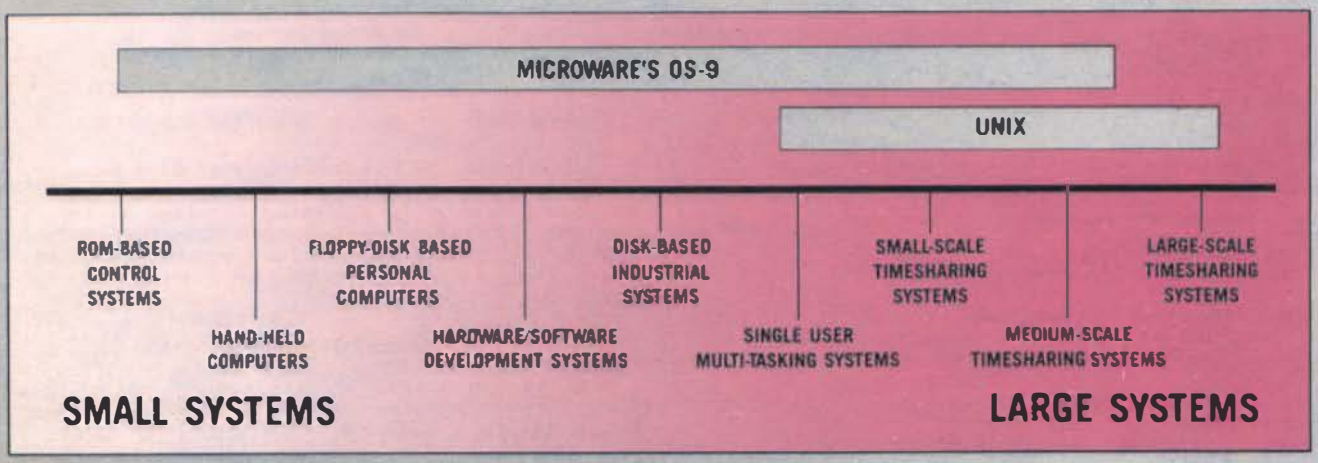
Up to Three Terminals - Instant expansion

* Trademark of Technical Systems Consultants

SWTPC

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

Only Microware's OS-9 Operating System Covers the Entire 68000 Spectrum



Is complicated software and expensive hardware keeping you back from Unix? Look into OS-9, the operating system from Microware that gives 68000 systems a Unix-style environment with much less overhead and complexity.

OS-9 is versatile, inexpensive, and delivers outstanding performance on any size system. The OS-9 executive is much smaller and far more efficient than Unix because it's written in fast, compact assembly language, making it ideal for critical real-time applications. OS-9 can run on a broad range of 8 to 32 bit systems based on the 68000 or 6809 family MPUs from ROM-based industrial controllers up to large multiuser systems.

OS-9'S OUTSTANDING C COMPILER IS YOUR BRIDGE TO UNIX

Microware's C compiler technology is another OS-9 advantage. The compiler produces extremely fast, compact, and ROMable code. You can easily develop and port system or application software back and forth to standard Unix systems. Cross-compiler versions for VAX and PDP-11 make coordinated Unix/OS-9 software development a pleasure.

SUPPORT FOR MODULAR SOFTWARE — AN OS-9 EXCLUSIVE

Comprehensive support for modular software puts OS-9 a generation ahead of other operating systems. It multiplies programmer productivity and memory efficiency. Application software can be built from individually testable software modules including standard "library" modules. The modular structure lets you customize and reconfigure OS-9 for specific hardware easily and quickly.

A SYSTEM WITH A PROVEN TRACK RECORD

Once an underground classic, OS-9 is now a solid hit. Since 1980 OS-9 has been ported to over a hundred 6809 and 68000 systems under license to some of the biggest names in the business. OS-9 has been imbedded in numerous consumer, industrial, and OEM products, and is supported by many independent software suppliers.

Key OS-9 Features At A Glance

- Compact (16K) ROMable executive written in assembly language
- User "shell" and complete utility set written in C
- C-source code level compatibility with Unix
- Full Multitasking/multiuser capabilities
- Modular design - extremely easy to adapt, modify, or expand
- Unix-type tree structured file system
- Rugged "crash-proof" file structure with record locking
- Works well with floppy disk or ROM-based systems
- Uses hardware or software memory management
- High performance C, Pascal, Basic and Cobol compilers

microware® OS-9™

MICROWARE SYSTEMS CORPORATION
1866 NW 114th Street
Des Moines, Iowa 50322
Phone 515-224-1929
Telex 910-520-2535

Microware Japan, Ltd
3-8-9 Baraki, Ichikawa City
Chiba 272-01, Japan
Phone 0473(28)4493
Telex 299-3122

'68' MICRO JOURNAL

Portions of the text for '68' Micro Journal were prepared using the following furnished Hard/Software:

COMPUTERS - HARDWARE

Southwest Technical Products
219 W. Rhapsody
San Antonio, TX 78216
S09 - 5/8 DMF Disk - COSI - 8212W - Sprint 3 Printer

GIMIX Inc.
1337 West 37th Place
Chicago, IL 60609
Super Mainframe - OS9 - FLEX - Assorted Hardware

EDITORS - WORD PROCESSORS

Technical Systems Consultants, Inc.
111 Providence Road
Chapel Hill, NC 27514
FLEX - Editor - Text Processor

Great Plains Computer Co., Inc.
PO Box 916
Idaho Falls, ID 83401
Stylograph - Mail Merge - Spell

CONTENTS

Send All Correspondence To:

Computer Publishing Center
68' Micro Journal
5900 Cassandra Smith Rd.
Hixson, Tn. 37343

Phone (615) 842-4600 or Telex 558 414 PVT BTH

68' Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage Paid ISSN 0194-5025 at Hixson, Tn. and additional entries. Postmaster: send form 3597 to 68' Micro Journal, POB 849 Hixson, Tn. 37343.

Subscription Rates

1 Year $24.50 U.S.A., Canada & Mexico Add $5.50 a Year. Other Foreign Add $12 a Year for Surface, Airmail Add $36 a Year. Must be in U.S. currency.

Items or Articles For Publication

Articles submitted for publication should include authors name, address, telephone number and date. Articles should be on either 5 or 8 inch disk in STYLOGRAPH or TSC Editor format with 3.5 inch Column width. All disks will be returned. Articles submitted on paper should be 4.5 inches in width (including Source Listings) for proper reductions. Please Use A Dark Ribbon!! No Blue Ink!!! Single space on 8X11 bond or better grade paper. No hand written articles accepted. Disks should be in FLEX2 6800 or FLEX9 6809 any version or OS-9 any version.

The following TSC Text Processor commands ONLY should be used: .sp space, .pp paragraph, .fi fill and .nf no fill. Also please do not format within the text with multiple spaces. We will enter the rest at time of editing.

All STYLOGRAPH commands are acceptable except .pg page command. We print edited text files in continous text form.

Letters To The Editor

All letters to the editor should comply with the above requirements and must be signed. Letters of "gripes" as well as "praise" are solicited. We reserve the right to reject any submission for lack of "good taste" and we reserve the right to define "good taste".

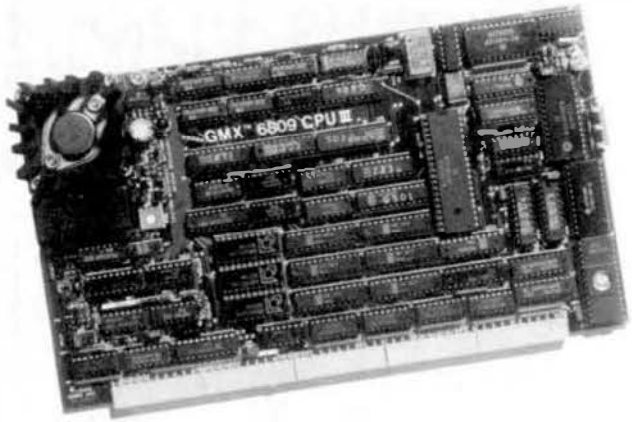
Advertising Rates

Commercial advertisiers please contact 68' Micro Journal advertising department for current rate sheet and require nts.

Classified Advertising

All classified ads must be non-commercial. Minimum of $7.50 for first 20 words and .35 per word after 20. All classifieds must be paid in advance. No classified ads accepted over the phone.

GIMIX STATE OF THE ART 6809 SYSTEMS FOR THE SERIOUS USER.



GIMIX has 19MB or high performance 47MB Winchester Drive Systems and/or Floppy Disk Drive Systems.

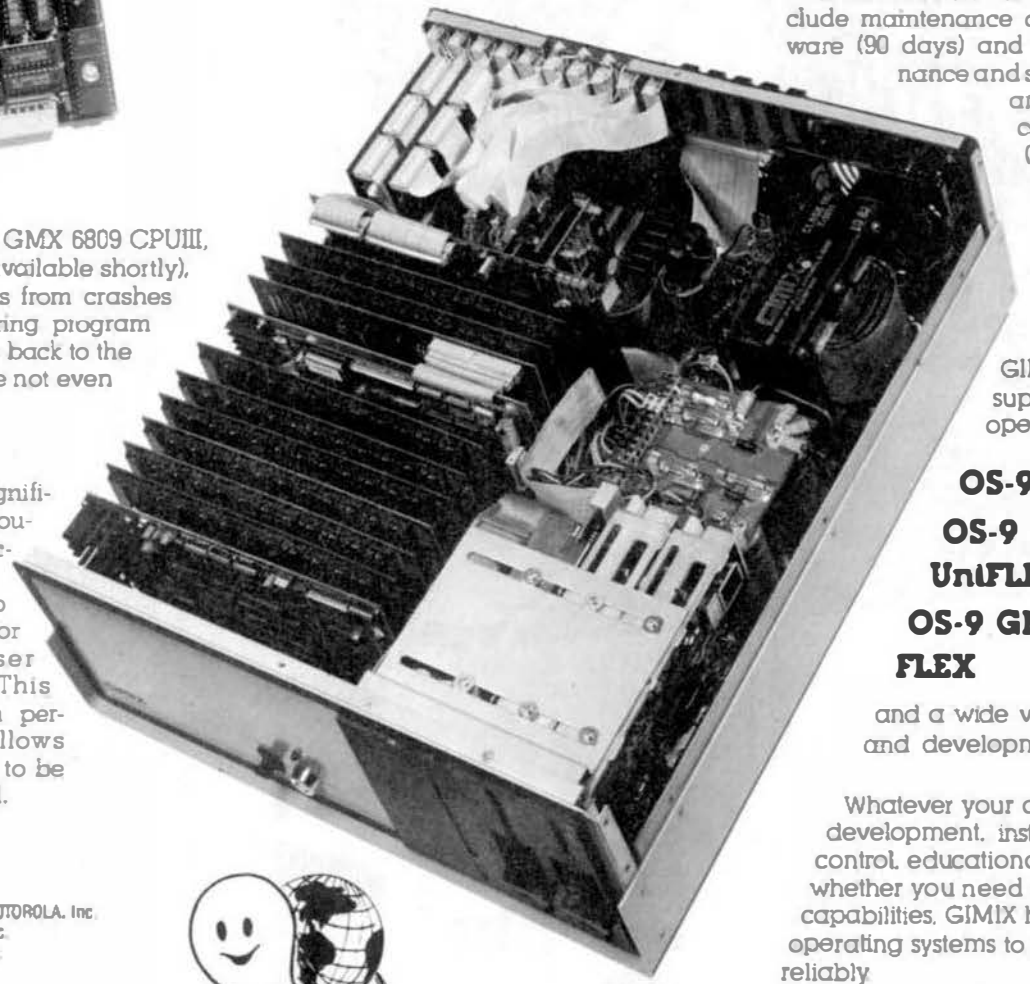
For the user who appreciates the need for a bus structured system using STATIC RAM and powered by a ferro resonant constant voltage transformer.

GIMIX has single user systems that can run both FLEX and OS-9 or Multi user systems for use with UniFLEX or OS-9.

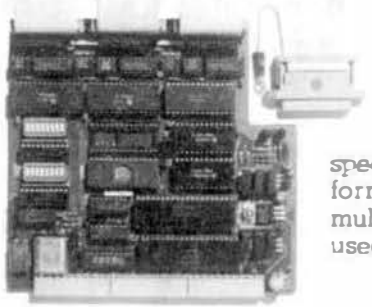
GIMIX versions of OS9 and UniFLEX include maintenance and support by Microware (90 days) and TSC (1 year). Maintenance and support after this period are available at extra cost.

(NOTE: this support and maintenance is only for use with approved GIMIX hardware)

For the ultimate in performance, the Unique GMX 6809 CPUIII, using either OS-9-GMXIII or UniFLEX GMXIII (available shortly), gives protection to the system and other users from crashes caused by defective user programs. e.g. During program development, a programmer who crashes goes back to the shell or the debugger, while the other users are not even aware anything occurred.



The intelligent serial I/O processor boards significantly reduce system overhead by handling routine I/O functions, thereby freeing up the host CPU for running user programs. This speeds up system performance and allows multiple terminals to be used at 19.2K baud.



GIMIX 6809 systems support five predominant operating systems:

OS-9 GMX III,
OS-9 GMX II,
UniFLEX,
OS-9 GMX I,
FLEX

and a wide variety of languages and development software.

Whatever your application: software development, instrumentation, process control, educational, scientific or business; whether you need single or multi-user capabilities, GIMIX has hardware and the operating systems to get the job done reliably.

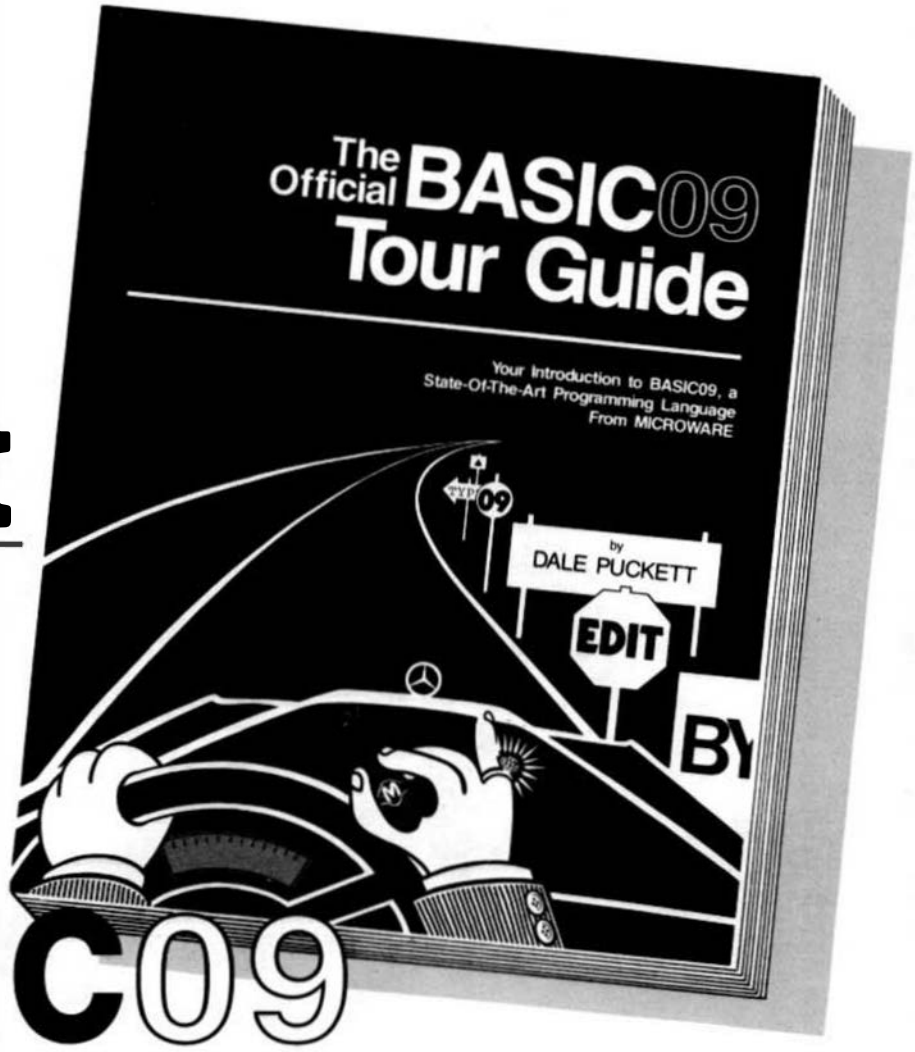
Please phone or write if you need further information.

BASIC-09 and OS-9 are trademarks of Microware Systems Corp. and MOTOROLA, Inc.
FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc.
GIMIX, GHOST, GMX, CLASSY CHASSIS, are trademarks of GIMIX, Inc.

GIMIX INC.

1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609 • (312) 927-5510 • TWX 910-221-4055

Get the most out of BASIC09



The **OFFICIAL BASIC09 TOUR GUIDE** is skillfully written in a friendly and easy-to-read style. Just perfect for those new to computers and to BASIC09. It's also a *valuable reference book* for programmers, engineers, students and hobbyists, providing an in-depth look at BASIC09 plus an overview of the OS-9 operating system. Comprehensive reference sections on BASIC09 and OS-9 commands are also included.

The book "maps" out your route through the Mercedes of Basics . . . BASIC09 and puts you in the driver's seat in no time. Fasten your seatbelt, sit back and enjoy the ride to perfecting your programming skills.

MICROWARE . . .

The **OFFICIAL BASIC09 TOUR GUIDE** comes from the people who wrote BASIC09. As the leader in 6809 system software, we at MICROWARE care about our users and want to help you get the most from our products.

It's Easy to Order.

Phone orders are accepted from MasterCard or VISA cardholders or for COD shipment. You can also order by mail using the coupon below. Quantity discounts are available to educational organizations and dealers. For further information contact Microware.

microware®

Specialists in system software for 68-family microprocessors since 1977.

OS-9 and BASIC09 are trademarks of Microware and Motorola

Microware Systems Corporation
1866 N.W. 114th Street
Des Moines, Iowa 50322
Telephone 515/224-1929
Telex 910-520-2535

Please send _____ copies of the **Basic09 Tour Guide** book at $18.95 each. Add $2.00 for UPS shipping in the U.S. or $5.00 for overseas air mail per book. Iowa residents add 4% sales ta .

Name _____

Address _____

City _____

State _____ Zip _____

☐ I have enclosed a check

☐ Charge to my bank card:
MasterCard ☐ VISA ☐
Card Number _____
Expiration _____

SOFTWARE DEVELOPERS!

YOU'VE JUST BEEN GIVEN THE BEST REASON YET
TO GET OUR 68000/UNIX® DEVELOPMENT SYSTEM

THE VAR/68K® SERIES

VK-5XW20 (List price ~~$10,100~~.) **$5,000.**
Includes: Terminal, 20 Mb hard disk, 512K RAM, 8 ports and REGULUS®

VK-5XW20T20 (List price ~~$12,900~~.) **$6,500.**
Includes: all of above, plus 20 Mb tape streamer

RESELLERS!

Even more attractive specials are available to qualified resellers!

Smoke Signal has been designing, developing and manufacturing microcomputers based on the Motorola family of processors for the past six years. The VAR/68K is the most recent addition to our family of multi-user computers.

VAR/68K is a registered trademark of Smoke Signal
REGULUS is a registered trademark of Alcyon Corp.
UNIX is a registered trademark of Bell Laboratories

Due to the extremely low prices being offered, we can only accept cash or C.O.D. orders, and we must limit purchases to one per customer. This is a limited time offer.

TO OBTAIN YOUR VAR/68K AT THESE LOW PRICES, CONTACT:

SMOKE SIGNAL
31336 VIA COLINAS
WESTLAKE VILLAGE, CA 91362
(818) 889-9340 / Telex 910-494-4965

Flex User Notes

Ronald W. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

Disk Compatibility

I just received another letter from Kent Meyers and it contains some interesting information regarding our disk format compatibility problems. Kent has found one problem and then has discovered the actual difference between the GIMIX (and Frank Hogg COCO) FLEX and the SWTPc (and Data Comp and STAR-DOS) FLEX disk formats for double sided and double density disks. I suppose the folks at GIMIX and at SWTPc have known this forever but purposely not issued changes that would make our disks compatible.

Kent outlines two problems that exist. I have known about both problems for some time, but until now have not had a real clue as to the causes. Kent explains it quite well. Let me quote his letter.

"In my previous letter I stated that the only real problem that I had with FLEX on the COCO was that some SS-50 bus systems seemed to put a large part of the track header in the disk's index hole, causing my hardware to be unable to read Sector 1 on random tracks. This became so annoying in exchanges with a friend that I finally had him send me copies of his NEWDISK utilities. Disassembly of the two revealed immediately that the problem lay not with the hardware, but with the format program itself. In single density both of them laid down a track header of exactly EIGHT bytes before the first Address Mark on the track. This is patently ridiculous. The FHL formatter and Leo Taylor's both have a header in the range of 5 to 6 times that long. In addition, no Address Index Mark is written in either single or double density.

"But the most interesting part of the disassembly was finally discovering what the actual difference was between the Gimix and SWTPC double density disk formats. I don't remember ever seeing anything about it in '68 Micro except for the bald statement that the two formats were mutually incompatible. Because of a hardware decision, all of the SWTPC single density sectors have the side byte in the id. field set to zero, and all of the double density sectors have it set to one. So running a disk test program on a SWTPC disk with a GIMIX system will show all the single density sectors on side one as bad, and all the double density sectors on side zero as bad.

"I have a complete disassembly of my current version of FLEX, so I did a little digging and discovered that the ever-flexible COCO could handle both formats (at least with FHL FLEX). The key is simply disabling the side compare during any sector read or write. This function is completely redundant on FLEX systems anyway. Each sector on a track in FLEX has its own unique number. If the side select on a drive should fail the system would still not be able to locate a sector for read or write even with the side compare not disabled. In addition, the modifications are completely transparent to the system, not affecting operation with the normal format. Formatting the disk in the SWTPC format turns out to be just as easy, requiring only two bytes to be changed.

"Here are the changes involved to FHL version 5.0:4:

```
$E11A change $86 to $84
$E1A9 change $A6 to $A4
$E204 change $86 to $84
```

"The changes to NEWDISK or NEWDISKA (FHL) are:

```
$03CD change $D6 to $12
$03CE change $24 to $12
```

"This has been tested. I formatted a double sided, double density SWTPC compatible disk on my system and it worked without a hitch on a standard SWTPC system.... One more note of caution: the modifications to FLEX must be made to allow the format programs to work."

First let me be quick to say here that having a complete disassembly of FLEX is not illegal if you disassembled it yourself, which Kent did. Few of us have the time or the patience for such a project, however. Thanks Kent for unraveling the two problems all at once. I guess what we need is a universal formatter program and enough detective work to be able to patch the other versions of FLEX as well. Several people have reported the "can't read the first sector on some tracks" problem to me, and I encountered it myself once. The disk format compatibility problem has been a headache. I've had to move a pair of disk

drives between systems and boot a different FLEX now and then to read a disk sent to me by a reader. Now, GIMIX, SWTPc, TSC, can't you all get together and issue a new common standard for disk formats? I know, TSC will say "We were first.", SWTPc will say "We just did it the way TSC did." and GIMIX will say "We did it right." In the end it will be up to Leo Taylor and Bruno Puglia to come up with a universal formatter and "hackers" (that is what Kent calls himself, and it is not meant in any critical sense of the word) like Kent Meyers to suggest patches to FLEX for each version that is floating around out there.

User's Group Has Formed

I received a letter recently from Honolulu from Dr. John Current, describing his attempt to start a FLEX user's group along the lines of the "CPMUG". Their purpose is simply to collect and distribute user contributed software in an economical manner. Dr. Current is smart enough to realize that the job will be overwhelming for volunteer help and he has formed a company whose purpose is NOT to make money, but simply to run on a non-profit basis. The company is called Aloha Software. They will have paid help, and therefore must charge enough for each disk full of software to cover cost of disks, mailing, and copying. Of course there is work involved in cataloging and testing of software.

If you are interested, write them a letter at:

Aloha Software
P.O. Box 30107
Honolulu, HI 96820

They will send you their "Interim Guidelines" and a software submission form. I've just put a couple of my utilities on a disk and sent it off to them. One good function of a user's group such as this that comes to my mind instantly, is the project mentioned above of finding the proper places to patch all the versions of FLEX for the disk compatibility problems. Maybe between all of us we could find the proper places in all the versions and arrive at a "universal disk format" and uniform handling of side and density select so that our disks would truly be interchangeable.

I suggested that it might be worthwhile to poll interested people as to what kinds of software they would be interested in obtaining. The "want list" could be published so that those of us who want to submit software could have some idea of what the others are looking for. (User's groups inevitably consist of "givers" and "takers".) In this case, those who donate software that is accepted, will be given their choice of one free disk (one volume) of software in exchange for their contribution.

Sorts on the Half Shell

Catchy title, don't you think? What brought this subject up was a problem I had the other day of sorting some data in a reasonable time. I think probably all of you have programmed a bubble sort at one time or another. The basic idea is that you can sort a list (an array) of items in order of increasing value (alphabetically if string comparisons are used, numerically if number comparisons are used), by starting at the top of the list and comparing the first two items. If the first is larger than the second (for sorting in ascending order), swap them and then compare the second and third items, etc. The process is repeated until the list has been "passed through" once with no swaps having taken place, at which point the list is sorted. If you think about it a little, the effect of a pass is to "sweep" the largest item to the bottom of the list. On the second pass, we can stop one item short of the bottom, etc. That little refinement cuts the sorting time in half.

The problem with the bubble sort is that it is what is called an n squared process. That is, the time it takes to sort a list is proportional to the square of the number of items in the list. Double the number of items, and it takes four times as long to run. There are considerably more complex sorting methods that run much faster. The next "least complex" one that runs considerably better than the bubble sort is known as the Shell Metzner sort after the people who "invented" it. This sort starts by comparing items that are far apart in the list. It swaps items for any case in which the larger item is above the smaller in the comparison. The reasoning was that the largest item would quickly find its way to the bottom, and the smallest to the top. It works rather astonishingly better than the bubble sort as the list gets large.

One of the problems with the Shell-

Metzner sort (usually called the "Shell Sort" by way of abbreviation and not to omit credit to Metzner) is that when it swaps a pair of items, it saves the current item pointers and backs up one "sort interval" if the first pointer doesn't back up beyond the start of the list, and it again compares and swaps if necessary. While this extra step surely must reduce the number of passes through the list required, it also adds to the complexity of the program. Normally one makes passes through the list with the initial sorting interval until no swaps are made, and then divides the interval in half, repeating until the interval reaches zero, at which point the sort must be done. (Note that the sorting interval is an integer value so that 3/2 = 1 and 1/2 = 0. If you must use a "real" number, than you must use the INT function after performing the division.)

I decided to implement only the first half of the sort (hence the title Half Shell). The length of the list to be sorted is divided by 2 and the value obtained is used as the first sort interval. When no swaps are made on a pass, the interval is divided by two and the process repeated. It is informative to print the sort interval and the number of passes (indicated by asterisks) to the terminal as the sort progresses. Adding the sorting interval complication to the bubble sort is easy, and adds only a few lines to the sort portion of the program. I wrote the programs in BASIC and timed them for lists of 200, 400, and 800 items. Later I decided to see what would happen if the sort interval were reduced by less than a factor of 2, and changed the reduction factor to 0.7 (easy in BASIC but perhaps more complicated in other languages). The resulting program was again a significant improvement over the previous attempt.

After obtaining timing information, I added code to the programs to count the number of comparisons and number of swaps for the sorts, and obtained that information by rerunning the programs. Counting comparisons, particularly for the bubble sort, slows the program down considerably. (I was able to use integer variables to hold the comparison count for the other programs, but had to use a floating point variable in the bubble version since even at 400 items the number of comparisons greatly overflowed integer arithmetic. The results were as follows:

Sort		Time	Swaps	Compares
Bubble	200	87	10396	19795
	400	338	40008	78474
	800	1313	151312	317455
H. Sh.	200	18	937	6603
	400	49	2322	18744
	800	124	6233	47000
(approx.)				
0.7	200	17	585	6473
	400	40	1383	16032
	800	104	3504	43000
(approx.)				

The compare count for the partial Shell sorts also overflowed the integer arithmetic, but didn't wrap around all the way so I was able to figure them out.

Sorting programs are somewhat sensitive to the data that is input, so I seeded the random number generator in TSC Extended BASIC by the statement X= RND(-700) in each case so the three sort programs would work on the same data and I could make small changes and have meaningful comparisons. The variable SW% is used to keep track of whether there were any swaps made on a given pass. It is set to zero at the beginning of the pass, and set to 1 if the swap subroutine is used once or more during the pass. One thing that strikes me as inefficient about the Shell sort is that one pass is made at each sorting interval during which no sorting takes place. That is, the pass in which there are no swaps, which signals decreasing the sorting interval to its next value. It would seem in spite of this, that the partial Shell sort would be a much better choice than the Bubble sort for lists of any length over a few dozen items.

I have found some of my old test results on various sorts, and I find that the full Shell sort reduces the number of comparisons for a list of 1000 items to about 1/3 of those for the partial Shell sorts. I have run some of the data for 1000 items, and I find the following approximate times to hold:

Bubble	3200 seconds
H. Shell	155
0.7 Sh.	122
Shell	94
Quick	46

All times reported above are for a 2 MHz 6809 system running in TSC Extended BASIC. Some of the times in the table immediately above are extrapolated from other data. You can see clearly that the first major improvement is made by using "anything but the Bubble sort". The full Shell sort runs in about 2/3 the time of the simplified version. Quicksort is a considerably more complex sorting method that uses stacks to hold partially sorted fragments of the list (or pointers to them). However, as you can see, it is about three times as fast as the "Half Shell" version and twice as fast as the full Shell-Metzner version. I note here that the 0.7 ratio simplified Shell sort made just over half as many swaps as the full Shell sort for 1000 items. It made 3.5 times as many comparisons, and therefore ran more slowly. In some applications and with some languages comparisons are very fast and swaps are very slow, so that the "0.7 Half Shell" version might look more favorable.

If some of you express further interest in sorting methods, I'll include BASIC implementations of the full Shell sort and the Quicksort in a later column, along with results on the same lists of random numbers.

Another O.S.

Well, the company has just bought an IBM "look alike" and some software for P.C. board layout and artwork generation (Wintek's SmArtwork software). We've just gotten the system up and running, and I will therefore be learning about MS DOS. I suspect a few of you might be interested in my impressions, so I will present them here briefly after I've had a chance to try the system out for a while. First impressions of the system are that it is simple to operate. The software has just about the right level of "automatic". That is, it is not too manual to be useful nor too automatic to be understood without a very long learning time.

K-BASIC Again

Having some little interest in the first BASIC compiler to be usable to compile TSC Extended BASIC programs rather directly, I receive almost weekly progress reports from Frank Hoffman at Lloyd I/O in Portland OR. Frank tells me that he has eliminated the requirement of no spaces in an arithmetic expression, gotten random files implemented, and pretty well debugged the full implementation of PRINT USING. The compiler has gotten easier to use and considerably more efficient than the first versions in the process. Frank is planning to offer the user a choice of two floating point math packages. Presently he has a 15 digit BCD math package that is superb for financial calculations, but a little on the slow side for serious scientific number crunching applications. He plans to offer a 9 digit binary math package that should run considerably faster and have more than sufficient accuracy for scientific applications.

- - -

Editor's Note: We here at CPI feel that K-BASIC could be a milestone for our group. However, there are a couple of **small** items that **MUST** be taken of first. Mainly free of 'bugs' and capable of compiling rather large programs with moderate amounts of RAM available.

There is a wealth of software running under the TSC XBASIC format. The primary problem was after loading in the BASIC program it's self, then the BASIC source program, then reserving RAM space for the stack, tables, etc., not much space left for 'scratch' RAM. Therefore, the application program is limited in many ways due to time consuming system functions, such as chaining, loading additional source, etc. This heavy operational overhead is vastly reduced if the code is reduced to binary object only!

Thus a BASIC compiler - source to pure object - is the perfect solution. Provided it compiled good code. Right off the bat the doing-away-with of the BASIC program saves about 79 sectors, and the source compiled is a '.CMD' file, running like any other command file under FLEX or OS-9.

Speaking of OS-9, K-BASIC, with a FLEX to OS-9 porting program (S.E. Media, 'OF') will allow a FLEX XBASIC program to be compiled and run in the OS-9 system. This ports over most of that FLEX applications base that before could not be run under OS-9. K-BASIC solves a lot of problems for those OS-9 users needing business and other applications software.

I also am following this development with **MUCH** interest, IT MEANS THAT ALL THOSE OLDER SWTPC, GIMIX, SSB, HELIX and any other system capable of running FLEX and/or OS-9 has a whole new ballgame to play - lots more

useable RAM, faster execution and all the other better things brought on by no intermediate, inefficient, cumbersome interpreter.

DMW

- - -

```
10 REM BUBBLE SORT
12 REM NOTE THAT THIS PROGRAM CONTAINS THE INSTRUCTION:
14 REM   EXEC, "TIME" IN TWO PLACES.  THIS CAUSES MY SYSTEM
16 REM   TO PRINT THE CURRENT TIME ON THE SCREEN.  IF YOU HAVE
18 REM   A CLOCK CHIP YOU CAN SUBSTITUTE YOUR INSTRUCTION,
19 REM   OTHER WISE DELETE THESE TWO LINES OF EACH PROGRAM.
20 INPUT 'HOW MANY NUMBERS',MX%
30 DIM AR%(MX%)
35 X=RND(-700): REM SEED GENERATOR
40 FOR N%=1 TO MX%
50 AR%(N%)=INT(RND(0)*MX%)
60 NEXT N%
70 PRINT 'SORTING' : REM START SORT
75 EXEC,"TIME"
80 L%=MX%-1
90 SW% = 0
100 FOR N%=1 TO L%
110 IF AR%(N%) > AR%(N%+1) THEN GOSUB 200 : REM SWAP
120 NEXT N%
130 L%=L%-1
140 IF SW%<>0 THEN 90
145 EXEC,"TIME"
147 INPUT"SORTED LIST",R$ : IF R$<>'Y' THEN 175
150 FOR N%=1 TO MX%
160 PRINT AR%(N%),
170 NEXT N%
175 PRINT
180 END
190 REM SWAP SUBROUTINE
200 T%=AR%(N%)
210 AR%(N%)=AR%(N%+1)
220 AR%(N%+1)=T%
230 SW%=1
240 RETURN

10 REM PARTIAL SHELL SORT
20 INPUT 'HOW MANY NUMBERS',MX%
30 DIM AR%(MX%)
35 X=RND(-700):REM SEED RANDOM
40 FOR N%=1 TO MX%
50 AR%(N%)=INT(RND(0)*MX%)
60 NEXT N%
70 PRINT'SORTING' : REM START SORT
80 EXEC,"TIME"
90 L%=MX%
100 I%=L%/2
110 PRINT I%;
120 SW% = 0
130 FOR N%=1 TO L%-I%
140 IF AR%(N%)>AR%(N%+I%) THEN GOSUB 290
150 NEXT N%
160 PRINT "*";
170 IF SW%<>0 THEN 120
180 I%=I%/2
190 PRINT:PRINT I%;
200 IF I%<>0 THEN 120
210 EXEC,"TIME"
220 PRINT : INPUT "SORTED LIST";R$
230 IF R$='N' THEN 265
240 FOR N%=1 TO MX%
250 PRINT AR%(N%),
260 NEXT N%
265 PRINT
270 END
280 REM SWAP SUBROUTINE
290 T%=AR%(N%)
300 AR%(N%)=AR%(N%+I%)
310 AR%(N%+I%)=T%
320 SW%=1
330 RETURN

10 REM PARTIAL SHELL SORT
20 INPUT 'HOW MANY NUMBERS',MX%
30 DIM AR%(MX%)
35 X=RND(-700):REM SEED RANDOM
40 FOR N%=1 TO MX%
50 AR%(N%)=INT(RND(0)*MX%)
60 NEXT N%
70 PRINT'SORTING' : REM START SORT
80 EXEC,"TIME"
90 L%=MX%
100 I%=L%*0.7
110 PRINT I%;
120 SW% = 0
130 FOR N%=1 TO L%-I%
140 IF AR%(N%)>AR%(N%+I%) THEN GOSUB 290
150 NEXT N%
160 PRINT "*";
170 IF SW%<>0 THEN 120
180 I%=I%*0.7
190 PRINT:PRINT I%;
200 IF I%<>0 THEN 120
210 EXEC,"TIME"
220 PRINT : INPUT "SORTED LIST";R$
230 IF R$='N' THEN 265
240 FOR N%=1 TO MX%
250 PRINT AR%(N%),
260 NEXT N%
265 PRINT
270 END
280 REM SWAP SUBROUTINE
290 T%=AR%(N%)
300 AR%(N%)=AR%(N%+I%)
310 AR%(N%+I%)=T%
320 SW%=1
330 RETURN
```

OS9 USER NOTES

By: Peter Dibble
517 Goler House
Rochester, NY 14620

64 Thousand Bytes Inside the Computer

Before I get into the meat of this column I want to mention the name of the file this text is stored in, Column24. It's hard to believe I've been at this that long!

The folks at '68 Micro Journal sent me a letter from a CoCo owner who had upgraded to version 1.1 of Radio Shack OS-9. For some reason, when he used OS9GEN to put the OS9Boot file on a new disk, the new boot file booted up a system that used more memory than the master disk's. He had the same problem with version 1.0, but it was worse with 1.1. There wasn't enough free memory left to run C. That's not too surprising. The C compiler uses a LOT of memory. It took some powerful shoehorning to wedge it into the memory available on a CoCo. Take a little memory away and you just squeeze it out.

I tried to solve the problem for him, but I couldn't get my system to reproduce it. He sent me convincing documentation; I'm sure he wasn't imagining things. I was interested enough that I had it in mind as a good subject for this month's column. Stubbornly, I'm going to write about it anyhow.

The obvious tools to attack a memory allocation problem with are mdir, mfree, and procs. Each of these commands give some information about memory usage. No commands give you all the gossip about your memory without some work on your part. Even with work, debug is your best tool. If you're ready to write a short program, a program that copies most of memory to disk is useful. With everything in a file you can use dump to get a hard copy. I'm old-fashion enough to like paper I can mark on when I'm tearing a system apart. A warning: If you dump all your system's memory (like I did) we're talking about a big pile of paper.

The place to start finding out where your memory has gone is the system direct page. This is located at address $0000. When I dumped my CoCo's memory it looked like this:

```
Addr  0 1  2 3  4 5  6 7  8 9  A B  C D  E F
0000  2600 2200 0000 0000 004F 0201 A901 A47F
0010  2329 3C00 0000 007F 2A26 0129 3329 4129
0020  0200 0220 BE00 0300 0400 FD46 F1CC F2C2
0030  F1D8 C26F F1DD F1D8 F215 D02B F1EA F215
0040  F29B F2C2 0292 0222 02EF 00EF C0EF 80EF
0050  4000 0054 0A15 1304 2938 3C06 0000 0000
0060  BD6C BD00 BC00 BE00 EE32 00E4 46B9 00EF
0070  0055 0074 127F FF03 B7FF DF7E F00C 0000
0080  00F2 2700 0000 0000 0000 0000 0000 0000
```

If you want to follow this tour in your own machine (Level One only for this month, sorry) note the numbers at:

$0020-0023	($0200 $0220 in this dump)
$0049-004A	($EF00 in this dump)
$0060-0061	($BD6C in this dump)
$0064-0065	($8C00 in this dump)

These numbers are the addresses of the free memory bit map, the process descriptor table, the device table, and the path descriptor table respectively. Each of these tables contains information about memory. All but the free memory bit map contain memory that's hard to get without digging.

The free memory bit map looked like this:

```
Addr  0 1  2 3  4 5  6 7  8 9  A B  C D  E F
0200  FF80 0000 0000 0000 0000 0000 0000 000F
0210  FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
```

The system direct page gave both the beginning and the end of the map. Each bit that's off indicates a free page. Thirty-two bytes at eight bits each gives a map of 256 pages. At 256 bytes per page that's 64K. In this map the first nine pages are allocated as are the last 132. That left me with 115 contiguous free pages.

The process descriptor table can tell you about the data memory allocated for each process. You can find out how much memory is allocated to each process from the procs command. The process descriptors can tell you the address of each process's memory as well as its size.

This table, like most others in OS-9 Level One, is managed with the F$All64 family of system calls. The first 64 bytes of the first 256-byte block are a list of the high-order bytes of the

addresses of other blocks in the table. In this case, there is only one 256-byte block so only the first byte in is non-zero. The next 64 bytes (starting at $EF40) contain the first process descriptor. Check in your DEFS directory for information about all the fields in it. The page number of the beginning of a process's memory can be found seven bytes from the beginning of the descriptor. For process one, that's at $EF47. The value is $04 so process one's data memory starts at $0400. The next byte in the process descriptor gives the number of pages allocated, in this case, one. Process two has three pages allocated to it starting at $0500.

Addr	0 1	2 3	4 5	6 7	B 9	A B	C D	E F
EF00	EF00	0000	0000	0000	0000	0000	0000	0000
EF10	0000	0000	0000	0000	0000	0000	0000	0000
EF20	0000	0000	0000	0000	0000	0000	0000	0000
EF30	0000	0000	0000	0000	0000	0000	0000	0000
EF40	0100	0002	04E2	0004	0100	0000	00B0	0000
EF50	0000	E2AC	F29B	F29B	F29B	BD6C	0000	0002
EF60	B06C	0000	003C	0101	0100	0000	0000	0000
EF70	0000	0000	0000	00E4	1E04	0000	0000	0000
EFB0	0201	0003	0700	0005	0300	0000	00B0	0000
EF90	0000	E4CD	F29B	F29B	F29B	B06C	0000	0002
EFA0	B06C	0000	0020	0102	0101	0000	0000	0000
EFB0	0000	0000	0000	00E5	0705	0000	0000	0000
EFC0	0302	0000	0BF3	000B	0100	0000	01A0	EF80
EFD0	0000	7C00	F29B	F29B	F29B	BD6C	0000	0002
EFE0	B06C	0000	0020	0102	0100	0000	0000	0000
EFF0	0000	0000	0000	0000	0000	0000	0000	0000

Memory that mysteriously vanishes has generally been used for some kind of I/O buffer. The last two control blocks I'll show you point to I/O buffers.

When a device is opened for the first time some memory called device static storage is allocated for it. That memory will generally stay allocated until you reboot. The address of each device's device static storage can be found in the device table. It should look something like:

Addr	0 1	2 3	4 5	6 7	B 9	A B	C D	E F
B060	0000	0000	0000	0000	0000	0000	BE00	BB00
B070	C10B	D1A1	03C1	C7B9	00CA	B2DE	9302	0000

This device table starts at $BD6C (remember the pointers from the system direct page). Each entry is nine bytes long. The static storage address is two bytes in. For the first device in the table that's $BB00, for the second, $B900. To find out the size of these areas check the device descriptor module pointed to by the sixth and seventh bytes in each entry.

If some device static storage gets allocated in the middle of memory you'll wish it hadn't. If the device descriptor isn't in your boot, try unlinking it. You may be able to remove the device descriptor and its static storage from memory.

You can prevent the problem by opening any devices you intend to use before you start running programs. You don't need to do anything with them; just open and close.

If you want to be elegant, write a program that just does an I$ATTACH for the device; that's the part of open that allocates the storage and puts a device in the device table.

The last table in this tour is the path descriptor table. These are system paths not user path numbers. You'll see their numbers in the "paths" area of each process descriptor. Don't think that these path numbers are arranged standard in, standard out, ... The table looks like:

Addr	0 1	2 3	4 5	6 7	B 9	A B	C 0	E F
BC00	BC00	0000	0000	0000	0000	0000	0000	0000
BC10	0000	0000	0000	0000	0000	0000	0000	0000
BC20	0000	0000	0000	0000	0000	0000	0000	0000
BC30	0000	0000	0000	0000	0000	0000	0000	0000
BC40	0103	0BB0	7500	07EB	BB00	B075	00FF	FF00
BC50	0000	0000	0000	0000	0000	0000	0000	0000
BC60	0001	0100	0101	0001	100B	1B00	1B04	0117
BC70	0305	0B07	0000	002A	0000	00B0	7500	0000
BC80	0202	02B0	6C03	0BF3	B500	0000	00BC	0000
BC90	00FF	FF00	0000	0000	A500	00C7	C31E	BB0F
BCA0	0100	0020	0100	2301	0000	1200	1204	0B00
BCB0	0000	000B	0000	1100	0002	0000	0060	BD6C
BCC0	0001	00B0	6C00	09DF	7B00	0200	0013	0000
BCD0	0276	0000	0000	0000	0000	0276	0000	BB0F
BCE0	0100	0020	0100	2301	0000	1200	1204	0B00
BCF0	0000	0000	0000	0000	0000	0000	0000	B06C

It's another table managed by the F$A1164 family. The first path descriptor starts at $BC40. At eight and ten bytes from the beginning of each descriptor are addresses of allocated memory. Bytes eight and nine are the address of the buffer for the path

(path one has a buffer at $B800, path two at $B500). Bytes ten and eleven are the address of the file manager's static storage (if any). The first path has some allocated at $BD75. It appears that the file manager for the second path (SCF) doesn't need any static storage.

You can poke around in these control blocks with debug. If crashes disturb you don't change anything in these tables. OS-9 will notice a change and expire. plunk.

I copied memory to disk with a dumb program that pointed X at the memory I wanted to dump, put the length in Y and wrote it to standard output. I used X=0 and Y=$FFFF to dump everything. You'll probably be clever and only dump what you need.

First Aid

If you always need a little more memory the best solution is to move to Level Two (even better, OS-9 68K). You'll still have memory problems, but they will be more tractable. While you're saving up, try this trick for temporary help: build a new smaller boot file. The boot file must contain IOMan, RBF, SCF, CCIO, CCDisk, D0, TERM, Shell, Clock, and SYSGO. The other modules are optional. If you have two disk drives, you need D0 and D1, D2 and D3 are wasted space. If you don't have a printer, terminal, or modem; PRINTER, RS232, P and T1 are useless.

Don't remove the pipe modules except in desperation. Many people don't use pipes. If you don't use them and don't intend to, they can go. All the OS-9 programs from Microware (or Tandy) that I know of can be used without pipes. You can certainly compile C programs and run Basic09 without pipes. If you take all those modules out of the boot file, you will have decreased the its size by 1225 bytes. That translates into at least four more pages of memory for your programs

A Preview

I have a pile of programs from the JBM Group for review. My first impression of them is that programmers who want to create business software would be fools not to look at it. Other groups will find at least some of JBM's stuff interesting. More on this software over the next months.

"C" User Notes

Edgar M. (Bud) Pass, Ph.D.
1454 Latta Lane
Conyers, GA 30207

This month's column discusses several problems in the C compilers available on the 6809, provides another list of references on the C language, and comments on the availability of the Apple MacIntosh C compiler.

Next month's column will begin a multi-column tutorial on the C language, due to popular demand. It is based upon short C courses I have presented and upon tutorials I have developed in the past, in addition to some new material developed for it.

C PROBLEMS

My primary intention in presenting these problems is not to criticize the developers, but to publicize the problems so that other users will not fall into the same traps. Of course, I would be pleased if the developers of the various packages would fix the problems documented in this column. If a user encounters a problem not previously documented here, I would appreciate receiving a copy of a test program, on disk if possible.

In attempting to complete a project involving the use of the C language, I recently encountered several problems with the Microware/Tandy C compiler and the Windrush C compiler. Both of these compilers are based upon compilers developed by James McCosh. I also had a problem with the Introl C compiler, and have a problem with the Dyna-C compiler to report.

The most immediate problem, which threatened the viability of the entire project, was that the Microware C compiler is so large that it cannot compile a very large C module on a standard level 1 system. Even generating a bare-bones OS/9 system did not quite provide enough free memory to use the C compiler to compile several of the program modules.

I solved this problem, as far as the Microware C compiler is concerned, by using a Color Computer, which provides more free memory than a standard level 1 system. It is also considerably slower

than my other system, requiring about one hour to compile a program which the other system will compile in about twenty minutes (in FLEX).

The Microware/Tandy assembler which comes with the C compiler can handle only about 600 labels for each generated assembler module on an OS/9 level 1 system. Since each string in an initializer requires a label, this restricts the size of initialized tables, even if each table is compiled separately from all other code. Remember that the McCosh family of C compilers always generates string pointers, not the string contents, in initialized tables, and this process requires a label for each string in the tables.

The Introl C compiler has somewhat similar size problems on an OS/9 level 1 system. It has a command-line option which reduces the memory requirements by eliminating the use of initializers. Since most large C modules with which I work have initializers, this is only marginally acceptable.

The obvious solution to this problem is to increase the number of C compiler and assembler phases, in order to reduce the size of each phase. Since this would be such a major change, I do not anticipate actually seeing it, however.

A serious bug which I encountered in the Microware C compiler relates to strings containing an octal constant generating a characters with the high-order bit set. The compilation seems to proceed normally, but the linker generates a large number of undefined variable names of the form "_nnn".

When I initially talked to Kim Kempf at Microware about this problem, he thought it might be due to the size of the module. Eventually, I sent a copy of the C program to him, and continued to attempt to get the program to compile and link properly. I determined that the bug was as described above at about the same time that Kim did. The assembler was quitting upon encountering an octal 377 (hex FF) value in the intermediate assembler-language file. The compiler attempted to generate FCB constants for characters not representable in FCC strings, but neglected to check for characters with values larger than hex 7E. The Windrush C compiler avoids this problem by coding all strings as the equivalent FCB decimal constants.

Kim was very helpful to me in this matter, and has corrected the bug in the current release of the Microware C compiler. However, older versions of the Microware and Tandy C compilers still have the bug, so octal constants in strings must be used carefully with these compilers.

The use of both the Microware and Tandy C compilers is made somewhat inconvenient by their insistence on correct case in some command-line options. This is especially inconvenient on the Color Computer, in which the shell converts lower case to upper case on the command line, by default. The manuals compound the inconsistency by documenting the use of all options in upper case, whereas the compilers insist on the use of lower case for some of the options. Hopefully, the compiler, not the manuals, will be corrected to eliminate this inconvenience.

I also had several problems with the Windrush C compiler, which runs under Flex.

One minor problem concerns the use of command-line arguments in the compiled program. The Windrush C manual states on page 6 that arguments on the command line are normally delimited by spaces or commas, but may enclosed be in double quotes, in which case they may contain spaces or commas. This is not true, as may be verified by running the "echo" program on page 111 of the K and R book. Double quotes have no special meaning in Windrush C. No other known Full C compiler on the 6809 has this problem.

A more serious situation exists in the Windrush C compiler with respect to reading and writing arbitrary data values from and to disk files. Although most other implementations treat "open" and "creat" as low-level functions, Windrush C treats them almost the same as "fopen", without the binary option. Thus, "fopen" must be used, with the binary option, when reading and writing disk data files, since "open" and "creat" ignore the mode parameter, which is normally used to indicate the difference between text and executable files.

Note that the only safe, general manner in which to read and write arbitrary data values in Windrush C is with the "fread", "fwrite", "putc", "putchar", and "putw" functions. None of the input functions based upon "getc" may be used, since "getc" cannot distinguish between reading a character with value hex FF and encountering end-of-file. None of the string I/O functions may be used, since they normally assume null-terminated strings. None of the standard C I/O functions distinguish between end-of-file and logical or physical I/O errors, such as insufficient disk space. These last three situations describe a problem with the definition of the C language, not with the Windrush C compiler alone.

The Windrush C manual is somewhat misleading in its heading of the "fread" and "fwrite" functions as "read/write binary data", in that they do so only if the file was opened with the binary option of the "fopen" function.

The Windrush C compiler does not check file names for invalid characters. This can lead to some rather strange circumstances in which the output file of a program overwrites the input file, assuming they differ only in the suffix. This is because only the part of the file name up to but not including the first invalid character is used, possibly causing the suffix to be ignored.

I have had one report of a verified bug in Dyna-C. If a function has no code-generating statements in its body (between "{" and "}" characters), a call to that function falls through to the next function, or to whatever code follows.

C REFERENCES

I have had several requests for more information on references to learning and using the C language. Following is a list of the titles, authors, and suggested retail prices of all of the major books on the C language now known. If anyone knows of other references or has comments on any of these, please let me know.

A Book on C, Pohl, $23.50
C Notes, Zahn, $21.50
The C Programming Handbook, Plum, $21.50
The C Programming Tutor, Plum, $19.50
The C Primer, Hancock, $18.50
C Programmer's Library, Purdum, $21.50
C Programming Guide, Purdum, $19.50
C Programming Guidelines, Plum, $27.50
The C Programming Language,
    Kernighan and Plauger, $21.50
The C Puzzle Book, Feuer, $16.50
C User's Handbook, Weber, $16.50
Learning to Program in C, Plum, $26.50
Programming in C, Kochan, $20.50
Programming in C for Micro Users,
    Traister, $19.50
Small-C Handbook, Hendrix, $16.50

Dr. Dobb's Journal has published a significant number of lengthy articles on C programming and applications over the past few years. Bound reference volumes and back issues are available.

'68' Micro Journal has published a number of articles on C programming and applications, especially as they relate to the 6809 implementations of the C language.

In addition, I have a small number of copies of my tutorial on C which I will send to anyone in the US or Canada for $2 for shipping.

C FOR THE MACINTOSH

In April 1984, Apple announced that it would produce a Full C compiler for the MacIntosh which would have access to all facilities of the machine, would require two disk drives, and would be available in December 1984. I inquired about the availability of it recently, since I wanted to use it for developing MacIntosh software, and received a telephone call from an Apple representative. She told me that Apple had significantly reduced the priority of the development of the C language for the MacIntosh in favor of the development of their own version of a BASIC interpreter. She could give me no date for when an Apple C compiler might be available. In fact, she suggested that I investigate one of the other C compilers already implemented on the MacIntosh by third-party developers. These range from Small-C ported implementations to complete C development package systems. I will report on them as I receive more information.

I also inquired about Apple's assembler-language development system. It is available, but it requires two

MacIntoshes (connected together by a common external bus) or a Lisa and a MacIntosh in order to operate. However, several third-party developers have announced similar development systems which will run on one 128K MacIntosh with one disk drive.

C PROBLEM

Last month's problem was to write a C version of the BASIC "instr" function. This may be accomplished as follows:

```
/*
    instr (fir, pat, src) searches "src",
    searching for the first instance of
    "pat" in "str" starting with position
    "fir" (base 1), returning the
    position of the first character of
    the matching string or zero;
    both strings are null-delimited.
*/

instr (fir, pat, src)
int fir;
char *pat, *src;
{
    char x, *p, *t, *s = (src + fir - 1);

    while (*(t = s++))
    {
        for (p = pat;
            (((x = *p++) == *t++) && x); );
        if (!x)
            return (s - src);
    }
    return (0);
}
```

Write a complete program which replaces one string of characters with another in text copied from one file to another. Take the command-line problem reported in Windrush C into account, if you can. A sample command line for such a program might be as follows:

```
replace file1 file2 string1 string2
```

EXAMPLE C PROGRAM

Following is this month's example C program; it removes tabs and other undesirable control characters from a text file. This could be useful when attempting to use files transmitted from a CP/M system. Since Flex uses the tab character as a leadin for space compression, tabs in Flex files are particularly undesirable, but they are also undesirable in OS/9 files. The program actually replaces each tab with a space, rather than removing it entirely.

```
/*
    untab.c - removes tabs and other
        controls from text file
*/

#include "stdio.h"
#include "ctype.h"

#define MODE "rb"
                /* MODE "rb" for FLEX
                   MODE "r" for OS/9  */

main(argc,argv)
int argc;
char **argv;
{
    char *input,*output;
    int c;

    input = stdin;
    output = stdout;
    if (argc > 1)
    {
        if ((input = fopen(*++argv, MODE))
            == NULL)
        {
            fputs ("can't open input\n",
                stderr);
            exit (1);
        }
    }
    if (argc > 2)
    {
        if ((output = fopen(*++argv, "w"))
            == NULL)
        {
            fputs ("can't open output\n",
                stderr);
            exit (1);
        }
    }
```

```
    while ((c = getc (input)) != EOF)
        if ((c < 0x80) && ((c > 0x1f) ||
            (c == 0x0d)))
            putc (c, output);
        else
        if (c == 0x09)
            putc (0x20, output);
    exit (0);
}
```

Editor's Note: It might be fair to point out that the Microware OS-9 version of the McCosh C compiler is supported solely by Microware and there will be some differences between it and the several other McCosh C compilers.

In the long haul I believe that will be a better situation for OS-9 users of this version of C. OS-9 is growing in use to the point that it's version of C will be widely distributed worldwide. And used on many different types of systems, from various manufacturers. The resources necessary to maintain such a network seem to be in place, however it will require a reasonable length of time to get things running smoothly.

All of the compiler developers, I have talked to are more than willing to set things right, it may take some time but I have faith in our suppliers. They have too much at stake. I know Some fine software developers (and hardware) who let their S50 Bus commitment go slack in order to attempt a try at other non-S50 markets. And lost all around. A reputation of support is one of the MOST IMPORTANT assets a manufacturer must develop and maintain! We have never had the most of anything, but in some areas we sure have had as good or BETTER!

You should see some of the 'better' C compilers running on some other systems (non-68XX). Go down to your local 'other' store and ask to see their C compiler documentation. It may look prettier but the scope of many 'other' C compiles borders on junk. I guess that is the proof of the pudding - or something.

DMW

\- - -

68000 USER NOTES

Phillip Lucido
2320 Saratoga Drive
Sharpville, PA 16150

68020

As mentioned last month, I have acquired Motorola's user's manual for the 68020, the **MC68020 32-Bit Microprocessor User's Manual**, Motorola part number MC68020UM(ADI). The book is actually published by Prentice-Hall, ISBN 0-13-541418-0. That should enable you to find it, should you be so inclined. This month's column is mainly a distillation of this book, covering the new features in the 68020.

Before I get started, though, I have a few thoughts on the whole idea of 32 bit microprocessors and the advancing state of the art, prompted in part by Ron Anderson's column in the February 1985 issue. According to his column, he has yet to come across an application where a 68000 is really preferable to a 6809. I wholeheartedly agree, since very few applications need huge amounts of RAM or great speed. So why do I use a 68000, and why am I now studying the 68020?

I suppose that I am guilty of being a gadget freak when it comes to computers, especially in regards to the software. I am a systems programmer, and enjoy playing around with large programs, such as editors or compilers. Most of my interest in new microprocessors, then, derives from their use in software development systems and general purpose computers. This is quite divorced from the use of microprocessors in dedicated designs, such as system controllers or limited purpose computers like word processors.

Still, why should I need a 68000 at the heart of a general purpose computer? Isn't a 6809 powerful enough? In most respects, a 6809 provides all the power I need. After all, the microprocessor used in a computer is only one factor in its speed. A much better way to improve a computer system might be to install a hard disk drive or more RAM memory. For instance, at the OS-9 seminar, I saw a 68000 computer running OS-9/68K which was performing C compilations in something like 10 or 15 seconds. My current system takes more like 60 to 90 seconds, even with a hard disk. At first, I thought this speed was due to the power of the 68000. While the 68000 probably helped, the speed was really due to the 2 megabytes of

RAM installed in the machine! All of the program phases in the C compiler were preloaded into memory, and all files, including temporaries, used a RAM disk, so that an entire compilation could be performed without once using a floppy disk or hard disk. The same trick could have been performed using a 6809, though maximum memory would likely have been 1 megabyte instead.

I still haven't said why a 68000 is a good idea. There are some valid reasons. Even using a 68008, it is possible to write programs which run faster than the same program with a 6809, because of the greater number of registers and the more powerful instructions. The removal of the 64K limit on program size is very important, allowing huge, complex programs to be written without worrying about most memory limits. This alone means that programs will be developed for the 68000 which will be difficult or impossible to port back to the 6809.

Unfortunately, the overriding reason for my having a 68000 may simply be that 'gadget freak' tendency I mentioned before. There is always the inclination to have the latest and the best equipment possible. This is not entirely bad, as I am sure that readers of this magazine understand the 'toy' aspect of having a computer. After all, they are fun, and it's always nicest to have the newest, shiniest toy on the block. So, in that spirit, let me get on with talking about the 68020, the newest of the new toys around. While I am unlikely to have one anytime soon, it is still useful to see what the future holds.

New and Improved! 68020 Instructions

As mentioned last month, there are two major groups of new instructions, bit field operations and coprocessor operations. The bit fields operations deal with a variable length bit field, from 1 to 32 bits long, without respect to byte or word boundaries. A bit field is specified with three numbers: the base address, the bit offset, and the bit field width. The base address is specified using the normal effective address calculations applicable to most 68020 instructions. The offset is either an immediate value, from 0 to 31, or a value from -2^31 to (2^31)-1, held in a data register. Finally, the field width is either an immediate value or a value in a data register. In either case the field width is from 1 to 32 bits.

There are 8 different bit field opcodes. BFINS is used to transfer a bit field from a data register to its position in memory. BFEXTS and BFEXTU are used to retrieve a bit field from memory, placing the result right justified in a data register. BFEXTS performs sign extension on the field, while BFEXTU performs zero extension. BFCLR clears a field to all 0's, while BFSET sets a field to all 1's. BFCHG complements each bit within a field. BFTST checks if a bit field is all 0's, as well as if the most significant bit in the field is a 1. Finally, BFFFO scans through a bit field, looking for the first 1 bit. If a 1 bit is found, then the offset of that bit from the base address is stored in a data register. If no bits in the field are set, then the value of the field offset plus the field width is stored in the data register.

The coprocessor instructions are too complicated to discuss in any detail here. Basically, though, the coprocessor interface depends upon a satellite chip which follows a very strict protocol for communicating with the 68020. The 68020, in return, implements 7 different opcodes dedicated to the use of coprocessors. The cpGEN opcode is the general function code, used to request most operations from the coprocessor. The actual format of this instruction is determined by the coprocessor, with extension words specifying the actual operation to be performed. There are 4 conditional opcodes used to test conditions within the coprocessor. These are cpBcc, cpDBcc, cpScc, and cpTRAPcc, each of which functions analogously to the normal 68020 instructions Bcc, DBcc, Scc, and TRAPcc. The meaning of the condition codes depends entirely upon the coprocessor. These are not necessarily the same condition codes which can be internally tested in the 68020. Finally, there are 2 opcodes, cpSAVE and cpRESTORE, which are used to save the internal state of a coprocessor in memory and restore that state at a later time.

There are new forms of the multiply opcodes in the 68020. The normal 68000 MULS/MULU opcodes performed 16 bit by 16 bit multiplication, yielding a 32 bit result. These are now the MULS.W/MULU.W opcodes. The 68020 adds two new long forms (e.g. MULS.L). One of these performs 32 bit by 32 bit multiplication, giving a 32 bit result with possible overflow. The other does 32 x 32 bit, with a 64 bit result. The result is

left in two different data registers, which are not necessarily adjacent. For instance, MULU.L (A4),D0:D3 would multiply the 32 bit value at (A4) by the 32 bit value in D3, with the high order 32 bits of the result stored in D0 and the low order 32 bits in D3.

In the same way, there are new forms of the divide opcodes. In the 68000, the DIVS/DIVU opcodes performed 32 bit by 16 bit division, giving a 16 bit remainder and 16 bit quotient, both of which were packed into a single result data register as high and low word. In the 68020, these are the DIVS.W and DIVU.W opcodes. There are three new long forms. The first long form performs 32 bit by 32 bit division, resulting in a 32 bit quotient. The remainder is discarded. In the second long form, a 64 bit value, in two data registers, is divided by a 32 bit value, giving a 32 bit remainder and a 32 bit quotient. The remainder is loaded in the data register which held the high 32 bits of the 64 bit value, while the quotient is loaded into the other data register. The final long form uses a new opcode, DIVSL.L/DIVUL.L, to perform 32 bit by 32 bit division, giving a 32 bit remainder and 32 bit quotient. For instance, DIVSL.L (A4),D0:D3 divides the value in D3 by the value at (A4), with the resultant remainder stored in D0 and the quotient in D3.

There are many miscellaneous new instructions. EXTB sign extends a byte value in a data register to a long value, using one operation instead of EXT.W followed by EXT.L. PACK and UNPK are used to pack and unpack BCD values. I think that these two instructions were originally planned for the 68000, though they make their first appearance here. There are two new instructions for checking a value against an upper and lower bound, CMP2 and CHK2. CMP2 compares a value (byte, word, or long) in a data or address register against two values stored in memory. If the register value is less than the first value, the lower bound, or greater than the second value, the upper bound, then the carry flag is set. There are not separate opcodes for signed and unsigned compares. Instead, the type of compare is determined by the bound values themselves. For a signed compare, the arithmetically smaller value should be first, while for unsigned compare, the logically smaller value should be first. The CHK2 performs in the same way, except that an out of bound comparison results in a TRAP to the CHK instruction exception vector. As the final new instructions, there are CALLM and RTM, call module and return from module. Like the coprocessor instructions, these are too complicated for this current column. Suffice it to say that these instructions allow the use of external hardware to control prioritized access to various independent program modules. This is an extension of the system state/user state of the 68000, allowing finer control and better memory and execution protection.

New Addressing Modes

In addition to the new instructions, the 68020 has added some powerful new addressing modes. In the 68000, there are addressing modes such as (d16,An) and (d8,An,Xn). The first allows a 16 bit signed value to be added to the contents of an address register, giving an effective address. The second allows an 8 bit signed value to be added to the contents of an address register and a word- or long-sized general register, giving an effective address. Nowhere is there the ability to use 32 bit offsets, though. Not surprisingly, that ability is found in the 68020.

What the 68020 has implemented is one new catch-all addressing mode, which covers just about every possibility. Start with a long value, either in an address register or the program counter. Now, add a constant value, called the base displacement, which is either a sign-extended word value or a long value. Next, an index register is specified. The index register is found in D0 to D7 or A0 to A7, and is either a sign extended word or a long value. The resultant index register value is multiplied by a scale factor of 1, 2, 4, or 8 (with no clock cycle penalties), and added to the address register plus optional base address. In most cases, this gives a final effective address. The assembler syntax for all of this is (bd,An,Xn.SIZE*SCALE). Using the scale factor, it is now possible to access arrays of integers, pointers, and the like without using any shift instructions.

There are further extensions to this basic format, having to do with memory indirection. Yes, the indirect memory addressing modes which were so useful in the 6809, but were left out of the 68000, are back. There are two forms of indirection, called memory indirect post-indexed and memory indirect pre-indexed. Post-indexed,

with assembler syntax ([bd,An],Xn.SIZE*SCALE,od), involves adding the base displacement to the base address register. The result is used as an address to fetch a long word, which is added to the scaled index register, and finally, added to an outer displacement 'od'. Pre-indexed, ([bd,An,Xn.SIZE*SCALE],od), is similar, with the memory indirection taking place after the index register is added. In both cases, the outer displacement, like the base displacement, is either 0, 1, or 2 words long.

There is yet a further extension. While performing the effective address calculation, any of the various values involved (base register, base displacement, index register, and outer displacement) may be optionally omitted. This gives rise to a ridiculous number of possibilities. For instance, omitting everything but the base displacement results in either a new way to specify absolute long or short, or with indirection, a way to specify absolute indirect, ([bd]). As another example, the base register and indirection may be omitted, and a long base displacement may be used as the actual address in memory of an array. Loading the index register with the array index, then, allows immediate access to an array element:

```
        MOVE.W  INDEX,D0
        MOVE.L  (TABLE,D0.W*4),D1
        ...
TABLE:  DS.L    100
```

As a final exercise, it is always interesting to determine the maximum length of any instruction in a computer. For the 68020, this requires a MOVE, since two complete effective addresses are needed. The longest effective address takes 1 word for the extension, 2 words for a long base displacement, and 2 words for a long outer displacement, giving 5 words. Two effective addresses take 10 words, so the MOVE takes one word more, or 11 words total. The longest 68020 instruction, then, is 22 byte long!

As Ever, To Be Continued

As normally happens, there is still more to be said. I'll finish this up next month, with some attention to the hardware aspects of the 68020.

CoCo User Notes

by **Carl Mann**

COCO PLAYS FRONT-OFFICE HARDBALL
or,
How the Micro Stole the Christmas Bonus

Editors Note: Carl was our first "regular" Columnist with Color Micro **Journal**, and redily agreed to continue his Column in '68' Micro. Even if you have NO interest in the Color Computer, I think you will look foreward to reading this Column each month (its a standing joke here in the Office; if someone busts out laughing while working at a Computer, we know that they are working on Carl's Column). As those of you who have not seen his Column before can see from this first 'installment', "CoCo **Users Notes**" comes from a CoCo User, and ANYTHING is fair game. We hope that you enjoy it as much as we have, and feel free to drop him a note on anything relevant to the CoCo (and I mean ANYTHING!). Anyway, as they say on TV, "H-e-r-e-'-s **Carl**...
-RLN-

It isn't easy to design a single computer system to satisfy everybody's needs. The human imagination is such a fertile breeding ground for new ideas, motivations, and techniques that no sooner does the "Ultimate Machine" arrive than some innocent soul asks in all seriousness, "Yes, and can it also...?".

That's the situation I walked into when I took a full-time job as a Technical Writer in a company not far from Route 128. (The signs used to read, "128: AMERICA'S TECHNOLOGY HIGHWAY". But the traffic along 128 is Snail City from about 8:00 AM on to past suppertime. Local folks started to make jokes about the state of the traffic, the technology, and American Industry in general. Now there are obvious "fixes" applied to the signs; they now read "128: AMERICA'S TECHNOLOGY [REGION]". The word "REGION" is tacked over the old wording with sheet metal screws. Guess you can't please everybody.)

I was eager to start work, so I (characteristically) tackled the situation on two simultaneous fronts. First, I let it be known that I felt the need to be hooked up to the company's mainframe as soon as possible. Then I got serious. I

dismantled my computer room and packed CoCo, printer, disk drives, and all in to work the very next day. Two hours later I was up and running, with hardcopy rolling off the assembly line at a heartwarming rate. Within a week CoCo was churning out the first computer-generated engineering drawings ever seen within company walls. Within a month the Interface Manual that had been forthcoming for many moons was in the hands of the review committee. The boss was impressed. The workers were impressed. The company president was NOT impressed. He did NOT care for the idea of a "toy computer" in the operation. (Everybody knows that "Real Men" don't own Toy Computers, right?)

Within two weeks the proper arrangements had been made to complete my hookup to the mainframe. (It's an IBC Super Cadet, running the OASIS operating system with Spellbinder, Userbase, and some other stuff.) No sooner did I start to learn the wordprocessing software than I knew I was in trouble.

The first sign of trouble was when the boss (a charming Swiss-German fellow with a laugh that has been known to uproot even the most entrenched employee on contact) told me he had a favor to ask. "I vant you to, you know, help out za gurls vith zis Spellbinder ting", he said. "None of zem realy know how to use it, and ve don't haf za time, but vith your computer back-grrownd..." He stood in the classic "Who Knows??" posture.

What the heck. I'm a sucker for Waifs, Orphans, and the like anyway. Besides, I was itching to give that IBC a run for its money. I perched myself on a moribund Back Chair I had found in the coffee room and set to work. The reward for my impetuosity was not long in arriving.

You know, I really don't think I fully apreciated CoCo until that day. Don't get me wrong. The IBC is a mighty fine machine. So was the Steam Drill that old John Henry won his final race against. Just one difference. This time, it was the Steam Drill that lost - but the challenger is still alive and kicking.

Just for starters, consider user space. Spellbinder on a multi-user IBC allows about 12K for text. A 64K CoCo running, say, VIP Writer, has at least 40.6K of empty space for text. (You can get a wee bit over 48K if you "DUMP" the high-resolution display.) Then there's ease of use. VIP Writer allows the user to program file chaining, printer commands, and suchlike with maybe four or five keystrokes per requirement. Spellbinder demands at least twice that many to accomplish the same jobs - and its own internal codes are in gross conflict with the most-often-needed printer codes. (That's why Margaret refuses to use SB. Everytime she wants to get a circle-R "Registered" symbol out of the Qume printer, she ends up with continuous underlining. I tried it too. I got it to go, but I had to stand on my head to make it happen.)

Disk handling proved to be Spellbinder's most disastrous pitfall. Three times I filled up that dinky buffer. Three times I issued the appropriate "Save To Disk" commands. (SB uses "GD" as its command for "Get Done". I interpreted it another way by the time I had finished.) Three times the SB software ate my text, refused to tell me where it had put my text, and at last refused to accept any more text. Or anything else. (At least SB isn't as bad as UserBase. I crashed the whole darn machine trying to get out of a corner I had painted myself into with the "FORMS" option from that package.) Maybe "Real Men" don't own Good Software, either!

Then there's graphics. The IBC doesn't do them. We have a Hewlett-Packard that WILL do graphics. But the software costs $10,000.00. Yup. Ten Thousand clams. It's enough to choke a Missouri hawg. Besides, we don't have it. Not the software, anyway. And probably not the money, either - at least, not for that. For my money, we'll do just fine with my CoCo, a couple of excellent printers, and the old standbys: Stylograph (which I just got) and VIP Writer (with an occasional bow to Telewriter, although VIP works better) for text, and the amazing SDP and Graphicom for all those wonderful engineering drawings that the Engineering Department (bless its overloaded little soul) might get done "Any Month Now". Not to mention the budding little Forth Interest Group that's springing up around that EXCELLENT FORTH that Hoyt Stearns Electronics markets to CoCousers (much more to follow on this later). Can't beat it - its lexicon is full-featured, and its Semigraphic-8 editor is the best of ANY package. (Does FORTH even run at all on Z-80 systems?) Move over, you makers of musclebound micros. Stand back, you purveyors of pot-bellied minis. CoCo's still in town. (Catch one if you can.)

'Till next month,

BASIC OS-9

by Ron Voigts

Editors Note: Ron has been writing the "BasicO9 BASIC" Column in the Color Micro Journal on programming with BasicO9 since last summer. Since there is very little difference between CoCo OS-9 and the SS-50 Bus System's OS-9, and because OS-9 is still a relatively new Operating System, we have renamed Ron's Column to "BASIC OS-9" to allow him to cover OS-9 IN GENERAL. This will allow Ron to broaden his coverage to anything pertaining to OS-9 for the newer Users, including how to use OS-9 and many of its features, introductions to Pascal and C along with continued coverage of BasicO9, etc. We also feel that this kind of information will be of help to those that have not yet taken the "plunge" into OS-9 in that they can get a better feel for the fundamentals of this excellent Operating System. It has been interesting to note that the large majority of the "ex-FLEX" Users that have taken the time to really LEARN how to use OS-9 prefer it over FLEX (and those that are using OS-9 Level II are almost UNANIMOUS in their preference for OS-9), which is saying a lot considering that FLEX is such a flexible and easy-to-use Disk Operating System. We hope that the addition of Ron's Column, along with the excellent "OS-9 Users Notes" Column from Peter Dibble, will provide a broad coverage that will be of interest to all of the Readers of the '68' Micro Journal.

-RLN-

Sorting Things Out

Besides being able to crunch numbers and play games the computer has the ability to store data. It can store tremendous amounts of information. I don't think there is anyone in this country who has not been touched by the computer. If you carry credits cards, you're in a computer somewhere. The Gas, Phone and Electric companies have you in their computers. If you own or drive a car, you're in a computer. If you work or are in school, probably some computer knows about you. Not only you, but records are kept on everyone. There are millions and millions of records.

If the computer was only capable of keeping records, that would not be a great feat. Records have been kept for centuries, although I have to admit the computer does a more accurate job. The greater contribution is that the computer helps us to understand what we have stored in it. It puts the data in order. It sorts things out. Imagine trying to look up a phone number in a phone book that was not listed alphabetically. I dare say finding a particular name and number would be impossible. Thank goodness things are listed in alphabetical order. Looking for the phone number takes no more then a minute or two.

The OS-9 system is ideal for record keeping. It permits you make directories on a disk for specific purposes. Under these directories you can have more directories or actual data file. (The OS-9 Commands Manual, chapter 2 contain a detailed description.) In the files you can keep track of many things. They may contain addresses, checking account, home finances, income tax records and almost anything else you can think of. Being able to sort the record in your files can save a lot time and help to better understand them.

With BasicO9 you can write procedures to sort your files. Addresses can be put in alphabetical order, check numbers in numerical order and your income tax can be sorted by category. Going through your files can be much easier once they have been put in order.

The simplest sort is the "selection sort". Here two arrays are used. The first holds the unsorted list. The other receives the items from the list as they are sorted. Also two variables are used. One is a temporary holder for the item and the other its place in the array. An array of 6 numbers to be sorted may look like:

```
S: 14 23 39 10 45 19
T:
M=14  P=1
```

Here S is the unsorted array. T receives the sorted items. M is the first item in the array S and P is its position. Our sort goes down S until we find a number smaller. When 10 is found:

M=10 and P=4

Since no smaller numbers are found, 10 is put into array T and its position in S is flagged. Our sort now looks like:

```
S: 14 23 39 XX 45 19
T: 10
M=14  P=1
```

XX marks 10's original place in S so we don't use it again. 14 is again tested against every number in the array. This time at the end, M=14, so it occupies the second position in T. Next 23 is tested, but at the end 19 is smaller, so it gets the 3rd position in T. This continues until all numbers have been tested. The final outcome looks like:

S: XX XX XX XX XX XX
T: 10 14 19 23 39 45

All the numbers have been checked. T now holds the sorted array and S has 6 XX's indicating that the all numbers have been sorted. The biggest problem is that this method is wasteful. Whatever the size of the data to be sorted is, at least twice as much memory must be allocated. I have never seen anyone use this sort since it is memory hungry. If you're running Basic09 on a 64K CoCo, you have to be a little frugal with memory.

The next type of sort is called the "selections and exchange", better know as the "bubble sort". This sort needs only on array to hold the data and two variables. In this sort we start with the first item looking for something smaller. Here is a new set of numbers to be sorted:

S: 12 22 7 19 25 30
M=12 P=1

M is the temporary holder for the smallest number and P is its position in the array. Comparisons are made going through the list looking for something smaller. Eventually we get

M=7 P=3

The selection part has been completed and now for the exchange. 7 is put into the first position and 12 into the 3rd position. We now start with the second item and a go through the list again. This time the 12 ends up in the 2nd position. We keep doing this until all the items have been done. Eventually our list of numbers will be sorted and look like:

S: 7 12 19 22 25 30

The bubble sort gets its name from the way the smaller items rise to the top and heavier ones sink down. It is like bubbles in water; they rise to the surface.

Listing 1 is a short Basic09 demo program that lets you input in any 6 integers. The actual sort uses two FOR...NEXT loops. The "i" loop insures that each number in the array gets checked. The "j" loop starts with whatever value "i" is and sorts from there. The workings of this little program is exactly the way I described before. You can use this as a model for future bubble sorts. Its very easy to implement.

The "selection" sort takes the most number of comparisons to run. If you were to sort 100 items, it takes 10,000 comparisons (that's 100^2). The "bubble sort" for 100 items needs a 5050 comparisons (100+99+98+...+1=5050). If you can cut down the number of comparisons then the sort will go faster. This brings us to the third type of sort. It is called the "partition sort" or better known as the "quick sort". To sort 100 items it needs only 200 comparisons (100 * log 100 = 200). That is quite a savings in time.

Here is a list of 6 numbers to be sorted by the quick sort method. They are:

S: 25 5 50 7 48 32

The trick in this method is to find where the first item falls in the array and at the same time to sort items by greater or lesser then the first item. 25 is the first number in the array. Comparisons are made starting a the bottom of the list. When we get to 7 we see that it is smaller, so an exchange is made. The new list looks like:

S: 7 5 50 25 48 32

Now comparisons are made starting at the top of the list looking for anything larger. We stop at 50 and an exchange is made: The new list looks like:

S: 7 5 25 50 48 32

This sorting continues until the numbers below the number are larger and the ones above are smaller. In our list we have already achieved this. The 25 is at its correct position in the list.

Now comes the tricky part. We really have two new lists. There it the one above the 25 and the one below it. They are respectively:

7 5

and

50 48 32

Why not use the same sorting system on them? They can be treated as two separate list. Each one smaller than the previous list and therefore requiring fewer comparison. This method can be carried out until the list is down to one item. The 7 and 5 would be sorted:

5 7

and any further lists would be "one" long so no further sorting is necessary. The 50, 48 and 32 would find the 50 at the bottom. The new list would be:

48 32 50

This would leave 48 and 32 to be sorted.

One more pass would leave it sorted:

32 48

Even though the list is being subdivided and sorted the actual members are still in the original array so the outcome would look like:

S: 5 7 25 32 48 50

Does this all seem a little confusing? Just remember that we are seeking to subdivide the list into smaller units. This is why it is called the "partition sort". The savings in time is because the resulting lists from the original become smaller and smaller at a rapid pace. Less and less time is spent in sorting.

A GENERAL, ALL PURPOSE SORT

I thought this month I would give you a sort using the "quick sort" technique. This months program consists of 3 procedures, shown in **Listing 2**. There is "sort", which is the main procedure that does all the management work. Next is "qsort" which does the actual sorting. And finally there is "swap" which switches two data fields. This sort will take a file of up to 100 lines, at 80 characters or less per line, and sort them in ascending order. Let's say your file is "myfile". You would enter:

```
sort("myfile")
```

As long as "sort" is packed in the commands directory, it would load "myfile" (up to 100 lines) and began sorting it. If you appended a ":d" to the file name it would also print statistics while it sorted. So you could enter:

```
sort("myfile:d")
```

Either way it eventually prints a sorted file called, "myfile SRT". The original file is left intact. You can do whatever you want with the files. Perhaps you want to delete the original and rename the sorted file. Or maybe just leave them the way they are. This sort will take less then 100 lines and the line length can be 80 characters or less. You can change these parameters if you like. The only constraint you have is memory, as I will explain in a moment.

The first procedure is "sort". It will accept a file of up to 100 lines, each being a maximum of 80 characters. This procedure reads in a file, sorts its lines in ascending order and writes a new file with the old name and a " SRT" added on. If a ":0" is added on to the file name, a boolean variable, "dflag" is set true. This tells "sort" to print documentation about the sort like starting time, ending time, and whether the file is longer then 100 lines.

The next sort is "qsort". The LOOP...ENDLOOP sorts the first item in the list and puts it in the proper place. The first REPEAT...UNTIL searches from the bottom of the list. If it finds the top item is larger, it swaps the two lines around and begins another search in the next REPEAT...UNTIL searching from the top of the list. The search continues until a larger top item is found. Two variables insure that we do not cover the sam territory. They are "ttop" and "tbot". These are temporary top and bottom markers that are readjusted after every swap. Finally, if "p", which is our movable marker, ever equals the "ttop" or "tbot " than we have reached the end of the list.

We now have two lists. The smaller above "p" and the larger below. As long as the list are at least two lines in size we can run "qsort" on them again. This brings up an interesting technique. It is called recursion. Recursion is an interesting concept. To put it simply, "recursion" is where a procedure "runs" itself. In "qsort", it calls itself to sort the two sublists it has created. Those two running "qsorts" may call "qsort" again. This could be almost endless, but the last two IF...THEN statements prevent this from happening by insuring that our lists are at least two lines long. "Qsort" does not have to reload since the same code is reused, but a separate data area is necessary for each incarnation of "qsort". The "s" array was dimensioned elsewhere so it does not add any extra overhead, but the other variables require a new data area for every call to "qsort". It requires a little over 30 bytes of data area. If it is called 100 times in the sort that is 30X100 or 3000 bytes. Add in the original 8000 bytes for the file, plus the procedures and RUNB, and you are looking at a lot of memory tied up. This is one reason why I limited the sort 100 lines. If you feel ambitious you might try to increase this amount. I think it may be possible to get up to perhaps 200 lines.

The little procedure "swap" simply swaps two lines around. Once you've entered the procedures, you are ready to pack them. You have two choices on how to PACK them. You can pack them under one file name by entering:

```
B:pack* sort
```

and they will be packed together in the

CMDS directory under the name sort. Having them under one name makes it easier to transfer them to another disk. Just copy one file. The disadvantage is that if you wrote another procedure and wanted to use "swap" or "qsort", the new procedure would have to have a separate copy of them to use. So the alternate plan is to pack them separately by entering:

B:pack sort
B:pack qsort
B:pack swap

Now when you run "sort", "qsort" and "swap" will be loaded when needed. If you transfer "sort" to another disk, you will have to remember to copy its companions too. But then again "swap" and "qsort" can be used by other procedures. Either method works since, in a Basic09 procedure, when a RUN is encountered, memory is searched first for the module, and then the execution directory.

I think you'll be surprised with the quick sort method. Some years ago I wrote a bubble sort in basic. Whenever I used it, I found it was a good time to go and raid the refrigerator, watch a little TV or do something while it ran. You won't find this the case here. I ran "sort" and on a 100 line file it took about 30 seconds to run. Keep in mind some of that time was devoted to reading and writing the files.

Play around with some of these sorts. Devise some of your own. If you develop one you're rather proud of, send it in, and we'll share it with the other readers of this column. Kernighan and Plauger's "Software Tools in Pascal" (Addison-Wesley Publishing Co., 1981) contains an excellent Chapter on Sorting (and is one of the "bibles" on HOW to Program -- also, Pascal is fairly easy to convert to Basic09), and THE bible on Sorting is Donald Knuth's "The Art of Computer Programming; Volume 3: Sorting and Searching" (Addison-Wesley, 1973). Whether you try some sorts, or work on something else, the more you program the more you'll understand. See ya next time!

```
PROCEDURE BSORT
(* Bubble sort demonstration *)
(* Enter any 6 integers and  *)
(* it will sort  them in     *)
(* ascending order           *)
DIM i,j,m,p,b(6):INTEGER
(* input 6 integers *)
FOR i=1 TO 6
  INPUT "ENTER INTEGER>> ",b(i)
NEXT i
(* do a bubble sort *)
FOR i=1 TO 6
  m:=b(i)
  p:=i
  FOR j=i TO 6
    IF m>b(j) THEN
      m:=b(j)
      p:=j
    ENDIF
  NEXT j
  b(p):=b(i)
  b(i):=m
NEXT i
(* print the sorted intgers *)
FOR i=1 TO 6
  PRINT b(i)
NEXT i
END
```

```
PROCEDURE sort
(* This program will sort a file in ascending order *)
(* it allows a line to be 80 characters long        *)
(* And will sort up to 100 lines                    *)

(* file to be sorted *)
PARAM file:STRING[32]

(* other variables used *)
DIM i,j,path:INTEGER
DIM line(100):STRING[80]
DIM dflag:BOOLEAN

(* find out if dflag is TRUE or FALSE *)
file:=TRIM$(file)
dflag:=FALSE
IF RIGHT$(file,2)=":0" OR RIGHT$(file,2)=":d" THEN
  dflag:=TRUE
  file:=LEFT$(file,LEN(file)-2)
ENDIF

(* print start message *)
IF dflag=TRUE THEN
  PRINT
  PRINT "SORTING FILE: "; file
  PRINT "START TIME: "; RIGHT$(DATE$,8)
  PRINT
ENDIF

(* read in file, up to 100 lines *)
OPEN #path,file:READ
i:=0
WHILE NOT(EOF(#path)) DO
  i:=i+1
EXITIF i>100 THEN i=100
ENDEXIT
  READ #path,line(i)
ENDWHILE

(* middle message *)
IF dflag=TRUE AND i=100 THEN
  PRINT "MAXIMUM NUMBER OF ENTRIES READ"
  PRINT "SORT MAY NOT REFLECT TOTAL FILE"
  PRINT
ENDIF

(* SORT FILE *)
RUN qsort(1,i,line)

(* write sorted file *)
CREATE #path,file+"_SRT":WRITE
FOR j=1 TO i
  PRINT #path,line(i)
NEXT j
CLOSE #path

(* print final message *)
IF dflag=TRUE THEN
  PRINT "SORT COMPLETE"
  PRINT "SORTED NAME: "; file+"_SRT"
  PRINT "END TIME: "; RIGHT$(DATE$,8)
  PRINT
ENDIF

END
```

```
PROCEDURE qsort
PARAM top,bot:INTEGER; s(100):STRING[80]
DIM test:BOOLEAN
DIM ttop,tbot,p:INTEGER
ttop:=top
tbot:=bot
p:=tbot

(* FIRST ITEM IS SORTD TO FIND ITS POSITION *)
(* IN THE LIST.                             *)
LOOP
  REPEAT  \REM sort from bottom
    test:=s(ttop)>s(p) OR p=ttop
    p:=p-1
  UNTIL test
  p:=p+1
EXITIF p=ttop THEN ENDEXIT  \REM no more to sort
  RUN swap(s(ttop),s(p))

  tbot:=p
  p:=ttop
  REPEAT  \REM sort from top
    test:=s(tbot)<s(p) OR p=tbot
    p:=p+1
  UNTIL test
  p:=p-1
EXITIF p=tbot THEN ENDEXIT
  RUN swap(s(tbot),s(p))
  ttop:=p
  p:=tbot
ENDLOOP
(* At this point p divides the list into *)
(* lesser and greater items. Now Qsort   *)
(* is run on the two sublists            *)

IF top<p-1 THEN
  RUN qsort(top,p-1,s)
ENDIF
IF p+1<bot THEN
  RUN qsort(p+1,bot,s)
ENDIF
END
PROCEDURE swap
PARAM i,j:STRING[80]
DIM k:STRING[80]
k:=i
i:=j
j:=k
END
```

OS-9 - Need Applications??

HELP is HERE

K-BASIC

Ever since Microware released their fine OS-9 disk operating system, and that was some years back, I have been asked maybe a thousands times or better, "When are we going to start seeing more real and serious 'applications' software?" Well here it is, a way to run most all serious applications software ever written for the S50 bus systems or the 68XX! With BASIC09 and now K-BASIC, you have the best of both.

K-BASIC, in it's latest form, is the OS-9 version of a directly compatible TSC XBASIC (eXtended BASIC) programming language, and as you should know, there has been more 68XX **serious** (business, educational, scientific, etc.) software written for the TSC XBASIC version than **ANY** other programming language - including assembler. Therefore, it stands to reason that **K-BASIC** is the most efficient way (actually -**ONLY** way) to 'port' all that software from FLEX to OS-9. Finally the OS-9 user has a greatly expanded source of software to choose from. (See all those pages of XBASIC software advertised in **68 MICRO JOURNAL** over the years).

We hope to have a complete review of **this** version of K-BASIC soon, however, because we have received so many calls, from OS-9 users wanting to know when this particular version will be available, we use this means to let you know - K-BASIC the TSC XBASIC OS-9 Version is **NOW** being shipped by **S.E. MEDIA** and other distributors - see Press Release this issue, and other advertising.

The FLEX version will be following soon. Watch these columns and advertising, press releases, etc., for additional info. Reserved orders (for FLEX version) and OS-9 **immediate** orders may be placed through the **toll FREE WATTS line 1-800-339-6800,** S.E. Media - See advertising this issue.

You might note the generous update policy for K-BASIC. Now there is **no** excuse for complaining about applications software' for OS-9, or any other reason to wait!

\- - -

Editor's Note: The folks over at S.E. **MEDIA** have informed us that in order to make it easier for you OS-9 users to get started, they pass along the following **limited time** offer:

Special - **Complete Package**

The only thing additionally needed, assuming you have an OS-9 computer system and some XBASIC FLEX software, is a porting program, such as S.E. Media's **OF** - FLEX to OS-9 to FLEX software porting or transfer program and K-BASIC.

K-BASIC & OF both for the normal price of K-BASIC alone:

$199.95

So you save about $80.00, and that is a bargain! This offer is not retroactive, and will conclude April 15, 1985, so I am told by the folks at S.E. Media and Lloyd I/O.

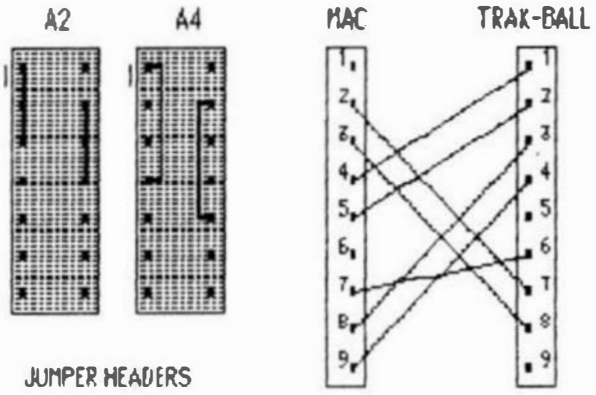
DMW

Hacking on the Mac

by
Mike Wolf
3195 Arizona
Los Alamos, NM 87544

Replace the MOUSE with a TRAK-BALL

If your desk is like mine it often gets so full of stuff (cluttered with junk) that it's hard to find enough room to maneuver the mouse. I solved that problem by adapting a Atari trak-ball to the Mac. The Mod is quite simple and is reversible so when the kids want to use it on their Atari you can restore it in a few minutes. It consists of replacing 2 ICs inside the case with jumper headers and building an adapter plug to get the connections to the correct pins. All in all about a 30 minute job. The TRAK-BALL costs $49.95 retail and less at discount stores so is a pretty good deal.



HARD DISK

I have had a Davong 10 M byte hard disc for a couple of months. This is a nice add on for the Mac. It connects to the modem port . It is a lot faster than the floppy (about 4-5 times) and really cuts down on the wait times to load files. The best part is the little message on the right side of the top line that says "867SK available" after loading system files MacWrite, MacPaint, Basic, Multiplan, all font files, and a bunch of data files onto the disc. It requires booting off the supplied floppy, but once booted the floppy can be ejected and the disc operates with the same ease as the floppy, only faster and bigger. It costs about $1600. If you need more space you can get up to 40 M bytes. The worst part is it does tie up the port. This makes it hard to use a modem or second printer. Another problem is that some software doesn't allow you to copy it to the hard disc and run from the copy. If you should get a hard disc be sure to check with software vendors to be sure you can load it onto the hard disc before you buy.

Macintosh Authors Wanted

Computer Publishing Inc. is presently investigating the possibilities of Publishing a "Pure Macintosh" Magazine. We are, therefore, requesting that offering our '68' Micro Journal Readership "first shot" at becoming Contributing Editors to the new Mac Journal Magazine.

The Macintosh will, quite obviously, become a real force in the low priced, 68000-Based, Personal Computer arena. Apple says that they sold "over 250,000" Macs the first year, and that there is "more than 300 Software Products" available NOW. We feel that with the number of Mac Users and Advertisers, we can put together an excellent Magazine and PAY the Authors of the Articles and Reviews that we accept for Publication. The payment will be consistent with the Industry.

Mac Journal will be primarily oriented towards "How to USE the Mac and the Products that are available for it". While we hope to be able to devote a small section of Mac Journal to "advanced" topics such as Programming, we realize the the "normal" Mac User is more interested in "How do you USE a specific Data Base Management Package?", or "how to use it more efficiently?", or "What does this particular Program or Product provide, and what can be done with it?". In other words, the material should somewhat instructional in nature.

If you feel that you would be interested in writing about the Mac and its Products, and get PAID for it, send for a

Mac Authors Application to

Mac Journal
5900 Cassandra Smith Road
Hixson, TN 37343

DON'T LET THIS GOLDEN OPPORTUNITY
PASS YOU BY --
Contact us N-O-W!!

P.S. -- If you think you could do a good job writing Articles for Mac Journal, but don't HAVE a Mac yet, go ahead and get in touch with us. We are working on the possibilities of being able to provide a few Macs for the specific purpose of supporting the Mac Journal.

FLEX Routines

General Applications

I have not seen any articles showing the general usefulness of the the FLEX internal user callable routines such as OUTDEC, to do conversion of internal binary to ASCII decimal in memory. I recently needed to do this, to convert the binary volume number found in a disk's System Information Record to a string in ASCII in memory, for use in Dennis Milligan's program DIRECTOR. This program produces a sorted master directory of all my FLEX disks (see Nov. 1981 issue of '68').

Rather than incorporate the whole 9 yards of the conversion code from binary to ASCII decimal, it seemed possible to use the existing user routine in FLEX. The only trouble is that OUTDEC is intended to print its result, instead of handing it over in a register. Fortunately there is a way built into FLEX to accomplish the desired conversion. FLEX has a jump table with vectors for outputting to a printer or CRT, but these may be altered to steer the output to other devices. The steering mechanism is not limited to hardware devices and their drivers, however. The output can just as easily be steered to a subroutine, where it can be written to memory for other uses.

The decimal conversion is more complicated and longer than the others, but this same steering can be done for the simpler routines OUTHEX and OUTADR in the same manner.

In order to create an ASCII string of decimal digits, given a 16 bit binary number in memory, the following procedure can be used.

1. Set up a pointer to the area to receive the string, store it in memory, name this ITEMP2.

2. Call FLEX routine RSTRIO ($AD2A/$CD2A) to clean up I/O vectors.

3. Increment the flag at $AC81/$CC81 to suppress interference from TTYSET parameters.

4. Set the address at OUTCH ($AD18/$CD18) to the address of your subroutine that will store the string. Do not overlay the jump instruction at AD0F/CD0F.

5. Load X with the address of the binary number and load B with a 1 if you wish leading blanks, zero for blank suppression, and call OUTDEC.

6. Call RSTRIO to reset vectors, and exit.

A sample program segment incorporating this is given below. This actually is a patch to Dennis Milligan's DIRECTOR program, so that the operator does not have to type in the disk description. The disk number is taken from each disk's System Information Record, converted and placed in the same string space as in the original program. I have found better than a 2 to 1 increase in speed running the program, and even more counting my frequent mistakes. There is a preliminary segment which reads the SIR from track 0, sector 3 and moves the disk number from it to scratch memory called ENI. The conversion package picks it up from there and lays it down as ASCII at DISKN2.

The location of the patch can be identified by the label NOFILE in the original source listing of DIRECTOR. The entire segment down to OUTIT is replaced.

DIRECTOR PATCHES

Ralph Moglund
Seattle, Washington

```
                        *
                        *THIS LISTING IS FOR PATCHES TO THE DIRECTOR
                        *PROGRAM BY DENNIS MILLIGAN, '68 JOURNAL,
                        *NOVEMBER 1981, PAGE 25.    THE PATCHES PERMIT
                        *READING THE VOLUME NUMBER OF THE DISK FROM
                        *THE DISK ITSELF, RATHER THAN BY THE OPERATOR
                        *TYPING IN THE INFORMATION.
                        *
                        *EQUATES TO FLEX
A000                    BASE    EQU     $A000
AD03                    WARMS   EQU     BASE+$D03
AD15                    GETCHR  EQU     BASE+$D15
AD1B                    INBUFF  EQU     BASE+$D1B FILL KEYBD BUF
AD1E                    PSTRNG  EQU     BASE+$D1E
AD24                    PCRLF   EQU     BASE+$D24
AD27                    NXTCH   EQU     BASE+$D27
AD2D                    GETFIL  EQU     BASE+$D2D
AD33                    SETEXT  EQU     BASE+$D33
AD3F                    RPTERR  EQU     BASE+$D3F
AD34                    ADDBI   EQU     BASE+$D34
AD18                    PUTCHR  EQU     BASE+$D18
AD39                    OUTDEC  EQU     BASE+$D39
AD48                    INDEC   EQU     BASE+$D48
AC22                    OUTBW   EQU     BASE+$C22
AC21                    SPIO    EQU     BASE+$C21 SPL IO FLAG
AD10                    OUTCHA  EQU     BASE+$D10
AD2A                    RSTRIO  EQU     BASE+$D2A
AC24                    FILOA   EQU     BASE+$C24 FILE OUTP ADDR
F018                    OUTCH   EQU     $F018
B403                    FMSCLS  EQU     BASE+$1403
B406                    FMS     EQU     BASE+$1406
BE7F                    STACK   EQU     $BE7F
                        *
                        *STORAGE
                        *
0002                    ENI     EQU     2       SCRATCH AREA ON BASE PAGE
0003                    XLOW    EQU     3
0024                    ITEMP1  EQU     $24
0024                    ITEMP2  EQU     $24
0006                    DISKN2  EQU     6
0023                    DRIVE   EQU     $23
                        *
                        ****************************************
                        *READY TO COPY DIRECTORY
                        *
                        ****************************************
                        *
0000 CE 09 F6   NOFILE  LDX     #SIRFCB GET FROM SIR
0003 96 23              LDA A   DRIVE
0005 A7 03              STA A   3,X
0007 86 03              LDA A   #3
0009 A7 1F              STA A   31,X    POINT TO SIR
000B 6F 1E              CLR     30,X    TRACK

DIRECTOR PATCHES

000D 86 09              LDA A   #PROSECT
000F A7 0A              STA A   $,X
0011 BD B4 06           JSR     FMS
0014 27 03              BEQ     GETSIR
0016 7E 04 EB           JMP     ERROR
                        *
0019 CE 00 06   GETSIR  LDX     #DISKN2 PUT SPACES IN BLK
001C 86 20              LDA A   #$20
001E A7 00      GETSR2  STA A   0,X
0020 08                 INX
0021 8C 00 0E           CPX     #DISKN2+8
0024 26 F8              BNE     GETSR2
                        *
0026 CE 09 F6           LDX     #SIRFCS
0029 EE 5B              LDX     $5B,X   GET DISK NO
002B DF 02              STX     ENI     SAVE IT
0020 CE 01 BD           LDX     #MSG1
0030 BD AD 1E           JSR     PSTRNG
0033 CE 00 02           LDX     #IN1    POINT TO DISK NO
0036 C6 01              LDA B   #1      FLAG FOR SPACE SUP
0038 BD AD 39           JSR     OUTDEC  PRINT IT
003B BD AD 24           JSR     PCRLF
                        *
                        *CAPTURE OUTPUT OF OUTDEC AND PUT IN DISKN2
                        *
003E CE 00 06           LDX     #DISKN2 INIT POINTER
0041 DF 24              STX     ITEMP2
0043 BD AD 2A           JSR     RSTRIO  RESET ALL I/O
0046 7C AC 21           INC     SPIO    IGNORE TTYSET
0049 CE 00 5C           LDX     #LDBUF  RESET OUTCH VECTOR
004C FF AD 10           STX     OUTCHA
004F CE 00 02           LDX     #IN1    POINT TO BINARY
0052 C6 01              LDA B   #1      FLAG TO LEADING BPACES
0054 BD AD 39           JSR     OUTDEC  CONVERT
                        *DONE, SO RESET VECTORS
0057 BD AD 2A           JSR     RSTRIO
005A 20 0C              BRA     OUTIT   PROCEED
                        *
                        *OUTPUT OF OUTDEC GOES HERE
                        *
005C 07 24      LOBUF   STX     ITEMP1  SAVE X
005E DE 24              LDX     ITEMP2
0060 A7 00              STA A   0,X     PUT CHAR HERE
0062 08                 INX
0063 DF 24              STX     ITEMP2
0065 DE 24              LDX     ITEMP1  RESTORE X
0067 39                 RTS

DIRECTOR PATCHES

                        *
                        *FORMAT
                        *
                        *  FILENAME EXT  DISKNO
                        *  ------------  ------
                        *  DIRECTOR ASM  9116
                        *  FINANCE  BAS  113
                        *  CRUD     JRX  232
                        *  STOFF    TXT  101
                        *
                        ********************************
                        *
                        *LOGIC
                        *
                        * 1. GET DIRECTORY INFO
                        * 2. CHECK FIRST CHAR (MINUS OR ZERO)
                        * 3. COPY FILENAME.EXT TO MASTER
                        * 4. COPY DISKNO TO MASTER
                        * 5. COPY START/END TR/SECT
                        * 6. COPY CREATION DATE
                        *
                        *ON LAST ENTRY IN DIRECTORY, ASK OPR IF
                        *THERE IS ANOTHER DISK TO DO
                        *
                        *SET UP DIRFCB FOR DRIVE #1
                        *
0068 CE 08 B4   GOTIT   LDX     #DIRFCB
                        *.....................

NO ERROR(S) DETECTED
```

More on the 512K Mac

by
Frank Henriquez
500 Landfair Ave.
LA., CA. 90024

The Apple Macintosh was originally introduced with only 128K of memory (remember when you could impress your friends by showing them your 6800 computer with 32K of RAM?). Some argued that this wasn't enough memory for some of the more sophisticated programs that they were hoping to use or write. Still others thought that limiting a 68000 CPU (with a 16 megabyte address range) to only 128K was... well, obscene. While 128K of RAM may seem like a great deal of memory to those of us accustomed to only 64K, it is certainly not enough for the Macintosh. The Mac's high resolution screen takes up almost 20K of memory and the icon/mouse driven operating system (with all its bells and whistles installed) can take up another 20 to 40K chunk of memory. This leaves you with less than 80K for your applications and data ("application" is the chic new way to refer to a program). Of course, many programs won't take up 80K, but there are quite a few Macintosh programs that will gladly eat up 30 or 40 K of RAM, and since the "System Language" is Apple's UCSD Pascal, most "applications" tend to be BIG.

The solution to the Mac's memory crunch is simply to add more memory. By replacing the 16 64K memory IC's with their 256K counterparts, Apple was able to quadruple the Mac's memory capacity. However, to cut manufacturing costs and increase reliability, Apple soldered all the IC's to the main logic board. This means that, if you are not an expert with Soldering Tools, you will have to buy a new main logic board with the 256K memory chips already installed to upgrade to 512K (of course, this will also get you the latest ROM Version at the same time).

A 512K Mac acts much like a 128K Mac; you won't notice much of a difference until you try some memory intensive programs. If you're planning on doing any extensive programming, the extra memory will make your life a lot easier (some Mac assemblers will leave you with less than 47K of memory for your program on a 128K Mac, and the Compilers are worse). Also, there are new versions of MacWrite and MacPaint that make good use of the extra memory (you can now hold a complete MacPaint page in memory, which makes moving through the picture almost instantaneous). However, don't assume that a 128K Macintosh is useless; it's just that the extra memory adds so much more potential to an already powerful computer.

As a UCLA employee, I was able to get my Macintosh at a considerable discount through the University. I originally bought a 128K Mac, but I upgraded it to 512K as soon as the new boards became available. The upgrade is usually done at an Apple dealer, but I decided to do it myself when I heard that the dealer would keep the old board.

Taking the Mac apart was a bit more difficult than I had expected. After setting aside a clean workspace and covering it with a rubber mat to protect the Mac's exterior and screen, I located all the screws that hold the Mac together. The back shell of the Macintosh is attached by five #10 Allen-type screws (you can use a 3/32" Allen Wrench to remove them). Two of the screws are visible on the back, two are recessed below the handle, and the last is inside the battery compartment. The two exposed screws and the one inside the battery compartment were easy to remove, but the two screws below the handle were extremely difficult to reach. I made a special tool by cutting the Allen wrench off right below the bend. Then I glued and crimped the straight piece of the wrench into a 7 inch long tube to make it long enough to reach the two screws under the handle. You don't have to make your own tools, though. A screwdriver with a Torx T-15 tip and a 6 inch (or longer) shaft should do the job. As a last resort, you may try borrowing the tools from an Apple dealer; some dealers are quite friendly and will gladly lend you the tools.

After all the work I went through to remove the screws, I was surprised and frustrated to find that the back shell was still firmly attached to the rest of the computer. After much head scratching, I figured out that the shell was a tight friction fit, so with some gentle prodding and pulling (pushing on the output port connectors, and maybe use a ruler in the crack back of the Front Face of the Mac), I managed to get the shell off.

Inside, I saw the back of the CRT tube with the Video and Power Board to one side. Below the CRT, inside a small metal shield, is the internal disk drive ,and below that, inside it's own protective shield, is the main Logic Board. There are only two connections to the main logic board; a ribbon cable from the internal disk drive that attaches to a plug right behind the external disk drive connector, and a molex connector from the power and video board which powers the main board and takes the video and sound

signals from the main logic board to the video and power board. Once these two cables are removed, the logic board slid out freely. Before I touched any of the boards though, I made sure that I was grounded (to keep static discharge from damaging the computer). Now comes the easy part: I pulled out the old 128K board, slid in the 512K board, reconnected both cables, squeezed the back shell on and replaced the screws.

All told it took me about an hour to build the tool, remove the screws, pry off the back shell and replace the logic board; if I had to do it again it would probably take less than 15 minutes.

I examined my old 128K board and found it to be quite different from the new 512K board. This may no longer be true with more recent 128K Macs; Apple is using the same 512K board for both the 128K and 512K versions of the computer. The only difference is in the memory IC's, a jumper and some resistors. In fact, there are markings on the PC board showing which components have to be added or deleted to make either version.

You may be asking yourself "Gee, if I have the new version of the board, all I have to do to upgrade is unsolder the 64K IC's and replace them with the 256K...". Well, I hate to be a party pooper, but it's not that simple. The main logic board is a four layer PC board. This means that there are actually four copper layers running through the board carrying signals. Unsoldering a multilayered board is very tricky and difficult; if you don't have an excellent soldering iron and the proper tools (and the experience), you will probably damage a trace or two (say, in the second layer) and you'll ruin your computer. If you are experienced with soldering and unsoldering multilayered boards, then you may be able to save a bundle on the upgrade (at about $15 a chip, the sixteen 256K IC's would cost you about $240; that's less than a third of the cost of the upgrade).

You're probably wondering what the risks are in doing your own upgrade. Well, Apple is extending the warranty on every upgrade, so if 90 days of extra warranty are worth more to you than a spare board, then have a Dealer do it. If you've been dying for an excuse to peek into the Mac (like me) and would like to have a spare board, then get the proper tools and do it! But what about the spare board? Well, you could keep it in storage, in case the main board fails, or you could try upgrading it, or use it as the basis for a portable Mac, or you could explore the possibilities of adding an expansion bus, or even frame it and hang it on the wall next to your new, fetter Mac...

BIT Slicer

ADVENTURES OF THE BIT SLICER

by
Frank L. Hoffman
LLOYD I/O
19535 NE Glisan
Portland, OR 97230
USA
(503) 666-1097

I've been sitting at my keyboard slicing bytes into bits, killing bugs, and otherwise spending a lot of time chewing code. It has occurred to me that some of this "raw data" might be suitable for "entry" into the thought processes of other "68 Micro Journal" readers. Some of you will recall the mildly humorous articles in 'Forum 68' (a magazine now in the buggy hereafter) dealing with the inner workings of computers and assembly language. In this column, I am picking up where I left off (to the best of my recollection).

First, permit me to review a little of who am I and where I came from. I own a software house (LLOYD I/O) here in Portland Oregon, which has been doing business in the S50 market for three and a half years. I started out with the CRASMB(tm) cross assembler (supports 13 CPU's), ED/ASM(tm) (which was licensed by Frank Hogg Laboratory for their version of FLEX(tm) for the COCO), DO(tm) (a SHELL control language, or pica-BASIC which was licensed by GIMIX for their OS9(tm) systems), and CRASMB 16.32(tm) (a 68000 CPU cross assembler). More recent developments include the Search and Rescue Utilities(tm) (for OS9, free with most of our software), PATCH(tm) (a modem program for OS9), and K-BASIC(tm) (an XBASIC compatible BASIC compiler for OS9 and FLEX.)

I am your basic computer hacker, having recognized early on the computer's ability to save time by programming as opposed to building logic board ad infinitum. Frankly, I'd rather be here at the terminal than almost any other place on earth. I have a Televideo 924 connected to a GIMIX III system running OS9 and FLEX with a 19 Mb hard disk. This has been my basic development system for over a year. Previously I used GIMIX boards in a hollowed out SWTP box using the old mother board and power supply (which hick-ups and

resets the computer at the most inconvenient times.) At the ends of the peripheral I/O lines are a modem, plotter, and two printers.

During the course of generating these products and some vertical market software/hardware items, I have developed some unique solutions for some sticky problems. In this article, and others to follow, I will be sharing some of these solutions with you. I hope you'll be better informed, and to some extent entertained, as you tune in periodically for -- ADVENTURES OF THE BIT SLICER.

It is a very curious thing, how OS9 has existed for some 5 or 6 years (?) now, yet during most of that time a serious set of file manipulation utilities has not existed. Today there are about four sets of utilities available: Microware's, D.P. Johnson's, Frank Hogg's, and LLOYD I/O's. One of the biggest problems with OS9 was the necessity to type command lines over and over again to take some kind of action on different files. The solution to this problem has at last been implemented in OS9 for the 68,000. This is wild card/field file name search in SHELL. I believe most of the utility packages for 6809 OS9 provide a similar file name search program that sends matching file names to standard output. The other utility programs are connected to the search program using a pipe in order to receive the file names. (GIMIX includes our utility package with each of their OS9 systems.)

I honestly don't know why no one came out with utilities like these before. But, once the leak started to trickle, the dam broke. It's something to wonder about! Perhaps it was the lack of information about the inner workings of OS9.

Putting that aside for the moment, any OS9 user who is not using one of the available file-name match-list search programs is living in the dark ages. OS9 becomes much friendlier, speedier, and less prone to error.

For example, suppose you want to delete only the files ending in ".asm", ".r", and ".con". The following line does it for you.

```
SDIR . *.asm *.r *.con ! idel -i
```

This example assumes you are using our Search and Rescue Utilities package. "SDIR" is the search program. It will search the directory "." (the current data directory) for all files ending in ".asm", ".r", and ".con". The asterisk character is used to denote a wild field of any number of characters. SDIR sends the file names found to standard output which has been piped to the program "idel". "Idel" is a file delete program that can optionally get file names from standard input. Hence, the option "-i" causes it to do so. The whole intent of these utilities is to facilitate file manipulation. The list of possibilities includes:

* deletes
* copies
* moves
* lists with pagination
* searching for specific strings in a file
* filtering (removing or adding extra characters)
* comparing the file's last modified date to a specific time or to another file and if true send the file name to another utility through a pipe.

These are just a few examples. Filtering files is the most common use for these utilities. The reason lies in the fact that every programmer in existence has his own unique approach to filtering. For example, our K-BASIC run-time package is supplied in assembly source code. The main file is over 8,000 lines long. This file has been filtered to remove all comments, and extra spaces enabling us to supply it on one or two disks. I use a program called PACK to do it. It was developed for use with my OSM(tm) assembler, because it has the complete mnemonic look up table built in and knows how to handle the comments after an operand. Some mnemonics have no operand. PACK looks up each mnemonic to check if an operand follows and removes the comments. Mnemonics not found are considered to be macro calls and are handled as such. It works very well, and reduces a well commented source file by as much as 50%. The side benefit of a PACKed source file is quicker assembly because the assembler doesn't have to wade through the blank and comment lines and extra spaces.

UNIX has a utility program called MAKE which manages the compilation of a program. Its basic function is to compare the last modified date of a source code file to the

FINALLY!!

Now you can run

TSC XBBSIC Programs,

COMPILED to Asmb. Lang.,

under **OS-9™**, **CoCo OS-9**,

or **FLEX™** with

K-BASIC™

☎ TOLL FREE
1-800-338-6800
For Ordering

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

K-BASIC under **OS-9** AND **FLEX** will now compile
TSC BBSIC, XBBSIC, and XPC Source Code Files

K-BASIC now makes the multitude of **TSC BBBSIC** Software available for use under **BS-9**. Transfer your favorite **BASIC Programs** to **OS-9**, compile them, Assemble them, and **BINGO** -- usable, multi-precision, familiar Software is running under your favorite Operating System! -**BB**-, get **FBSTEB** operation with **LESS** Program Space under **FLEB**.

K-BASIC (**OS-9** or **FLEB**) **including** the **BSM Bssembler** ---- **$199.00**

Limited time **SPECIAL** offer!

Until **BPBIL 15, 1985**

South East Media and Lloyd I/O is offering OS-9

K-BASIC Compiler w/ OSM Assembler **BND**

O-F "FLEX <-> OS-9" Transfer Program

BOTH for **$199.95**

Offer **ONLY** valid when ordered as a **Package** from **South East Media**
Free Disk offer does not apply to this **Special Package Price**

OS-9 is a Trademark of **Microware Systems Corp.**
FLEX is a Trademark of **Technical Systems Consultants, Inc.**
K-BASIC is a Trademark of **Lloyd I/O**

VISA MasterCard

*FLEX is a trademark of Technical Systems Consultants
**OS9 is a trademark of Microware

☎ TOLL FREE
1-800-338-6800
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

FREE DISKETTE WITH EVERY $50 PURCHASE

TOLL FREE 1-800-338-6800 For Ordering

TELEX 558 414 PVT BTH

SOUTH EAST MICRO

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™

SOFTWARE

!!! SPECIALS !!!

"Year End CLEARANCE" --- While they Last

FLEX Software ----------

TSC "FLEX Utilities"	was $75.00, NOW only	$60.00
TSC "Sort Merge"	was $75.00, NOW only	$60.00
TSC "6809 BASIC"	was $75.00, NOW only	$60.00
TSC "Extended BASIC"	was $100.00, NOW only	$85.00
TSC "DeBug"	was $75.00, NOW only	$60.00
TSC "FLEX Diagnostics"	was $75.00, NOW only	$60.00
TSC "Text Processing System"	was $75.00, NOW only	$60.00
TSC "6800/1 Cross Assembler"	was $250.00, NOW only	$220.00

LUCIDATA "TEKPAK" (FLEX9, 5 1/4") -- A Pascal implementation of the Tektronix 40xx Terminal Control System, with Pascal SOURCE. The Manual includes a discussion of how to utilize this package in the graphical library in implementing Vector Drawing, Point Plotting, etc., up through Windowing and Clipping concepts.
Normal Price, $100.00; NOW only $85.00

OS-9 Software ------

MICROWARE "OS-9 File Handler Toolbox";	NOW only $70.00
MICROWARE "Relocating Macro Assembler";	NOW only $170.00

!!! NEW NEW NEW !!!

Computer Systems Consultants, Inc.

CHADEN TELECOMMUNICATIONS PROGRAM
Menu-Driven; supports Dumb-Terminal Mode, Upload and Download in non-protocol mode, and the CP/M "MODEM7" Christensen protocol mode to enable communication capabilities for almost any requirement. Written in "C".

FLEX, CCF, OS-9, UniFLEX; with complete Source - $100.00
without Source - $50.00

X-C-H -- TRUE (not Macros) CROSS ASSEMBLERS
Use your 6809-Based Computer System for developing Software for 1802/5, 6800/01/03/11, 6804, 6805, 6809, 6502/3, 8080/5, 8048, 8051, Z-80, and 68000 Systems. Provides the Assembler Language and Listings normally used on the target Systems. Written in "C"; produces Motorola S-Text for machine independence.

FLEX, CCF, OS-9, UniFLEX each - $50.00
any 3 - $100.00
the complete set (including the C Source) - $200.00

LLOYD I/O

K-BASIC -- A "Native Code" BASIC Compiler
Level 1 K-BASIC supports sequential files, floating point, 3 sizes of integers, string variables, and arrays. The single-pass compiler compiles to Assembly Language Source Code (which may be assembled by the included OSM Assembler, or by the CRASMB Cross Assemblers). Conditional assembly is used to reduce the size of the run time package. (See Review in Oct. '84 issue of '68' Micro Journal.)

FLEX, CCF, OS-9 Compiler with OSM Assembler - $199.00

OSM -- Extended 6809 Macro Assembler
Provides local labels, Motorola S-records, and Intel Hex records. Also generates OS-9 Memory modules under FLEX, allowing the maintenance of source code programs for both OS's on one System.

FLEX, CCF, OS-9 $99.00

CRASMB -- 8-Bit Macro Cross Assembler
Same features as OSM, cross-assembles to 6800/2/8, 6801/3, 6804, 6805, 6809, 6811, 6502, 1802, 8048, 8080/5, Z-8, Z-80. Fully supports the target chip's standard mnemonics and addressing modes.

FLEX, CCF, OS-9 full package -- $399.00

CRASMB 16.32 -- Cross Assembler for the 68000
Same features as 8-Bit Cross Assemblers above
$249.00

Compusense Ltd.

CRUNCH COBOL -- COBOL Compiler
This COBOL Compiler supports a large subset of ANSII Level 1 COBOL with many of the useful Level 2 features. Full support of the FLEX File Structures is implemented including Random Files and the ability to process Keyed Files. Large programs can be segmented and linked at runtime, or implemented as a set of Overlays. The System requires 56K and CAN be run with a single Disk System.

FLEX, CCF Normally $199.00
Special Introductory Price (while they last) -- $99.95

ASSEMBLERS

Southeast Media

ASTRUK09
A "Structured Assembler for the 6809" which requires the TSC Macro Assembler. Allows direct use of structured statements such as IF, ELSE, DO, REPEAT, etc., and provides indented level formatting of the listing so that the structure is apparent. Re. '68' Micro Journal, Sept. '83 (program was called "STASM09"; has been renamed due to conflicts).

A User reports
"... I'm very pleased and am now writing almost exclusively in (ASTRUK09). I've selected it over --- for all future systems development... As (one) of my early evaluations, I rewrote a rather elaborate routine originally done in assembly. Out of the 1000 bytes of code generated, the (ASTRUK09) version used only 20 more bytes than the original. --- could not handle this program since it uses triple-precision fixed point arithmetic... I have a large body of code already written that is incompatible with --- constructs. No problem with (ASTRUK09) and the structure sure helps in understanding the logic!"
F, CCF - $99.95

TSC

Macro Assembler
The FLEX STANDARD Assembler. Special -- F,CCF $35.00

Relocating Assembler w/Linking Loader
Use with many of the C and Pascal Compilers. F,CCF $150.00

Great Plains Comp. Co.

RMAC
Relocating, Recursive-Macro Assembler and Linking Loader.
F,CCF $120.00; w/Source $240.00

OmegaSoft

FRALLI
Relocating Assembler and Linking Loader
F,CCF $125.00; for One Year Maint., add $50.00

Windrush Micro Systems

MACE, by Graham Trott. F,CCF - $98.00

VISA MasterCard

** SHIPPING **
Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

TOLL FREE 1-800-338-6800 For Ordering
SOUTH EAST MICRO
5900 Cassandra Smith Rd. Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

Availability Legends ---
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

!!! Please Specify Your Operating System & Disk Size !!!

DISASSEMBLERS

Computer Systems Consultants

SUPER SLEUTH

Computer Systems Consultants Super Sleuth is a "Time Tested", reliable, PROVEN Disassembler that has gained acceptance through out the SS-50 Bus Community as an extremely POWERFUL, INTERACTIVE, Software Tool. The Super Sleuth Software Package consists of 3 Programs; SLEUTH (the Disassembler), CHGRAM (used to globally Change Labels to a meaningful Name), and XREF (a Cross Reference Generator for Source Code Files). SLEUTH will Disassemble Memory Resident 6809 Code and 6800, 6801, 6802, 6803 (the "Baby CoCo"), 6805, 6808, 6809, and 6502 (Apple, Atari, Commodore, etc.) Binary Disk Files. (See Aug. '83 '68' Micro Journal "Color Users Notes" Column for a full Review.)

Color Computer	SS-50 Bus (all w/ Source)
CCD (32K Req'd)	
Obj. Only $49.00	F, $99.00
CCF, Obj. Only $50.00	U, $100.00
CCF, w/Source $99.00	O, $101.00
CCO, Obj. Only $50.00	

ALL Computer Systems Consultants Software
runs on the Color FLEX Systems
ALL in stock
call 800-338-6800
for IMMEDIATE DELIVERY

Computer Systems Center

DYNAMITE +

An "easy to use", powerful Disassembler for Disk Resident 6809 and 6800 Binary Files. Allows the development of a "Control File" of various Program "Boundaries" during successive disassemblies; can use a Label File which automatically replaces a Hex Location with a Label Name; includes an XREF Utility; etc. Label Files provided for Mini-FLEX, FLEX2, FLEX9, Color Computer (for use with Color FLEX Systems), etc. OS-9 Version includes special OS-9 options.

CCF, Obj. Only $100.00	CCO, Obj. Only $ 59.95
F, " " 100.00	O, " " 150.00
U, " " 300.00	

COMPILERS and DECOMPILERS

6809 "Structured" Assembly Lang. Compilers

Windrush Micro Systems

PL/9

By Graham Trott. A combination Editor/Compiler/Debugger, all in ONE PACKAGE; provides a totally INTERACTIVE Program Development Cycle. The Single-Pass Compiler supports large Symbol Names; Variable Types; Pointers; Control Structures (similar to 'C' or 'Pascal'); Stack, A-, B-, and D-Register manipulation; etc. The Source-Oriented Trace/Debugger provides Single Stepping, Break-pointing, etc. An excellent Software Development Tool which provides for the maximum utilization of the power of the 6809.

F, CCF - $198.00

Whimsical Developments

WHIMSICAL

(Ver. 1.6, with Real No. support)

"..., I have to give this one a VERY High rating, ..."; Ron Anderson, Sept. '83 Issue of '68' Micro Journal.

Need the Ease of Design and Maintainability of "Structured Programming" WITHOUT losing the Speed and Control of Assembly Language? Then WHIMSICAL was designed for you! This Single Pass, Recursive Descent Compiler provides the tool for developing software from simple Utilities to MAJOR Systems. Easy to develop programs can be targeted for other 6809 Machines. Features unified, user-defined I/O; produces ROMable, relocatable, recursive, re-entrant Code; Structured style and statements with Procedures and Modules (Modules can be pre-compiled for inclusion after verification); supports Byte and Double-Byte primitives with 3 types of Integers (up to 32 bit), 6-digit Real Numbers, Char and Boolean, and unlimited sized Arrays (vectors only); Interrupt handling; unlimited length Variable Names; Variable Initialization (defaults to $00); Include "Source File" directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. No separate Run-Time pack required (Run-Time subroutines inserted as called during compilation). It is a FAST Compiler which produces FAST Code (normally produces 10% less code than PL/9).

F and CCF - $196.00

FREE DISKETTE WITH EVERY $50 PURCHASE

'C' COMPILERS

Windrush Micro Systems

C Compiler

By James McCosh. Full featured C Compiler for the FLEX Operating System (lacking ONLY "bit-fields"), including an Assembler. Requires the TSC Relocating Assembler IF the user wishes to implement his own Libraries.

F and CCF - $295.00

Introl

C Compiler

A full-featured C, streamlined for the 6809. Generates very efficient object code. Output "benchmarks" close to 10MHz 68000 in 8 Bit Operations; 1.5 times faster than a 4 MHz Z80 when using a 2MHz 6809 System (Re. p 43, '68' Micro Journal, May '83). Floats, etc.

F, CCF, and O - $375.00
U - $425.00
One Year Maint. - $100.00

PASCAL COMPILERS

TSC

PASCAL Compiler

Native Code Compiler (UCSD Oriented).

F and CCF - $200.00

Lucidata

PASCAL Compiler

P-Code Compiler (ISO Standard). Designed especially for Microcomputer Systems; Run-time System checks available resources for each task, allowing operation on even minimal computer systems. Allows linkage to Assembler Code for maximum flexibility.

F and CCF 5" - $190.00
F 8" - $205.00

OmegaSoft

PASCAL Compiler

For the PROFESSIONAL; ISO Based, Native Code Compiler. Primarily for Real-Time and Process Control applications. Use custom I/O devices in place of the Pascal INPUT and OUTPUT; Long (nt. (32 Bit); Dynamic length strings; Interrupt processing, ROM-able, PIC, Re-Entrant Code, etc. POWERFUL! Includes Source for the Symbolic Debugger, Runtime, and several Utilities. Requires a "Motorola Compatible" Relocating Assembler and Linking Loader.

F and CCF - $425.00
One Year Maint. - $100.00

DECOMPILERS

Southeast Media

DUB (A UniFLEX "basic" De-Compiler)

Re-Create a Source Listing from UniFLEX Compiled basic Programs. Easy to Use; works w/ ALL Versions of UniFLEX basic; Output to Disk or Terminal. Time TESTED and PROVEN; SOLID!

U - $219.95

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

!!! Please Specify Your Operating System & Disk Size !!!

FREE DISKETTE WITH EVERY $50 PURCHASE

TOLL FREE 1-800-338-6800 For Ordering

TELEX 558 414 PVT STH

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™

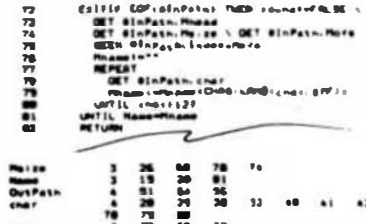
SOFTWARE

UTILITIES

Southeast Media

BasicO9 XRef

This BasicO9 Cross Reference Utility is a BasicO9 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also included is a Program List Utility which outputs the listing without the overhead of building the cross reference table, which allows it to run considerably faster when only a "pretty printed" listing with line numbers is desired. Requires BasicO9 or RunB for operation.



O and CCO - Obj. Only -- $39.95
O and CCO - w/ Source - $79.95

Southeast Media

O-F

---- OS/9 to FLEX - FLEX to OS/9 ----

Finally; the barrier has been removed between OS/9 and FLEX formatted disks! Now you can READ from, and WRITE to, a Single Sided 5" or 8" FLEX diskette from OS-9 with O-F. O-F is a new and unique program, written in BASICO9 (with Source), that performs the following functions;

REFORMAT: A BASICO9 Program that reformats a chosen amount of an OS-9 disk to FLEX format so it can be used normally by FLEX.

FLEX: A BASICO9 Program that does the actual read or write function to the special O-F Transfer Disk, all selected from a user-friendly menu. Functions provided include reading the FLEX Directory, Deleting FLEX Files, Copying both directions, etc. All selections are interactive and complete, including all necessary prompts to the operator.

FLEX users can read, write and use the special disk as any other FLEX disk, provid d the FLEX directory is not allowed to continue beyond track zero (too many files).

O - $79.95

Southeast Media

COPYMULT

--- Copy LARGE Disks to several smaller disks ---

The following FLEX utilities allow the backup of ANY size disk to any SMALLER size diskettes (Winchester to 8's or 5's, 8" to 5's, etc.). By simply inserting diskettes as requested by COPYMULT, a large disk system may be downloaded to your present floppy disk system, any size. No n ed to fiddle with directory deletions or any of the other tedious operations that must be done using the normal copy routines.

COPYMULT.CMD understands normal "copy" syntax and always keeps up with files already copied by maintaining directories for both host and receiving disk system, eliminating hours of tedious keyboard entries and other time consuming cleanup chores.

BACKUP.CMD is a special program that downloads "random" type files, any size.

RESTORE.CMD a special program to restructure copied "random" files for copying, or recopying back to the host system.

FREELINK.CMD a "bonus" utility that "relinks" the free chain of floppy or hard disk thereby eliminating fragmentation.

Completely documented source files included.

ALL 4 Programs (FLEX, 8" or 5") $99.50

Southeast Media

XDATA

A COMMUNICATION Package
for the UniFLEX Operating System

Allows UniFLEX Based Systems to Transmit and Receive files to and from other Computer Systems via Modem. Use with CP/M, Main Frames, other UniFLEX Systems, etc.

-- Verifies Transmission Integrity using checksum or CRC
-- Automatically Re-Transmits bad blocks
-- Transmits data in 128 byte blocks

U - $299.99

Southeast Media

OS-9 VDisk

Give your OS-9 Level I System the speed of memory access that can be several orders of magnitude over your present floppy disk drive. Use that Extended Memory capability of your SWTPC or Gimix CPU card (or any other that has the same format DAT). The size of the Virtual Disk is completely variable in whole increments of 4K up to 960K, which is all that these systems can address beyond the base page that OS-9 Level I uses. By putting all of your CMDS Directory on your Virtual Disk, you can have the fastest execution speed possible (next to eating up System Memory with all of them). You can also set up high speed inter-process communications via random virtual disk files and not eat up valuable system memory with pipe buffers. Some Assembly Required - Level I ONLY.

O, obj. only - $79.95
w/ Source - $149.95

Lucidata

PASCAL UTILITIES
Requires LUCIDATA Pascal ver 3.

XREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

F and CCF - $25.00

INCLUDE -- allows the inclusion of other Files in a Source Text; has unlimited nesting capabilities. Also allows Binary File inclusions.

F and CCF - $25.00

PROFILER -- produces an Indented, Numbered, "Structogram" of a Pascal Source Text File. Allows viewing the overall structure of large programs, and pro id s clues as to the integrity of the program. Supplied as Source Code; requires compilation.

F and CCF - $25.00

VISA MasterCard

** SHIPPING **
Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

TOLL FREE 1-800-338-6800 SOUTH EAST MEDIA 5900 Cassandra Smith Rd. Hixson, TN 37343 info (615) 842-4601 CoCo OS-9™ FLEX™ SOFTWARE

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

*FLEX is a trademark of Technical Systems Consultants
**OS9 is a trademark of Microware

!!! Please Specify Your Operating System & Disk Size !!!

Lucidata

COPYCAT
Pascal NOT required

Allows reading TSC Mini-FLEX, SSB DOS68, and Digital Research CP/M Disks while operating under FLEX 1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform Miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Includes Utilities to List Directories, Copy Files, and convert Text Files when required. Also includes a Utility for Investigating Physical Compatibility problems. Programs supplied in Modular Source Code (Assembly Language) to make it easier to solve unusual problems.

F and CCF 5" - $50.00
F 8" - $65.00

Computer Systems Consultants

FLEX DISK UTILITIES

Eight (8) different FLEX Utilities that should be a part of every FLEX Users Toolbox; Assembly Language (Source Code):

Copy a File with CRC Errors, so it can possibly be salvaged; **Test Disk for errors**; **Compare two Disks**; a fast **Disk Backup Program**; **Edit Disk Sectors**; **Linearize Free-Chain** on the Disk; **print Disk Identification**; and **Sort and Replace** the Disk Directory (in sorted order).

PLUS

Ten BASIC Programs to:

A **BASIC Resequencer** with EXTRAs over "RENUM"; works with ALL Versions of FLEX BASIC AND the Precompiler, checks for missing label definitions, processes Disk to Disk instead of in Memory.

Compare, Merge, or Generate Updates between two BASIC Programs, **check BASIC Sequence Numbers**, compare two unsequenced files, and 5 Programs for establishing a **Master Directory** of several Disks, and sorting, selecting, updating, and printing paginated listings of these files.

A **BASIC Cross-Reference** Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in **TSC BASIC, XBASIC, and PRECOMPILER** BASIC Programs.

ALL Utilities include Source (either BASIC or Source Code). An EXCELLENT Value!

F and CCF - $90.00

BUSINESS
WORD PROCESSING

Windrush Micro Systems

SCREDITOR III

EXTREMELY Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; EXCELLENT Documentation (over 300 pages), including a full Tutorial Section to help you learn how to use the system. Features Cursor-based editing, dynamic Screen Formatting (what you see is what you get), Multi-Column display and editing, "decimal align" columns (AND add them up automatically, if wanted), define multiple keystroke macros, even and odd page number headers and footers, imbed printer control codes in text, full justification series of commands, full "help" support, store common command series on disk for future use, etc. Easy "Set-Up" (for example, you just hit the key you want to use for a specific function, such as "cursor up", and the System reads an stores that key - no digging into tech manuals for codes, etc.); use supplied "set-ups", or remap the keyboard to what you are used too. Except for proportional printing, this package will DO IT ALL!

6800 or 6809 FLEx or SSB DOS, OS-9 - $175.00

Southeast Media

SPELLB "Computer Dictionary"
OVER 120,000 words!

No more "Let your fingers do the walking through the Dictionary" while you are entering Text with your favorite Editor or Word Processor. SPELLB is more than just "another Spelling Checker"; it allows you to **look up a word** from within your Editor or Word Processor so that you **KNOW** it is right WHEN YOU TYPE IT IN with the SPH.CMD Utility (which operates in the FLEX Utility Space). Yes, it ALSO allows you to check and update the Text after you are finished; along with allowing you to ADD WORDS to the Dictionary, "Flag" questionable words in the Text for evaluation later, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of **Small Disk Storage** systems.

F and CCF - $129.95

Great Plains Computer Co.

STYLOGRAPH

A full-screen oriented WORD PROCESSOR -- (now runs on the **Data-Comp** and FHL Color FLEX Systems; uses the 51 x 24 Display Screens). Full screen display and editing (i.e., what you see is what you get); supports the Daisy Wheel proportional printers.

SPECIAL CCF and CCO - $99.00
F and O - $295.00 U - $395.00

SPELL

Fast Computer Dictionary.
F, CCF, OS/9 - $125.00 U - $175.00

MAIL MERGE

Greatly extends the power and flexibility of STYLOGRAPH.
F, O - $145.00; SPECIAL CCF and CCO - $125.00; U - $175.00

Southeast Media

JUST
Text Formatter

JUST, a Text Formatter developed by Ron Anderson, provides numerous features which make it a valuable addition to any FLEX Users Software Library. JUST is designed for formatting Text Output for Dot Matrix Printers and provides many unique features:

- Output the "Formatted" Text to the Display for format analysis and change.
- Output the "Formatted" Text to a Text File for use with the supplied FPRINT.CMD for producing multiple copies of the Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (this Utility is very useful at other times also, and worth the price of the program by itself).
- "User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); provides for up to ten (10) imbedded "Printer Control Commands", such as Italics on and off, boldface on and off, etc.
- Automatic compensation for a "Double Width" printed line.
- Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc.
- Use with ANY Editor.
- Supplied with "Structured Source" (Windrush PL/9); easy to see the flow of the program.

F and CCF - $49.95

** SHIPPING **
Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

!!! Please Specify Your Operating System & Disk Size !!!

FREE DISKETTE WITH EVERY $50 PURCHASE

TOLL FREE TELEX 510 414 PVT BTH

1-800-338-6800
For Ordering

SOUTH EAST MICRO

5900 Cassandra Smith Rd.
Hixson, TN 37343

for information
call (615) 842-4601

CoCo OS-9™ FLEX™

SOFTWARE

DATA BASE MANAGEMENT SYSTEMS

Westchester Applied Business Systems
XDMS

Possibly one of the most powerful Database Management Systems' available, this machine language program is small enough to operate on a single sided 5" disk, yet provides the speed of M.L. and power limited only by the user's imagination. This DMS supports Relational, Sequential, Hierarchical, and Random Access File Structures, and has Virtual Memory capabilities for those Giant Data Bases. XDMS Level I provides a functional "entry level" System which provides for defining a Data Base, entering and changing the Data, and producing Reports. XDMS Level II adds the POWERFUL "GENERATE" facility which uses an English Language Command Structure in manipulating the Data to create new File Structures, Sort, Select, Calculate, etc. XDMS Level III adds several special "Utilities" which provide additional ease of working with the various structures, changing System Parameters, etc.

XDMS Lvl I - F & CCF - $129.95
XDMS Lvl II - F & CCF - $199.95
XDMS Lvl III - F & CCF - $269.95
XDMS System Manual only - $24.95

The Virginia Company
BIZPACK

BIZPACK is used for storing accounting, numeric, and financial data which can then be used for planning, budgeting, forecasting, analyzing, etc. While "Electronic Spreadsheets" are extremely useful in many situations, BIZPACK excels in businesses where there are numerous expense columns, revenue sources, significant business indicators, large numbers, erratic week-to-week and month-to-month fluctuations, etc. BIZPACK helps determine statistical relationships, establish trend lines, "smooths" data via moving averages, analyze seasonal data, adjusts for inflation, lags data in Statistics or Column functions, plots data, etc. BIZPACK is oriented toward time series analysis of businesses. The Program displays information on the screen in Columns of Information with each Row conforming to a defined Period of Time (weeks, months, years, etc.), and is very easy to use (data is easy to enter, change, and modify; commands can be renamed to suit the users requirements; unlimited ability to create specialized commands using common BASIC Statements; etc.). Requires TSC's Extended BASIC.

F and CCF - $135.00
with Source - $250.00

Computer Systems Consultants
TABULA RASA SPREADSHEET

TABULA RASA is similar to DESKTOP/PLAN and provides for the generation and maintenance of tabular computation schemes often used for analysis of business, sales, and economic scenarios. Its menu-driven user interface provides these capabilities even to those users with no programming experience. Its extensive report-generation capabilities allow the user to generate professional results with minimum effort. It requires TSC's Extended BASIC.

F and CCF - $100.00, U - $200.00

Computer Systems Center
DYNACALC

THE Electronic Spread Sheet for 6809 Computer Systems. An extremely POWERFUL Business Tool, this Program will find an unlimited number of "non-business" applications, also (for example, a Full Junior College Electronics Curriculum was set up using DYNACALC). Advanced features like "Table Lookup" make Income Tax work easy; Column or Row Sorting for numerous applications; etc. Completely "Memory Resident", Machine Language, this Program is FAST. Provides STANDARD FLEX Text File output for use with BASIC, Word Processors, Pascal, "C", etc. Also available for Data-Comp and FHL FLEX systems using the 50 x 24 Displays.

F and SPECIAL CCF - $200.00
U - $395.00

ACCOUNTING PACKAGES

Great Plains Computer Co. and **Universal Data Research, Inc.** both have Business Packages written in TSC XBASIC for FLEX. CoCo FLEX, and UniFLEX ----

- - - - Call 800-338-6800 for more information - - - -

Computer Systems Consultants
FULL SCREEN INVENTORY/MRP

The Full Screen Inventory System provides a means of maintaining small inventories. Using a linked, keyed random file structure based upon the item field, it keeps the file in alphabetical order for easier inquiry. With the FIND command, the user may locate and/or print all records matching on partial or complete item, description, vendor, or attributes. Items in backorder or below minimum stock levels may be located and/or printed thru the same process. Printed output may be produced in item or vendor order. A materials requirement planning (MRP) capability for manufacturing environments is included to allow the maintenance and analysis of Hierarchical assemblies of items in the inventory file. It requires TSC's Extended BASIC.

F and CCF - $100.00, U - $150.00

ODDS AND ENDS

Computer Systems Consultants
FULL SCREEN FORMS DISPLAY

This package supports any Serial Terminal with cursor control of Memory-Mapped Video Displays. The package substantially extends the screen input/output capabilities of TSC's Extended BASIC programs by providing a simple, table-driven method of describing and using full screen displays. These table entries are easy to set up and maintain, and are normally stored on disk and read as required. A simple, interactive means of generating the forms and the data field definitions is provided.

F and CCF - $50.00, U - $75.00

Computer Systems Consultants
FULL SCREEN MAILING LIST

The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Using a random fill structure based on the first character of the name field, it maintains the file in alphabetical order for easier inquiry. With the FIND command, the user may locate all records matching on partial or complete name, city, state, zip, or attributes. Printed listings and output to labels may also be produced on the same selective basis. It requires TSC's Extended BASIC.

F and CCF - $100.00, U - $110.00

VISA MasterCard

** SHIPPING **
Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

SOUTH EAST MICRO
5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601
CoCo OS-9 FLEX
SOFTWARE

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

!!! Please Specify Your Operating System & Disk Size !!!

Southeast Media — **RAPIER**

CHESS 6809

Requires FLEX and DISPLAYS On Any Type Terminal

Features:

*Four levels of play.
*Swap side. *Point scoring system.
*Two display boards. *Change skill level.
*Solve Checkmate problems in 1-2-3-4 moves.
*Make move and swap sides. *Play white or black.

This is one of the strongest CHESS programs running on any microcomputer, estimated USCF Rating 1600+ (better than most 'club' players at higher levels).

F and CCF - **$79.95**

Southeast Media

DIET-TRAC Forecaster

DIET-TRAC Forecaster is an XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate. Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual.

Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. When a weight goal is given (either gain or loss), and a calorie plan is agreed upon between the computer and the individual, the number of days to reach the weight goal is projected. The starting and ending rate of weight loss is calculated, and a daily calendar with each day's weight for a 30-day period is printed.

F - **$59.95**
U - **$89.95**

COLOR COMPUTER SOFTWARE

Stearns Electronics

FORTH

Intrigued by Forth??? Here is a FORTH package tailored to the Color Computer! This package is supplied on Tape, with instructions for transferring it to disk if you wish. Written primarily in machine language, it's **speed is unparalleled.** A full Semigraphic-8 Editor is provided, along with "goodies" like Graphics and Sound Commands, Printer Commands, Auto-Repeat and Control Keys, etc. If you are interested in Learning FORTH, a Trace Feature is provided which is invaluable. If you are a FORTH Pro, this package provides CPU carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. (Or; you won't "out grow" the Basic capabilities of this Implementation). Combine this package with Leo Brodie's EXCELLENT Book "Starting FORTH", and you will be a FORTH Expert before you know it (and have a lot of fun doing it!).

Color Computer TAPE - **$58.95**

Custom Software Engineering, Inc.

Color Computer **GRAPHIC SCREEN PRINT** Programs

Dumps any "PMODE" Screen to the Printer with the BASIC USR Function. Shift the Printout Left or Right or Reverse Print (Dark for Light Screen and Vice Versa). All Programs on Tape.

GSPR for R.S. LP-VII/VIII & OMP 100/200/400	**$7.95**
GSPRE for Epson w/ Graftrax and Graftrax +	**$9.95**
GSPRG for Gemini 10 and 15	**$9.95**
GSPRP for the Prowriter Printers	**$9.95**

Custom Software Engineering, Inc.

DATE-O-BASE CALENDAR Program

A Menu Driven **EXTENDED BASIC** Program which allows the entry of up to 12 Memos per Day, each of which may contain up to 28 Characters, for any day of the Month between the years 1700 and 2099. A **Graphic Calendar** shows which days contain Memos, and a "Key Word" Search is provided which can be output to the Screen or Printer.

TAPE DATE-O-BASE CALENDAR
(Each Tape File will hold up to 400 Memos) **$16.95**
DISK DATE-O-BASE CALENDAR
(4,000 Memos at 300/Month per Disk) **$19.95**

FREE DISKETTE WITH EVERY $50 PURCHASE

TOLL FREE 1-800-338-6800 For Ordering
TELEX 998 414 PVT BTN

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601

CoCo OS-9™ FLEX™
SOFTWARE

Custom Software Engineering, Inc.

That's INTEREST-ing

Interested in INTEREST (the Money kind)? An EXTENDED BASIC Program that will help you deal with numerous problems requiring interest calculations. Present Value, Rate of Return, Current Bond Yield and Rate of Return to maturity, Loan Repayment Amortization Schedules, etc.

TAPE - **$29.95**

Custom Software Engineering, Inc.

DISK DATA HANDLER 64K

An EXTENDED BASIC Data Management System w/ Mach. Lang. Routines. Allows a max of 246 Chars. and 14 Fields per Record, and another Record can be linked to the first: 8 Char. Field Names, up to 99 Chars. per Field. Powerful On-Screen editor for input and update, flexible Output capabilities including output to Disk Files for use by other Programs. Change File Definition without re-entering the Data, Split Files, etc. Allows Multiple Field Sorts, Select on any combination of Fields, etc. An extremely POWERFUL TOOL; instructions provide examples of Mailing Lists and a Financial Stock Profit and Loss Tracking System.

DISK - **$54.95**

Custom Software Engineering, Inc.

DISK DOUBLE ENTRY

DISK EXTENDED BASIC Accounting Program w/ Mach. Lang. Routines. A "Traditional" Accounting Package for Small Business, Clubs, Churches, Personal Use, etc. Up to four levels of subtotals with Trial Balance, Income Statement, and Balance Sheet Reports. DDE allows up to 300 accounts and a Trial Balance of $9,999,999.99. Transactions may be up to 14 lines long, and comments and explanations may be freely used. Accounts are traceable to the journal transaction, which may include comments. Screen reports allow review of past transactions and current balances.

DISK - **$44.95**

This Months STAR SPECIAL

K-BASIC & OF both for the normal price of K-BASIC alone:

$199.95

So you save about $80.00, and that is a bargain! This offer is not retroactive, and will conclude April 15, 1985, so I am told by the folks at S.E. Media and Lloyd I/O.

VISA MasterCard

** SHIPPING **
Add 2% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

TOLL FREE 1-800-338-6800 For Ordering
SOUTH EAST MEDIA
5900 Cassandra Smith Rd. Hixson, TN 37343
info (615) 842-4601
CoCo OS-9™ FLEX™ SOFTWARE

Availability Legends —
F = FLEX, CCF = Color Computer FLEX
O = OS-9, CCO = Color Computer OS-9
U = UniFLEX
CCD = Color Computer Disk
CCT = Color Computer Tape

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

!!! Please Specify Your Operating System & Disk Size !!!

object code file. If the object code file is older than the source, MAKE forces the compiler to re-compile. It assumes you have changed the source code and it needs to be re-compiled. In a recent issue of 68' MJ, Peter Dibble (OS9 USER NOTES) wrote about his version of MAKE. It's a good idea. I have implemented two utilities that compare a file's last modified date, one to another file, and the other to a specific date or "today's" date, by default. These work very well in managing file updates or backups. The utility YOUNGER compares two file dates. The first file name is sent to standard output when it is younger than the second file. The utility DRAFT compares a single file name to a specific date, older date, or newer date. If a date is not given, the current system date (hours and seconds deleted) is used for an exact match. DRAFT sends the file name to standard output when a true condition is met. Both of these utilities can get file names from standard input for use with pipes and a file search utility like SDIR.

Many of these utilities can be connected with a series of pipes to form a very complex file manipulation procedure. Another benefit is the ability of SDIR to use any OS9 command. One such use is with the ATTR utility. ATTR is used to examine and change a file's attributes. Unfortunately, it can handle only one file at a time. SDIR has an option to generate SHELL commands that can be piped to SHELL for execution. The following example enables the public read attribute for all the files in the current data directory:

```
SDIR . -scp 'ATTR' 'pr' ! (x -p)
```

SDIR's output is piped to SHELL using (x -p) which causes shell to abort on errors and suppress the "OS9:" prompt. If the current data directory only contained two files, one named 'hello' and the other 'prime', the following lines would be sent to SHELL:

```
t
tmode .1 -pause
load ATTR
ATTR hello pr
ATTR prime pr
unlink ATTR
tmode .1 pause
-t
```

SHELL would then call ATTR which in turn would enable the public read attributes of the files 'hello' and 'prime'. The option 's' tells SDIR to include the 't' and 'tmode' commands. The 'c' option says to generate commands using the command parameter. The 'p' option makes SDIR put the command parameter 'pr' after the file name.

SDIR sells for $35 and comes free with all LLOYD I/O products for OS9 (those selling for $100 or more.) Following is a complete description of the SDIR utility:

SYNTAX: SDIR [-cdempst?]
<directory>
[<'p1'> <'p2'> <'p3'> <'p4'>]
[<match list>]

FUNCTION: SDIR is used to generate pathlists to standard output. In its simplest form, SDIR can list all the files in a directory, one file name per line. The only parameter given in this case would be the directory name to search, such as the current directory '.' (dot). Example:

```
SDIR .
```

The next simplest would be a search for one or more files using their exact names. Example:

```
SDIR . myfile mynames
```

The next type of search would use wild cards in the match list. Wild cards replace parts of file names. SDIR has two types of wild cards:

? is the wild card character which is used to replace any single character in the match.

* is the wild card field which is used to replace any group of characters with limited use.

There are four general types of match lists:

c an exact match
*c wild card field ending in an exact match
c* exact match ending with a wild card field
c floating exact match any where in the names

where 'c' is one or more characters including the wild card character '?'. If no match lists are given, all files will be listed. Examples:

```
SDIR . MY*      find files beginning with 'MY'
SDIR . *ME      find files ending with 'ME'
SDIR . ???      find files that are 3
                characters long
SDIR . *il*     find files with 'il'
                somewhere in name
SDIR . *??il*   find files with 'il'
                somewhere in name and have
                two characters preceding the
                'il'
```

This is how SDIR searches using the match lists. SDIR reads each entry in the directory specified, and compares it to the first match in the match list. If the match is true, the file name is listed to standard output. SDIR continues through the directory until the end, then repeats the process starting on the next match in the match lists. This process continues until all matches have been searched.

If the option -c is used, <'p1'> must be used. <'p1'> becomes the name of a command written before the file name. <'p1'> may include command options. This is useful with all OS-9 commands that do file manipulation. The output stream includes a "load <'p1'>" at the beginning and a "unlink <'p1'>" at the end. This means the command won't have to be loaded each time it is used when the output of SDIR is piped to SHELL using '! (x -p)'.

```
Examples:       'attr'    'ilist -mn'
```

If the option -d is used, <'p2'> must be used. <'p2'> becomes a directory name that is used with each file name found, to make up a second path name listed to the right of the file name. It can then be utilized with commands requiring two path names such as COPY, ICOPY, and PACK.

```
Examples: '/d0/cmds'
          'files/names/customers/products'

SDIR -cd . 'copy' '/d0/cmds' myprogram

gives:    load copy
          copy myprogram /d0/cmds/myprogram
          unlink copy
```

If the option -e is used, all files found will be listed unless it matches with the match list. This is useful when it is desired to operate on most files in a directory except for a few. This option compliments the search logic; meaning that each entry in the directory is checked against the matches, and if a match is made, the file name is not listed.

If the option -m is used, option -c and <'p3'> must be used. <'p3'> is a set of options used to generate the shell I/O redirection and memory size modifiers for the commands generated when the -c option is used. <'p3'> consists of up to four sub-options as follows:

```
i=<path> generates '<path'   for input
         examples: i=/d1/names  i=files
o=<path> generates '>path'   for output
         examples: o=datafile   o=newnames
e=<path> generates '>>path'  for error
         examples: e=errorfile  e=/p
s=<size> generates '#<size>' for the memory
         examples: s=64 s=31k s=14k

Example:  'i=/d1/names o=/P e=errfile s=64'
```

If the option -p is used, <'p4'> must be used. <'p4'> becomes a list of parameters written out following all other parameters and path names. It is written out exactly as it appears in the string.

```
Example:  'pe e pr r w'
```

If the option -s is used, the option -c should be used. It writes out the following:

```
          t
          tmode .1 ~pause
          ..................normal output
          tmode .1 pause
          -t
```

This is useful when the output of SDIR is piped to SHELL using the format:

```
SDIR <options and parameters> ! (x -p)
```

If the option -t is used, the files matching the match lists will NOT be opened to check the user's ability to access a file. Normally, SDIR opens the files it finds matching, and immediately closes them as a general check to see if they can really be accessed. Note that this is a good check, but in some cases not a complete check.

If the option -? is used, SDIR will list its command syntax and options.

This command is intended for use with pipes and the commands: IDEL, ILIST, ICOPY, PACK, and any other command that can take its pathlists from standard input.

Last notes: SDIR will not search any directory other than that specified in the command line parameters. This means sub-directories will not be searched. Although the options can be given in any order, the parameters <'p1'>, <'p2'>, <'p3'>, and <'p4'> must be given in the proper order if they are used. Missing or improper termination of the parameters causes SDIR to display an error message and its command syntax.

EXAMPLES:

```
sdir .
```

lists all files in the current data directory

```
sdir test/cc hello
```

checks for the existence of the file 'hello' in the directory test/cc

```
sdir -t /h0/cmds i??* >myfiles
```

writes the names of all files and directories to the file 'myfiles' that begin with 'i' and are at least 3 characters long including the 'i'

```
sdir -d . '/d3/text' ! icopy -i
```

copies all files in the current data directory to the directory /d3/text

```
sdir . ! ilist -ipnt >/p
```

print all files in the current data directory using the pagination mode

```
sdir . -scp 'attr' 'pr' ! (x -p)
```

enables the public read attributes on all files in the current data directory.

```
sdir . -scdm 'copy'
                '/d3' 's=31k'  my* ! (x -p)
```

copies all files in the current data directory that begin with 'my' to the device /d3 using 31k of memory on each copy.

The above SDIR description was transcribed directly from the manual for the LLOYD I/O Search and Rescue Utilities Package. I have a two-fold motive for publishing it here: first, to shed a little light on the resources available to OS9 users, and second, (in all honesty) to whet your appetite for our products.

In a future article the BIT SLICER will be delving into the design philosophy of K-BASIC. The research and development of K-BASIC was an adventure in learning and design.

K-BASIC, PATCH, CRASMB 16.32, CRASMB, DO, ED/ASM, and SEARCH and RESCUE UTILITIES are trademarks of LLOYD I/O. OS9 is a trademark of Microware. FLEX is a trademark of TSC.

NOTE: All products listed are available thru:

S.E. MEDIA (see ad this issue) or LLOYD I/O (see Lloyd I/O ad also).

- - -

Software C.Dragon Exchange System

Tom Gilchrist
Brad Taylor
1450 N. Clarence #108
Wichita, Kansas

Every group around seems to have a Bulletin Board System (BBS) or Remote Computer System (like RCPM). The CoCo users have a number of systems they can call and OS-9 users even have their own SIG on Compuserve. But how about those of us who still cling to FLEX. Who has a system with software and information we can use? Well, I have put together the "C.Dragon" software exchange system. It's not a BBS (although there is a private mail system). It's for the exchange of public domain software.

I put the system on the air in December and I have been working the bugs out ever since. The system is for UNIX and FLEX users interested in C code. There are quite a few assembly language FLEX programs as well as C programs which are designed for FLEX. A number of these programs are being written up

for 68 Micro Journal so those of you who don't have modems will be able to get them if they are published.

To use C.Dragon, you need a modem (300 or 1200 baud) and modem software for your computer. There are a number of commercially available programs I have seen advertised in 68MJ. You can also use any of the programs which have been published including MODEM9(+).

You can download source files two ways. First, you can use the "list" command to simply list the desired file. If you have a "capture" mode in your modem program, you can capture and save the source to disk. If you want to make sure you get error free source, you can use the XMODEM protocol transfer of C.Dragon (the UCF program on C.Dragon).

There are a number of data bases on the system for help and general information. The ones which are active as of this date are...

Data Base	Description
HELP	Help with C.Dragon commands.
TOPIC	General information about Dragon.
NEWS	Information about new features, etc.
FLEX	General FLEX information.
UNIX	General UNIX information.
SALE	Items for sale, trade, or wanted.

Anyone can call the system and there are no membership dues. As long as people behave themselves, the system will remain open to everyone. The phone number is:

C.Dragon
316-943-9716
24 hours a day
1200/300 baud

New users will need to "sign in". This is done by entering a "7" <CR> at the login prompt. You are asked for your name, city/state, and a password (real names please!). You are then allowed access to the system. Below is a listing of the subjects found in the "HELP" data base:

Current C. Dragon commands (use command name for subject in help)

COMMAND	DESCRIPTION
bye	Ends a C.Dragon session
cat <file name>	List a text file (same as list)
drive [Drive_Number]	Display and change work drive.
flex [subject]	FLEX information, hints, ideas, etc.
fortune	A UNIX favorite!
help [subject]	This program.
list <file name>	List a text file (same as cat).
lls <lbr_name>	List the contents of a disk library.
ls [drive]	List root files and directories of given disk.
mail [login_name]	Send and receive mail.
more <file_name>	Like list, but controls screen.
news [subject]	Short term information and bulletins.
sale [item]	Items for sale or trade by users.
time	Show the time , date, and a calendar of the month.
topic [subject]	Long term information and topics.
ucf	XMODEM protocol transfer menu driven.
unix [subject]	Unix information, topics, hints, news, etc.
who	Who you are (your login name).
whois [string]	List the users of c.dragon.:

When you log in, you are given a "message of the day" which tells you about the system and current information. To learn how to use the C.Dragon system, you simply read the help, topic, and news data bases. For instance, to find out how to use the "LS" command, type "HELP LS <CR>".

When using C.Dragon please follow the following rules:

1) Try to limit your sessions to no longer than 1 hour.

2) Don't abuse the "mail" system. Send mail only to those you know. There are plenty of public BBS systems around and there is limited mail space on the system.

3) Remember that CTRL S and CTRL Q will turn on and off the scroll (as well as the standard FLEX ESC toggle). If the system seems to hang up, try ESC or a CTRL C.

4) 1200 baud is a little tricky long distance. If you have trouble, try 300 baud.

5) Type "BYE" when you are ready to quit.

Thanks to all those who have donated your software. Thanks to Brad Taylor for all the help with the code and to Kent Meyers for all his helpful feedback. Have fun!

- - -

Editor's Note: This arrived about two week after we quietly removed our BBS. We gained a lot of insight into BBS operations. Unfortunately, the stupid and thoughtless actions of a very few, caused the loss of this service for over 5,000 persons who had logged on while it was operational.

The control person/persons who look after the day to day operation of a BBS are to be congratulated! We spent hours, many hours each week, sometimes each day, screening files for garbage, profanity, filth, etc.

When we started it we knew that there would be some younger (7-12 years of age) logging in. For you older types I guess it would have not been all that much of a thing. For them we tried to keep it clean. Finally, it just became too much - we pulled the plug!

Many nights Bob Nay, or some of the others would sit by the system until the wee hours of the morning, watching incoming files. We were able to trace (or trick) a few in to revealing who and where they were (Ma Bell **STILL** does cooperate, given the right circumstances). One particular nasty fellow lived in Atlanta, but signed in with a fictitious name and a New York address. We actually got a trace on him - he sent looooooong files. Fact is, he was using the company WATTS line. Well, a reader in Atlanta made a personal and somewhat impressive visit to this fellows house. He was running a TRS80-1. No color computer, no 68XX anything, just a modem and the bosses WATTS line. I understand he was over 6 months finding decent employment again. Also he donated his TRS-1 to a charity organization, all on the recommendation of the 68 Micro Journal reader and BBS user (his small daughter also used our system and had ask her Dad what one of those files meant). This was not the only one, others we figured out by various methods and contacted them direct - However, it just became too much.

A couple of old hands at the BBS thing suggested that we just ignore them and they would go away. Long distance and all cost, you know. Well, for some of the lesser ones we did, and they did, but for the more vile ones we could not condone the waiting period - so the 68 Micro Journal BBS is gone!

In looking back I can say that it was a good thing. Most all, 99% of the offenders we could identify were NOT 68 Micro Journal readers. Some were local CoCo users, most of them were never any problem, and in fact contributed some good material to the system. However, for the problems we had, over 80% or so were non-68XX or S50 bus types, as best we could determine. Oh well, I guess it takes all kinds.

So for all of you out there I want to thank you for the material you supplied via our BBS. We placed some there, you did the rest. It was 95% a pleasure to operate and maintain the system, the other 5% made it impossible to continue. But, I guess that is the way of a lot of things in this world.

We still have the hardware, the phone number and line are still there, maybe someday we can get it going again, if and when we can figure out some way to prevent it from being as before.

Just thought some of you might like to know.

DMW

P.S. 68 Micro Journal will be happy to publish programs from the system - provided they do not infringe on someone elses rights. So if you contribute to the system, let us have a copy (media FLEX or OS-9) and we will attempt to let all **68XX** users know.

Might just make you world famous!

- - -

Diskette Inventory

Francis MASSEN
8 Cite Strauss
L-9357 BETTENDORF/ Luxembourg, Europe
tel: 808021

DISKETTE INVENTORY: THE BARE-BONES SOLUTION

Until today, I always used the price-winner program FILESORT, by Brian BAILY, published in the May 1981 issue of 68 MICRO JOURNAL to make a disk inventory. (hey DON, no next GREAT CONTEST looming around?). ** Maybe soon - anyone have anything for a GOOOD contest????

This marvelous program makes a complete inventory of the catalogues, and gives a lot of options to rearrange, delete or sort the resulting file. Alas, all that work is done in RAM-area, and with over 250 diskettes to look after, you will soon crash into a 'buffer overflow'!

In November 1984 Tom Weaver published another inventory program, doing essentially the same job, albeit offering not many (in fact none) options.

Here I want to suggest a bare bones method, enabling you to create an unlimited file of your catalogues, in a standard FLEX .TXT format, so you can easy edit and sort it later on.

You have to use 4 programs:

1. the LOG utility by N.Yssel, published in the November 1984 issue of 68 MICRO JOURNAL
2. the standard FLEX DIR.CMD
3. a little BASIC program from myself, named BUILDCAT
4. the standard FLEX SORT/MERGE program

Now here is what you have to do to create that complete inventory of your disks:

1. load and run the LOG.CMD typing for instance

   +++LOG 1.CAT84

2. using the DIR.CMD, look at every diskette you wish to inventiorate; all these DIR's will be logged into the file CAT84.LOG
3. having finished quit the logging action by typing

   +++LOG OFF

4. You now have on disk a catalog file, containing many lines with unwanted and unnecessary informations; run the basic program BUILDCAT to strip these off and create a catalog file containing only the file-name, disk-name-number, date of creation. I kept this little basic program intentially very primitive, to make it transparent and allow easy modifications.
5. Order the resulting .TXT file with the SORT/MERGE program.

That's all! Really, not a strike of genius, but a very simple and reliable method, and a demonstration of the usefullness of the LOG.CMD !

Here is that basic program BUILDCAT:

```
 5 REM*** Francis Massen    dec. 1984
10 REM*** MAKE A CATALOG FILE USING THE .LOG-
FILE
20 REM--
22 REM*** THIS PROGRAM IS TO BE USED TO BUILD
A FILE CONTAINING ALL 22
REM*** THE FILE-NAMES OF THE VARIOUS
CATALOGUES
23 REM***
40 REM--
90 GOSUB 2000
100 INPUT"READ FROM FILE.............",R$
110 INPUT"WRITE TO FILE.............",F$
120 PRINT:PRINT:PRINT
200 OPEN OLD R$ AS 1
210 OPEN NEW F$ AS 2
220 ON ERROR GOTO 10000
300 INPUTLINE#1,A$
310 IF (A$="") OR (A$=" ") THEN 300
320 IF LEFT$(A$,3)="DIR" THEN 300
330 IF LEFT$(A$,3)="+++" THEN 300
340 IF LEFT$(A$,5)="FILE#" THEN 300
350 IF MID$(A$,5,5)="FILES" THEN 300
360 IF LEFT$(A$,4)="DISK" THEN GOSUB 700:
GOTO 300
368 N$=MID$(A$,8,12)    :REM*** name of file
370 T$=MID$(A$,44,10)    :REM*** date of
creation
379 REM*** Z$ contains Name.ext + disk-name +
date
380 Z$=N$+"  "+D$+"  "+T$
390 PRINT Z$
400 PRINT#2,Z$
410 GOTO 300
700 REM-----------------------------------
710 REM*** extract disk-name and disk #
720 D$=MID$(A$,7,8)
730 I=INSTR(2,D$," ")
740 IF I=0 THEN 800 :REM disk-name is 8
caracters              long, so ok...
750 REM*** pad with spaces until 8
760 D$=MID$(D$,1,I-1) : REM*** this is name
770 FOR J=I TO 8
780 D$=D$+" "
790 NEXT J
800 REM*** look for disk number (begins with
a #)
810 I=INSTR(1,A$,"#")
820 D1$=MID$(A$,I,6)
830 D$=D$+"  "+D1$
890 RETURN
2000 REM--
2001 PRINT CHR$(27);CHR$(69);:REM*** CLS ON
JR19
2010UPRINT"**********************************
*******"
2020PRINT"    C A T A L O G   B U I L D E R
"
2030PRINT"*****************************************
*******"
2040 PRINT
2050 PRINT"This program takes a LOG-file
containing the DIR of the diskettes"
2060 PRINT"It strips the meaningless
informations, and builds for each file"
2070 PRINT"a line with FILENAME  DISK-NAME-
NUMBER - DATE"
2080 PRINT"The resulting file may then be
sorted by SORT-MERGE"
2090 PRINT
2100 RETURN
3000 REM--
10000 REM*** ERROR HANDLING
10010 IF ERR=8 THEN 10100
10020 PRINTCHR$(7);"FATAL ERROR<<<<<<<<<<<<<"
10030 STOP
10100 REM--
10110 PRINTCHR$(7);:PRINT:PRINT
10120 PRINT"Catalog-file finished!!"
10130 PRINT
10140 CLOSE 1:CLOSE 2
10200 END
10210 REM--
```

Pleasant PL/9

By: Lane Lester

In comparing FLEX to the other operating systems available for 6809-based computers, it seems that "Friendly FLEX" sums up pretty well the differences. The English-like and uncluttered nature of its command lines appeal to all but professional cryptographers and those who yearn for the days when the use of computers was a secret art available only to the few. As you can see from the title of this column, PL/9 is a language that provides a similar attraction. I'd like to introduce you to some of the features that make PL/9 so pleasant to use.

PL/9 is a procedural language like C and Pascal. In fact, the easiest way to describe PL/9 is to say that it's a cross between those two languages. The programmer has the same freedom that C provides, plus the English-like and self-documenting appearance of Pascal. To put it in a negative way, PL/9 does not impose on the programmer the heavy constraints of Pascal, nor does it have the cryptic appearance of C code. About the only thing I don't like about PL/9 is its name. When you tell someone about it they say, "Oh, yes, like PL/I."... whatever that is. I wish that Graham Trott had called it Galileo or something else that doesn't generate preconceived notions.

My first micro was a Color Computer, and I've been spoiled by the interpreted, ROM-based BASIC. For me, one of PL/9's best features is that its compiler and excellent line editor are in memory at the same time. I'm too sloppy a programmer to put up with separate editing, compilation, and in some cases, assembly, for every undeclared variable, missing ";", and typographical error. When compiling a PL/9 program and an error is encountered, the offending line is displayed with an arrow pointing to the place where the compiler ran into trouble, and an error message is displayed. The editor can imm diately be used to correct the problem and the compile command re-issued. For a more complex problem, a debugger program can be loaded from disk and the program run within its very powerful facilities. I wasn't always this sloppy and lazy. When I was learning FORTRAN in the late 60's by keypunching the programs and submitting the card decks to the high priests in Purdue's computer center, I tended to be much more careful.

One feature of PL/9 that I really appreciate is in the area of variable names. Meaningful names do a lot to make programs easier to read, and early BASICs with their one and two letter restrictions were most unhelpful in this regard. Microsoft's Color BASIC at least allows long variable nam s, but difficulties arise from the fact that the interpreter considers only the first two characters. A gasoline mileage program I wrote originally had PRICE and PRESENT as variables, which did not work well at all. Microsoft did upgrade their BASIC for the IBM PC by increasing the name size to 16. PL/9 allows names up to 127 (!) characters long and all are actually part of the nam . Case is ignored, so this gives you the freedom to use all upper, all lower, or a mixture, according to your whim. The availability of the underline character encourages multi-word names for added readability, e.g., LOAD_DATA.

If you're starting out to learn a language like BASIC, there are all sorts of help available for beginners. Radio Shack's Color BASIC manuals border on the infantile. CoCo magazines provide a continual supply of tutorial articles in the language. And in most towns there will be a group of CoCoNuts who help each other master the capabilities of the language. But the situation changes drastically if you move to something like Pascal, C, Forth, or PL/9. Then you're pretty much on your own. Most manuals that come with these languges have little or no tutorial mat rial, so you have to head off to B. Dalton's to get some help. You'll find books written for beginners for all of the languages... except PL/9!

Not very encouraging, you say. That would indeed be the case, were it not for the excellent manuals and sample programs that come with the package. Graham Trott has done an outstanding job of providing an introduction to the language for the beginner, plus a useful collection of reference material for the old hand. The 231-page users guide is "aimed at the complete newcomer" and provides quite an education in structured programming. Sample program fragments illustrate the explanations of each feature of the language. At the end of the users guide are 40 pages of example programs which provide further illustration of the way that PL/9 programs are constructed. The 193-page technical reference manual provides convenient access to information on the editor, compiler, debugger, and PL/9 language. At the end of the reference manual are the descriptions and listings of the library programs, which are also included on the PL/9 disk. These are included, as needed, in PL/9 programs to provide I/O with the keyboard, screen, printer, and disk. Also included are programs for string functions, real number manipulation, scientific functions, and a Shell sort.

There is plenty more that I could tell you about this fine system, but let's look at an actual PL/9 program, since that's one of the best ways to see what a language is like. About 803 of my computing involves word processing, so an initial project has been to convert my Color Disk BASIC Telewriter files to the FLEX ASCII format. Telewriter stores text in binary, so two types of conversion are necessary: binary to ASCII code and Disk BASIC to FLEX file formats. There were a number of ways this could be accomplished, but I ended up using a program I already had to move the binary file to FLEX, and then I wrote the following program to convert from binary to ASCII. CONVERT.CMD does two things that make it easier for me to edit the files with my current editor, DynaStar. First, of course, is the binary to ASCII conversion, but also helpful is that whereever a Telewriter control character is encountered it is replaced by a "", which is the control character used by JUST, the text processor I'm now using.

This program comes under the the heading of "quick and dirty". It's certainly no thing of beauty, just something thrown together to get a one-time job done. In fact, most of the code is from a program in the PL/9 manual that converts upper case text files to lower case. I just added the necessary stuff to convert Telewriter code to ASCII. As is true of Pascal and C, variables in PL/9 have to be declared as to type: byte (8 bits), integer (16 bits), or real (8-bit exponent and 24-bit mantissa). In Lines 5-6 a number of single variables and vectors (single-dimension arrays) are declared at specific locations in RAM. From Line 5 we see that the 16-bit integer POINTER is at $0000 and LAST is at $0002. Following that is the byte vector NAME at $0004, and the large text buffer BUFFER (how appropriate!) at $0018.

There are three procedures: ABORT, OUTPUT, and MAIN. Execution would start at Line 24, and Lines 25-29 get everything ready for reading a text file. Lines 31-40 read the file into the vector BUFFER. You'll notice what may be a strange loop: REPEAT... FOREVER! This construction is primarily designed for control applications where a routine is repeated as long as the machine is on. But BASIC programmers are no strangers to this sort of thing, as in:

```
10 IF NOT EOF(1) THEN INPUT1, A$(I): I = I + 1: GOTO 10
```

Lines 34-38 serve the same purpose of breaking out of the REPEAT-FOREVER when the end-of-file or some other error condition occurs.

The Telewriter file has the extension .BIN, so Lines 44-46 open a new file with the same name and the extension .TXT. In Lines 49-58 we have the REPEAT-UNTIL, a construction found in other languages. This loop outputs the contents of BUFFER to the new disk file after converting each byte from Telewriter to ASCII. This is done with an IF-CASE group which allows for more alternatives than the IF-THEN. Notice in Line 54 that this feature also has an ELSE which in this case does the most common job of the conversion: adding 32 to each byte. In Line 55 the added bell (or is it a whistle?) of changing the Telewriter control symbol to JUST's "" is performed. Then the converted byte is passed in Line 56 to PROCEDURE OUTPUT for writing to the file.

After the whole file is written, a final CR is sent (Line 59), the new file is closed (Line 60), and the old file is deleted (Lines 61-62). Line 63 is not part of the program and is placed there by the editor.

- - -

```
.la10
/* CONVERT, Converts Telewriter binary files to ASCII */
ORIGIN = $B300;
STACK = 4;

AT $0000: INTEGER POINTER, LAST: BYTE NAME(20),
          BUFFER(10000);

AT $C840: BYTE FCB, ERROR(319);
AT $CC14: INTEGER LINE_POINTER;

INCLUDE IOSUBS;
INCLUDE FLEX;

PROCEDURE ABORT;
REPORT_ERROR(.FCB);
FLEX;
ENDPROC;

PROCEDURE OUTPUT (BYTE CHAR);
PUTCHAR(CHAR);
IF CHAR=CR THEN PUTCHAR(LF);
WRITE(.FCB,CHAR);
IF ERROR THEN ABORT;
ENDPROC;

PROCEDURE MAIN;
GET_FILENAME(.FCB);
IF ERROR THEN ABORT;
OPEN_FOR_READ(.FCB);
FCB(59) = $FF;
IF ERROR THEN ABORT;

POINTER = 0;
REPEAT
   BUFFER(POINTER) = READ(.FCB);
   IF ERROR THEN
      IF ERROR = 8
         THEN BREAK
         ELSE ABORT;
   IF BUFFER(POINTER) = 92 THEN BREAK;
   POINTER = POINTER + 1;
FOREVER;
LAST = POINTER;
CLOSE_FILE(.FCB);

FCB(12) = 'T; FCB(13) = 'X; FCB(14) = 'T;
OPEN_FOR_WRITE(.FCB);
IF ERROR THEN ABORT;

POINTER = 0;
REPEAT
   IF BUFFER(POINTER)
      CASE  0 THEN BUFFER(POINTER) = SP;
      CASE  9 THEN BUFFER(POINTER) = ';
      CASE 94 THEN BUFFER(POINTER) = CR;
      ELSE BUFFER(POINTER) = BUFFER(POINTER) + 32;
   IF !BUFFER(POINTER) > $7D THEN BUFFER(POINTER) = '\;
   OUTPUT(BUFFER(POINTER));
   POINTER = POINTER + 1;
UNTIL POINTER >= LAST;
OUTPUT(CR);
CLOSE_FILE(.FCB);
FCB(12) = 'B; FCB(13) = 'I; FCB(14) = 'N;
DELETE_FILE(.FCB);
```

68' Micro Journal
1984 Index of Articles

Back Issues May Be Ordered
$3.95 Per Issue By Writing or Calling
Toll Free 1-800-338 6800

January 1984

FLEX USER NOTES, Anderson
OS9 USER NOTES, Dibble
A BASIC09 Financial Modeling Program, Bovet
EPSON HX-20 Review, Stark
ACORN MOTHER BOARD Review, Abrahams
CCSM Standard MUMPS Review, Dibble
Converting FLEX to OS9, Pass
Bit Bucket - Help - Classifieds

February 1984

FLEX USER NOTES, Anderson
OS9 USER NOTES, Dibble
CONVERTING UNIFLEX TO OS9, Pass
1983 68' MICRO INDEX of ARTICLES
DYNAMITE REVIEW, Dibble
PACK, Gottschall
WORD.BAS & XON.OVL, Robinson
UNIVERSAL SINGLE BOARD, Leverkusen
OS9 SIG on COMPUSERVE, Bellomo
DISK & RAM DISK TESTING, Johnshoy
UPPER & LOWER CASE CONVERSION, Lanpher
Bit Bucket - Help - Classifieds

March 1984

FLEX USER NOTES, Anderson
OS9 USER NOTES, Dibble
C USER NOTES, Commo
DATA STRUCTURE, Pass
Using MM58167 Real Time Clock, Gross
UNIFLEX General Ledger Review, Bram
SSB DOS Utilities, Fraser
Review of RMA and RLINK, Dibble
MIKMAN to enter MIKBUG, Trollope
MOTOROLA 68000 Software Support, Motorola
"PK" Programable Keyboards, Condon
OS-9 Users Group News Letter, Part I
Bit Bucket - Help - Classifieds

April 1984

FLEX USER NOTES, Anderson
OS9 USER NOTES, Dibble
C USER NOTES, Commo
Where is the 68000, Staff
Programming in C: A Review, Foster
The GIMIX III System, Staff
Remote Pulse Width, Craig & Ziolkowski
James Bellomo Dies
Day Command for FLEX-9, Mills
OS-9 User Group News Continued, Part II
68000 SAGE System, Lord
Budget, Craig
Bit Bucket - Help - Classifieds

May 1984

FLEX USER NOTES, Anderson
OS9 USER NOTES, Dibble
OS9 Users Group, Puckett
Reading Non-Flex Based Disks, Fraser
Modem 68, Moorfoot
Hazelwood DM-256 Review, Staff
Multi-Tasking w/o Interrupts, Troot
Repairing Mini Floopy Drives, Massen
Improvements ? TO TSC "FILES" Utilities, Walters
Bit Bucket - Help - Classifieds

June 1984

FLEX USER NOTES, Anderson
OS9 USER NOTES, Dibble
C USER NOTES, Pass
Rumors & Such, DMW
Reading Non-Flex Disk Continued, Fraser
Disk Speed Measurement, Johnshoy
PC, Korte
Hacking the MAC, Wolf
Bit Bucket - Help - Classifieds

July 1984

FLEX USER NOTES, Anderson
OS9 USER NOTES, Dibble
C USER NOTES, Pass
Rumors and Such, DMW
68000 User Notes, Lucido
Windrush IEEE-488 Review, McDaniel
VDisk Review, Pass
Disassembler, Stock
Reading Non-Flex Disks, Fraser
Metalab Review, Nunn
ELEKTRA Motherboard Review, Sims
Corrections
Bit Bucket - Help - Classifieds

August/September 1984

FLEX USER NOTES, Anderson
OS9 USER NOTES, Dibble
C USER NOTES, Pass
68000 User Notes, Lucido
Reading Non-Flex Disks, Fraser
Communicating with OS/9, Thompson & Pass
Review of S Disk, Bootfix & Filter Kit I, Pass
Disassembler, Stock
SWTPC Prospering Pioneer
Hyperdisk, Hazelwood
Bit Bucket - Help - Classifieds

October 1984

FLEX USER NOTES, Anderson
OS9 USER NOTES, Dibble
C USER NOTES, Pass
CAT Utility for Flex 1 2 & 9, Puglia & Taylor
LETNUM Routine, Gass

Microware Relocatable Assembler & Linker, Pass
68K Experiences, Salzwedel
Cyclic Redundancy Check, Magnus
Music on the 68000, Dildy
KBASIC
Corrections
Bit Bucket - Help - Classifieds

November 1984

FLEX USER Notes, Anderson
OS9 USER Notes, Dibble
C USER Notes, Pass
68000 USER Notes, Lucido
Single Board Computers, DMW
Cobol, Anderson
Data Systems 68 Product, Review
6809 FLEX Diskette Inventory, Weaver
LOG, Yssel
General Purpose Interface Bus, Moore
Bit Bucket - Help - Classifieds

December 1984

FLEX USER Notes, Anderson
OS9 USER Notes, Dibble
C USER Notes, Pass
Single Board Computers, DMW
Computer Excellence Memory Bd, Anderson
68000 USER Notes, Lucido
Turtle Graphics in PL/9, Cole
Software Product review, Ward
Extend a FLEX Dictionary, Fraser
Reading Hard Sectors Disks, Warren
US ALL!, DMW
Documentation, The Necessary Evil, Killebrew Jr.
Development Terminal Program, Hausler
Bit Bucket - Help - Classifieds

CoCo Users
Color Micro Journal Index

Send $1.95 to cover 1st Class Postage
Please Specify Volume and Issue

Or Call To Order
1-800-338 6800

Vol. 1 Issue 1

This 'N That; Introduction-RLN
BASIC Users Notes; Adv. BASIC-Baker
Baby CoCo; Model 10-Beckett
Inside the Dragon 32; a look-Rosen
Communicating with your CoCo; Modems etc-Rosen
Why CoCo Can't READ; Disk Problems?-Pass
Computer TAPES??; Considerations-LaLone
CHECKS/BAS; Home Use Program-Yates
Laser Wars/BAS; Game Program-Rogers
SUPER Lunar Lander/BAS; Game Program-RLN
Doodle Bug; Review-Patrick
Synther 7; Review-Patrick
Rail Runner; Review-Patrick
Megapede; Review-Patrick

Vol. 1 Issue 2

This 'N That; CoCo OS-9-RLN
OS-9 on Color Computer-RLN
Microware Seminar-DMW
CoCo 64K RAM Mod-Abrams
Make CoCo Silver-LaLone
FLEXNET BBS-Rosen
CoCo 2 64K and Disk-RLN
Simple MENU Program; Hints-Brumley
User Def Functions; Hints-Brumley
RAMSCAN-Lewis
Elite°Calc; Review-Lane
Geography Pac-Williams
ROMBACK-Roberts
Wizard 64-Patrick
Multi-Pak Interface-Patrick
FastLane-Hifter
SP-1-Williams
Sands of Egypt-Patrick

Vol. 1 Issue 3

This 'N That; CoCo 2 Disk Controller-RLN
Linking Loader for RS & CoCo FLEX (pt. 1)-Waggoner
Some Basic BASICS about Disk Systems-LaLone
MICROBOOKS; A Tape-Based BASIC Program for Small Business Use-LaLone
COPYNEW; A BASIC Disk Utility-Stark
LANDER; A BASIC "Lunar Lander" Game for 4K-Toscani
GRAPHIC LETTERS; Use a BASIC Subroutine to Display Graphic Letters-Stumpf
PRNTDUMP; a BASIC Program to Dump Graphics to a Paper Tiger-Euton
Tutor; Review-Williams

Vol. 1 Issue 4

This 'N That; OS-9 Users Group-RLN
OS-9 Tips; Useful Operating Hints-Aker
Computer Operting Systems; Part 1-staff
Linking Loader for RS & CoCo FLEX (pt.2) Waggoner
CoCo 2 Hardware Notes; The New Color Computer-Rosen
Where is the Computer NOW??-Pass
BASLIST/BAS; a BASIC "Pretty-Printing" Program-Deal
OOPS!! Clarification of last months MICROBOOKS Discussion-LaLone
X-Y Plots with the MX80; a Basic09 Program-Strunk
The Complete Personal Accountant; Review-LaLone
RDC-1; a CoCo Disk Controller-staff

Vol. 1 Issue 5

This 'N That; Basic09--and the C Compiler-RLN
Converting CoCo BASIC to OS-9 Basic09-Pass
RDC-1 Disk Controller Chip Clarification-Weeks
A TYRO Meets OS-9-Kahn
Linking Loader for RS & CoCo FLEX (pt. 3)-Waggoner
Use the MICRO WORKS 80C DISASSEMBLER with the FLEX DOS; program-Maltby
MEDICAL/BAS; program-Yates
Zaxxon; Review-Roberts
New RS Joysticks; Review-Roberts

Vol. 1 Issue 6

This 'N That; Computers, Software & Assembly Language Programming??-RLN
LEDs-The Readers
OS-9 SIG on Compuserve-Bellomo
Converting CoCo BASIC to OS-9 Basic09 Part II-Pass
Linking Loader for RS & CoCo FLEX (conclusion)-Waggoner
MYDIR, an OS-9 Asmb Lang 'Pipe' Program-Kahn
Copy M/L Tapes; Basic Program-Martinez
4 DRIVES/BAS-Use Double-sided w/RS DOS V1.1; program-Yates
FACE; Review-Garrett

Vol. 1 Issue 7

This 'N That; 6883/6847 Chip Problem, "Help Wanted"-RLN
Using the CoCo; Introduction-Mann
LEDs-The Readers
CoCo OS-9-Dibble
Converting CoCo BASIC to OS-9 Basic09 Conclusion-Pass
OS-9 Tips; "dsave" command-staff
OS-9 Users Group NEWSLETTER

Vol. 1 Issue 8

This 'N That; A "New CoCo" Rumor, RS Disk **Controller Swap and C Compiler Using the CoCo**; Computer Surplus-Mann
OS-9 Users Notes; Big Systems and OS-9 on the CoCo-Dibble
Working with 64K-Kahn
WORD-PAK-80 Cd. Board; Review
Macro-80C; Review-Pass
Repeat Key; Review-Mann
Raster Memory System (RMS); Product Announcement-Motorola

Vol. 1 Issue 9

This 'N That; CoCo DYNACALC and "Help Wanted"-RLN
OS-9 Users Notes; More about the CoCo Disk Driver-Dibble
Using the CoCo; CoCo Disk Driver-Mann
ROM to RAM-Kahn
OS-9 Users Group NEWSLETTER
Using BASIC's "RND" Function with BBLBOOKS/BAS-LaLone
Cookie Monsters LETTER CRUNCH; Review-Mann
MASTER WRITER; Review-Mann
GRAPHICOM; Review-Mann
BABYLON; Review-Patrick

Vol. 1 Issue 10

This 'N That; New Columns on the Way-RLN
Using the CoCo; Soldering for the Newcomer-Mann
Advertiser's Corner
A PRINTR for Basic09-Voigts
TAPE File Transfers-McGarry
ROM to RAM (update)-Kahn
Humor??
Helpful Hint; A "PCLEAR 0" for Ext. Basic & **Disk Ext. Basic**-Brumley
MAGIGRAPH; Review-Mann
ColorSoft SBA; Review-Perotti
DYPIM; Review-LaLone
TEXTPRO III; Review-Brumley

Vol. 1 Issue 11

This 'N That; Book Reviews, Etc.-RLN
Basic09 Basic; IN THE BEGINNING...-Voigts
Using Basic; Introduction-Brumley
INSIDE the CoCo; Keypads & Such-Chapple
Advertisers Corner
Tape File Transfer (pt. 2)-McGarry
CoCo OS/9 RS232 Module-Pass

Vol. 1 Issue 12

This 'N That; A "NEW GAME"??-RLN
Using the CoCo; Function Keys for CoCo-Mann
INSIDE the CoCo, Microprocessor-Chapple
Advertiser's Corner
ML Utilities; A Search of the Perfect Utility-Janecki
Basic09 Basic; No line Numbers-Voigts
Using Basic; Corrections to SDPRO-Brumley
First Step Into OS-9-Dibble
EDIT/BAS-LaLone
TAPE File Transfers (conclusion)-McGarry
Using the RS DSKCON Rountine-McLaughlin
The Peeper 2.0; Review-Mann

Vol. 2 Issue 1

This 'N That; Dragon Computer Update & Interpreters and Compilers-RLN
Using Basic; Sorting-Brumley
Inside the CoCo; Memories-Chapple
Using the CoCo; Letter Replies-Mann
Basic09 Basic; Packing-Voigts
ML Utilities; ORACLE II Cont.-Janecki
MIDITEXT/BAS-Belgard
Floating-Point Arithmetic-Pass
Using the RS DSKCON Routine-McLaughlin
Color Monitor; Review-Janecki
Star Trap; Review-Mann

Vol. 2 Issue 2

This 'N That; A "Bug" Fix, Computer(s) in the Schools-RLN
Using the CoCo; Upgrading CoCo to 64K RAM-Mann
Basic09 Basic; To Error is Human To Forgive Sometimes Takes A Computer-Voigts
ML Utilities; Disassemblers-Janecki
Using BASIC; Color BASIC-Brumley
Inside the CoCo; PIA's-Chapple
CP/M on the Color Computer-Nygren
BlackJack Royale; Review-Mann
Basic09 Tour Guide Book; Review-Voigts

Vol. 2 Issue 3

This 'N That; "Smoked" CoCo 2's-RLN
Using the CoCo; The Magic Pokefile Part I-Mann
Basic09 Basic; Number Crunches-Voigts
Using Basic; Tokenizing-Brumley
ML Utilities; Disassembler Pt. 2-Janecki
Inside the CoCo; More on the PIAs-Chapple
Programmers Calculator; Asmb Lang Program-Lewis
CP/M on the Color Computer; Final Part-Nygren
Fastape; Review-Mann

Vol. 2 Issue 4

This 'N That; Education and the CoCo-RLN
Using Basic; Looping-Brumley
Inside the CoCo; The Powerful "SAM"-Chapple
Educational Forum
Basic09 BASIC; Proceeding w/ Procedures-Voigts
Using the CoCo; The Magic Pokefile Part II-Mann
ML Utilities; Disassemblers P. 3-Janecki
Read RS Disk Directories from OS-9-McGarry
ADDRESS FILE/BAS-Chapple
Computerized Learning Processor; Review-Mann
Multiple Program Processor; Review-Mann
JBM "LdMac" Basic09 Utilities-Voigts

Vol. 2 Issue 5

This 'N That; Where to from here, Tandy?? and a "NEW" CoCo Sorta!!-RLN
Basic09 Basic; Pascal Programming and Basic09-Voigts
Using Basic; A Question of Output-Brumley
Inside the CoCo; The Powerful "SAM" Part 2-Chapple
Building PROTOTYPE Hardware for the CoCo-Hart
LASER/BAS-LaLone
Taxi; Review-Mann
ADOS; Review-Mann
"Lookup/Slookup" Basic09 Utilities-Voigts
Disk Utility 2.0 & 2.1-Mann
96KX-M Memory Expander-Mann
Elite*Word; Review-LaLone
Elite*File; Review-LaLone
OS-9 Text Tools-Kahn
COCODUMP 2.1-Mann
"String Arithmetic" Basic09 Utilities-Voigts

BIT BUCKET

DIGITAL SPECIALTIES
5124 HARTELODGE
ST. LOUIS, MO. 63129
(314) 892-0756

68 MICRO JOURNAL
5900 Cassandra Smith
PO Box 849
Bisbee, TX 79343

Dear Don

While teaching a course in BASIC programming at the local junior college, I found that the lack of a direct method of printing a program listing directly from TSC BASIC was to say the least, very cumbersome. The computers we are using, are all ages and vintages of SOUTHWEST TECHNICAL PRODUCTS. The old boxes have the I/O ports readdressed at $E000, and the ports are on 16 byte boundaries to be compatible with the newer units. All have parallel printer ports at port 7 or they have the newer MP-ID type printer port. All are using SOUTHWEST versions of FLEX, and TSC Extended BASIC.

The following two utilities will work from XBASIC, and can be called while XBASIC is running. They do not, however, require XBASIC and will work with any FLEX program that uses the standard console output vectors.

The first utility "PRON" will cause all output to be sent to the console and to the printer. The second "PROFF" will restore normal operation to the system. PRON checks to see if it is already running to prevent being executed twice and crashing the system. PROFF checks to see if PRON is running before trying to turn it off. If your printer supports "SELECT AND DESELECT" commands, they are provided for in both programs, and page eject is also provided.

These programs are provided for the use of all readers free of charge, but are not to be sold, or included with any other software that is sold for profit.

```
PRON                                                    PAGE    1

                        ************** PRON **************

                        **  PRINTER PASS THRU UTILITY  **

                        * ADDRESS OF PRINTER PIA

                        * PIA AT $DF82 OR MP-ID BOARD IF
                        * CPU TYPE INDICATES THAT MP-ID
                        * IS INSTALLED IN COMPUTER.
                        * OTHERWISE PIA IS TO BE THE
                        * "A" SIDE OF AN MP-L2 IN PORT 7
                        * SYSTEM MUST HAVE RAM AT $E800
                        * SYSTEM WILL HALT IF NOT THERE.

                        * SYSTEM EQUATES

CC33    CPUTYP  EQU     $CC33
CD03    WARMS   EQU     $CD03
CD12    OUTCH2  EQU     $CD12
CD1E    PSTRNG  EQU     $CD1E
CD24    PCRLF   EQU     $CD24
0070    PORT7   EQU     $70

                        * ASCII CODE EQUATES

0004    EOT     EQU     $04
000A    LF      EQU     $0A
000D    CR      EQU     $0D
0011    DVLINE  EQU     $11

                        * EXTERNAL LABEL EQUATES

E8EE    CONOUT  EQU     $E8EE

                        *DEFAULT PORT ADDRESS

E082    OUT_ID  EQU     $E082
E083    CNT_ID  EQU     $E083
E08E    D:DEVEC EQU     $E08E

PRON                                                    PAGE    2

E800                    ORG     $E800
E800 20  65     START   BRA     START1    BRANCH AROUND VERSION
E802 02 2E 01 3A        FCB     $02.$2E.$01.$3A.$07
E806 01
E807 0D 0A 50 52        FCC     CR.LF."PROPERTY OF DIGITAL "
E80B 4F 50 45 52
E80F 54 59 20 4F
E813 46 20 44 49
E817 47 49 54 41
E81B 4C 20
```

```
54   E818 53 50 45 43          FCC   "SPECIALTIES",CR,LF,"DISE"
     E821 49 41 4C 3A
     E825 40 A5 53 0D
     E829 0A 44 49 52
     E82D 45
55   E82E 43 54 53 20          FCC   "CTS ALL OUTPUT TO CONSOL"
     E832 41 4C 4C 20
     E836 4F 55 54 50
     E83A 55 54 20 54
     E83E 4F 20 43 4F
     E842 4E 53 4F 4C
56   E846 45 0D 0A 41          FCC   "E",CR,LF,"ADD PRINTER.  "
     E84A 4E 44 20 50
     E84E 52 49 4E 54
     E852 45 52 2E 20
     E856 20
57   E857 4C 4F 41 44          FCC   "LOADS AT E800",CR,LF
     E85B 53 20 41 54
     E85F 20 24 45 38
     E863 30 30 0D 0A
58   E867 34   12       START1 PSHS  A,X          SAVE A AND X
59   E869 8E   CCE7            LDX   #MESSG       POINT TO MESSAGE
60   E86C BD   CD1E            JSR   PSTRNG       PRINT IT
61   E86F BD   CD24            JSR   PCRLF        PRINT CRLF
62   E872 BE   CD13            LDX   OUTCH2+1     GET CURRENT OUTPUT VECTOR
63   E875 8C   E8D1            CMPX  #POUT        CHECK TO SEE IF PRINTER ALREADY
64   E878 27   52              BEQ   RETURN       IF SO RETURN TO FLEX
65   E87A 8F   E8EE            STX   CONOUT       IF NOT CHANGE OUTPUT VECTOR
66   E87D 8E   E8D1            LDX   #POUT        TO INCLUDE PRINTER
67   E880 BF   CD13            STX   OUTCH2+1     STORE PRINTER AS VECTOR
68   E883 35   10              PULS  X            DONT NEED X ANYMORE
69   E885 86   CC32            LDA   CPUTYP       GET CPU TYPE
70   E888 E0D2           OUTDAT FDB   OUT_ID      DEFAULT DATA REG
71   E88A E0D3           OUTCNT FDB   CNT_ID      DEFAULT CONTROL REG
72   E88C E0DE           OUTDIR FDB   DIREG       DEFAULT DIRECTION REG
73   E88E 04   02              ANDA  #2           STRIP ALL BUT MP-ID
74   E890 81   02              CMPA  #2           CHECK FOR MP-ID
75   E892 27   0E              BEQ   PINIT1       GO INIT PIA IF MP-ID
76   E894 86   70              LDA   #PORT7       IF NOT MP-ID CHANGE
77   E896 B7   E889            STA   OUTDAT+1     TO PORT NUMBER 7
78   E899 4C                   INCA
```

PROM PAGE 3

```
79   E89A B7   E885            STA   OUTCNT+1    "A" SIDE
80   E89D 86   7E              LDA   #PORT7+$E    PORT 7 DIRECTION CONTROL
81   E89F B7   E888            STA   OUTDIR+1
82   E8A2 86   34      PINIT1  LDA   #$34         TURN ON SDE
83   E8A4 A7   9F E88A         STA   [OUTCNT]
84   E8A8 86   FF              LDA   #$FF         ALL OUTPUTS
85   E8AA A7   9F E888         STA   [OUTDAT]
86   E8AE 86   3E              LDA   #$3E         TURN OFF SDS
87   E8B0 A7   9F E88A         STA   [OUTCNT]
88   E8B4 86   FF              LDA   #$FF         SET I/O BUFFER TO OUTPUT
89   E8B6 A7   9F E88C         STA   [OUTDIR]
90   E8BA 86   11              LDA   #$11         SELECT PRINTER
91   E8BC BD   CD12            JSR   OUTC03
92   E8BF 35   02              PULS  A
93   E8C1 7E   C003            JMP   WARMS        GO BACK TO FLEX
94   E8C4 6D   9F E888   E8ADY TST   [OUTDAT]     CHECK FOR PRINTER READY
95   E8C8 73   CCE6            COM   PCHE2
96   E8CE 39                   RTS
97   E8CC 35   12        EXTDSB PULS  A,X         EXIT IF ALREADY RUNNING
98   E8CE 7E   CD03            JMP   WARMS        GOTO FLEX
99   E8D1 AD   9F E0E8   POUT  JSR   [CONOUT]     PRINTER OUTPUT DRIVER
100  E8D5 17   E400      POUT1 LBSR  PCHE
101  E8D8 2A   FB              BPL   POUT1
102  E8DA 7F   CCE6            CLR   PCHE2
103  E8DD A7   9F E888         STA   [OUTDAT]     SEND CHARACTER TO PRINTER
104  E8E1 86   36              LDA   #$36         LOAD STROBE
105  E8E3 8D   02              BSR   POUT2        GO SEND STROBE
106  E8E5 86   3E              LDA   #$3E         LOAD STROBE STOP
107  E8E7 A7   9F E86A   POUT2 STA   [OUTCNT]     STROBE CONTROL
108  E8EB 39                   RTS
109
110  CCD0                      ORG   $CCD0
111
112  CCD0 70   CCE6      PCHE  TST   PCHE2      PRINTER READY ?
113  CCD3 2B   08              BMI   PCHE1
114  CCD5 6D   9F E88A         TST   [OUTCNT]
115  CCE1 1038 1B0F            LBMI  READY
116  CCE5 39             PCHE1 RTS
117  CCE6 FF             PCHE2 FCB   $FF
118  CCE7 50 52 49 4E    MESSG FCC   "PRINTER ACTIVATED".EOT
     CCEB 54 45 52 20
     CCEF 41 43 54 49
     CCF3 56 41 54 45
     CCF7 44 04
119
120                            END   START
```

0 ERROR(S) DETECTED

PROM PAGE 4

SYMBOL TABLE:

```
CNT_ID E0D3   CONOUT E0EE   CPUTYP CC33   CR     000D   DIREREG E0DE
EOT    0004   LF     000A   MESSG  CCE7   OFLINE 0011   OUTCH2 CD12
OUTCNT E0DA   OUTDAT E0D8   OUTDIR E0DC   OUT_ID E0D2   PCHE   CCD0
PCHE1  CCE5   PCHE2  CCE6   PCRLF  CD24   PINIT1 E842   PORT7  0070
POUT   E8D1   POUT1  E8D5   POUT2  E8E7   PSTRNG CD1E   READY  E8C4
RETURN E8CC   START  E000   START1 E867   WARMS  C003
```

PAGE 1

```
3                  ************** PROM ***************
4
5
6                  ** TURNS OFF PRINTER PASS THRU **
7
8
9                  * STANDARD FLEX LABEL EQUATES
10
11            C003 WARMS  EQU   $C003
12            CD0F OUTCH  EQU   $CD0F
13            CD12 OUTCH2 EQU   $CD12
14            CD1E PSTRNG EQU   $CD1E
15            CD16 PUTCHR EQU   $CD16
16            CD2A BSTRIO EQU   $CD2A
17
18                 * ASCII CODE EQUATES
19
20            0004 EOT    EQU   $04
21            000A LF     EQU   $0A
22            000C FF     EQU   $0C
23            000D CR     EQU   $0D
24            0013 OFLINE EQU   $13
25
26                 * EXTERNAL LABEL EQUATES
27
28            E800 CPURAM EQU   $E800     2K X 8 RAM ON CPU BOARD
29            E0EE CONOUT EQU   $E0EE     LOCATION OF CONSOLE OUTPUT
30
31
32       C100           ORG   $C100
33
34       C100 20   53  START  BRA   START1    SEARCH ARCHIVE VERSION
35       C102 82 2E 81 3A       FCB   $83,$2E,$81,$3A,$83 VERSION '
         C106 83
36       C107 0D 0A 50 52       FCC   CR,LF,"PROPERTY OF DIGITAL "
         C10B 4F 50 45 52
         C10F 54 59 20 4F
         C113 46 20 44 49
         C117 47 49 54 41
         C11B 4C 20
37       C11D 53 50 45 43       FCC   "SPECIALTIES",CR,LF,"ERSE"
         C121 49 41 4C 54
         C125 49 45 53 00
         C129 0A 52 45 53
         C12D 45
38       C12E 54 55 20 43       FCC   "TS CONSOLE OUTPUT VE TOE"
         C132 4F 4E 53 4F
         C136 4C 45 20 4F
         C13A 55 54 50 55
         C13E 54 20 56 43
         C142 43 54 4F 52
39       C146 53 0D 0A 54       FCC   "S",CR,LF,"TO TURN OFF ",'"'
         C14A 4F 20 54 33
         C14E 32 4E 20 4F
         C152 46 46 20 22
40       C156 30 52 4F 4E       FCC   "FROM","'",CR,LF
         C15A 22 0D 0A
41       C15D 34   12       START1 PSHS  B,A          SAVE X AND A
42       C15F BE   CD13            LDX   OUTCH2+1     LOAD CURRENT OUTPUT VECTOR
43       C162 8C   E800            CMPX  #CPURAM      SEE IF "PROM" IS RUNNING
44       C165 23   1D              BLS   EXIT         IF NOT QUIT
45       C167 86   0D              LDA   #CR          SEND ENTERS TO PRINTER
46       C169 BD   C810            JSR   PUTCHR
47       C16C 86   0C              LDA   #FF          SEND FORM FEED TO PRINTER
48       C16E BD   CD10            JSR   P_CHR
49       C171 86   13              LDA   #OFLINE      DESELECT PRINTER
50       C173 BD   CD10            JSR   PUTCHR
51       C176 8E   E0EE            LDX   CONOUT       LOAD X WITH ORIGINAL VECTOR
52       C179 EF   CD13            STX   OUTCH2+1     RETURN IT TO OUTCH2
53       C17C BD   CD2A            JSR   RSTRIO       RESTORE I/O VECTORS
54       C17F 8E   C18F            LDX   #MESSG1      POINT TO OFF MESSAGE
55       C182 20   03              BRA   EXIT1        GO PRINT IT
56       C184 8E   C1A4      EXIT  LDX   #MESSG2      LOAD NOT ON MESSAGE
57       C187 BD   CD1E      EXIT1 JSR   PSTRNG       PRINT MESSAGE
58       C18A 35   12              PULS  A,X          RESTORE A AND X
59       C18C 7E   C003            JMP   WARMS        GO BACK TO FLEX
60       C18F 50 52 49 4E    MESSG1 FCC   "PRINTER NOW OFF LINE".EOT
         C193 54 45 52 20
         C197 4E 4F 57 20
         C19B 4F 46 46 20
         C19F 4C 49 4E 45
         C1A3 04
61       C1A4 50 52 49 4E    MESSG2 FCC   "PRINTER WAS NOT ON".EOT
         C1A8 54 45 52 20
         C1AC 57 41 53 20
         C1B0 4E 49 54 20
         C1B4 4F 4E 04
62
63                         END   START
```

0 ERROR(S) DETECTED

SYMBOL TABLE:

```
CONOUT E0EE   CPURAM E800   CR     000D   EOT    0004   EXIT   C184
EXIT1  C187   FF     000C   LF     000A   MESSG1 C18F   MESSG2 C1A4
OFLINE 0013   OUTCH  CD0F   OUTCH2 CD12   PSTRNG CD1E   PUTCHR CD16
RSTRIO CD2A   START  C100   START1 C15D   WARMS  C003
```


NORSE softCODE
10871 Roseland Gate
Richmond B.C. Can.
V7A 2R1

Dear Editor: Re Debugging Tools for OS9

You don't include nearly enough material on OS9 in your magazine, so here is a contribution to help you out.

As a programmer my primary means of debugging in a ROM is to include lots of print statements. The following two subroutines serve somewhat the same function in assembler.

Include them in the source of your program with a LIB or USE statement then call them as shown in the comments.

Regards,
Greg Morse
14 Jan 85

```
*          PUBLIC DOMAIN SOFTWARE DONATED BY:
*               GREG MORSE
*            Richmond B.C. CANADA
*
* file shoregs.src  to be included with use statement
* uses  84. bytes of stack
*
* just put a "LBSR SHOREGS" statement wherever you like
*
endpath  set   2       use stderr for output
* next series of sets define local stack offsets
```

```
sholine  set   0
shoPC    set   sholine+5
shoCC    set   sholine+6
shoDP    set   sholine+12
shoB     set   sholine+13
shoA     set   sholine+14
shox     set   sholine+15
shoy     set   sholine+17
shou     set   sholine+19
shoS     set   sholine+21
shoStop  set   sholine+23
scrtop   set   scrtop
shoSCR   set   scrtop
shoSA    set   sholine+1
shoSB    set   sholine+2
shoSDP   set   sholine+3
shoSx    set   sholine+4
shoSy    set   sholine+6
shoSu    set   sholine+8
shoSpc   set   sholine+10
*
sho001   equ   *
         leay  shomsgen,pcr
         pshs  y,d                 y first then o
         leay  shomsg,pcr
         leax  sholine,u
sho002   ldd   ,y++
         std   ,x++                dumb msg is even number of chars
         cmpy  2,s                 all done?
         blt   sho002              go if not
         puls  d,y,pc              return
*
*  sho regs subrtn no calling parms
*
shoregs  equ   *
         pshs  u,y,x,dp,b,a,cc
         tfr   s,d
         addd  #12                 d=stack before call
         leax  -scrtop,x           reserv scratch area
         tfr   x,u                 set base reg
         bsr   sho001              setup text of msg
         leax  shoa,u              D is still valid
         bsr   showord             sho D as stk ptr
         leax  shocc,u
         lda   shoCC,s             get codes leave on stack
         bsr   shobyte
         leax  shob,u
         lda   shoA,s              get reg leave on stack
         bsr   shobyte
         leax  shox,u
         lda   shoB,s              get reg leave on stack
         bsr   shobyte
         leax  shoDP,u
         lda   shoSDP,s            get reg leave on stack
         bsr   shobyte
         leax  shox,u
         ldd   shoSx,s             get reg leave on stack
         bsr   showord
         leax  shoy,u
         ldd   shoSy,s             get reg leave on stack
         bsr   showord
         leax  shou,u
         ldd   shoSu,s             get reg leave on stack
         bsr   showord
         leax  shopc,u
         ldd   shoSPC,s            get reg leave on stack
         bsr   showord             sho instr after sho call
         leax  sholine,u
         ldy   #shomsg1
         lda   #shopath            path to use for output
         os9   I$Write             msg contains embedded cr/lf
         bcc   sho999
         os9   I$Exit
sho999   equ   *
         leas  scrtop,s              reset stack
         puls  cc,a,b,dp,x,y,u,pc    restore and return
*
*
showord  equ   *                   word in D x==>buffer
         pshs  b                   save lo byte
         bsr   cnvhex              convert hi byte to 2 hex digit
         std   ,x++                save hex digits in buffer
         puls  a                   now do lo byte
shobyte  equ   *                   byte in A x-->buffer
         bsr   cnvhex
         std   ,x++
         rts
*
sho010   equ   *                   cc in a   x==> buffer
         ldb   #'E                 upper case on printout
         bsr   sho110              if bit is on. Lower case
         ldb   #'F                 if it is off.
         bsr   sho110
         ldb   #'H
         bsr   sho110
         ldb   #'I
         bsr   sho110
         ldb   #'N
         bsr   sho110
         ldb   #'Z
         bsr   sho110
         ldb   #'V
         bsr   sho110
         ldb   #'C
sho110   lsla                      shift hi bit into carry
         bcc   sho111                go if hi bit was set
         orb   #$20                cnvrt to low case if off
sho111   stb   ,x+
         rts
*
cnvhex   equ   *                   1 byte in a into two chars in D
         tfr   a,b                 save number
         anda  #$F                 to nibble
         bsr   convrt              convert nibble to char
         exg   a,b                 save char in b
         lsra
         lsra
         lsra
         lsra                      hi nibble
convrt   equ   *     $00-$09 =>  $30-$39; $0A-$0F ==> $41-$46
         adda  #$90
         daa
         adca  #$40
         daa
         rts
*
*       constants area
*
shomsg   fcb   $0A,$0D
         fcc   "PC=.... ;i;i;i;i DP=   A=   B="
         fcc   "   X=     Y=     U=     S=   ",$0D,$0A
shomsgen equ   *
shomsg1  equ   *-shomsg
*---------- end of shoregs source code ----------
```

```
************ file SHOMEM.SRC ********************************
*           PUBLIC DOMAIN SOFTWARE DONATED BY:
*               GREG MORSE
*                   Richmond B.C. CANADA
*
* file shomem.src  to be included with use statement
* uses 88. bytes of stack
*
* load x with starting mem address, Y with number of bytes to sho
* then do LBSR SHOMEM
* uses stderr (path#2) for output
* destroys A&b
*
shopath  set   2         use stderr for output
Sespac   set   $20
SCR      set   $0D
SLF      set   $0A
SHOMEM   equ   *
         pshs  y,x                push y then x
         tfr   y,d
         leay  d,x                ending addr in (Y)
         pshs  y                  save it
         lda   #shopath           print heading line
         leax  ShoLine,pcr
         ldy   #LShoLine
         OS9   I$Write
         leas  -LShoLine,s        reserve area for output line
         ldy   LShoLine+2,s         get orig start addr
         pshs  y                  save it
shom10   equ   *                  sho 1 line (16 bytes)
         ldy   ,s                 get current mem ptr
         cmpy  2+LShoLine,s         past ending address?
         bhs   shom19             go if yes
         leax  2,s                point x at line area
         bsr   shom20             format current addr
         ldb   #16                bytes per line ctr
         pshs  b
shom11   lda   ,y+                get a byte
         bsr   shom50             format as 2 hex digits
         sta   ,x++
         lda   #$Sespac
         sta   ,x+
         dec   ,s                 dec byte count
         bne   shom11             bra back if not zero
         ldd   #$2020             2 spaces
         std   ,x++
         puls  b                  cleanup stack
         ldb   #16                begin display in ascii
         ldy   ,s                 get current mem ptr again
shom15   lda   ,y+                get a byte
         anda  #$7F               mask parity bit
         cmpa  #$Sespac           is it printable?
         bhs   shom17             go if yes
shom16   lda   #'.                replace by '.' if not
shom17   cmpa  #$7F               DEL is not printable
         beq   shom16             so go if DEL
         sta   ,x+                save in buffer
         decb                     done 16 chars yet?
         bne   shom15             go if not
         ldd   #$0D0A             get CR/LF pair
         std   ,x                 save in buffer
         sty   ,s                 save new mem ptr for next loop

         lda   #shopath
         leax  2,s                point x at start of line
         ldy   #LShoLine          y length
         OS9   I$Write            write the line (16 bytes)
         bra   shom10             go do another 16
shom19   leas  4+LShoLine,s       cleanup stack
         puls  a,y,pc             and return
*
*shoMem subroutines
*
shom20   equ   *                  format the address
         ldd   #$2020             2 spaces
         std   ,x++
         tfr   y,d                get hi byte of addr to A
         bsr   shom50             convert to hex
         std   ,x++               put in line buffer
         tfr   y,d
```

```
         tfr   b,a              lo byte of addr to A
         bsr   show50           convert to hex
         std   ,x++             save in buffer
         ldd   #$2020
         std   ,x++
         rts
*
show50   equ   *                binary in A to 2 hex digits in D
         tfr   a,b              save value
         bsr   show51           convert lo nybble to hex
         exg   a,b              save result in b
         lsra
         lsra
         lsra
         lsra                   move hi nybble to low
show51   anda  #$0F             keep only lo nybble
         adda  #$90             tricky but it works
         daa
         adca  #$40             trickier but still works
         daa                    A now has two hex digits in it
         rts                    from show51 and show50
*
ShoLne   fcc   '  ADDR  '
         fcc   '  !  .  .  .  '
         fcc   '  .  .  .  .  '
         fcc   '  !  .  .  .  '
         fcb   $4CR,$0LF
LShoLne  equ   *-@ShoLne
*---------------- end of showaa.src ----------------
```


MORSE softCODE
10871 Roseland Gate
Richmond B.C. Can.
V7A 2R1

Dear Editor: Re OS9 BUG in setting up standard paths

You don't include nearly enough material on OS9 in your magazine, so here is my contribution to help you out.

The following program shows that the standard OS9 paths STDIN and STDERR are not independent, and changing options on one will affect the other in the same way. This is seldom desireable.

The program includes a text string "helmsg" which contains an embedded carriage return. If this string is printed with the OS9 I$WritLn directive, the text after the CR should not be output IF THE LINE EDITING OPTIONS ON THE PATH ARE SET.

When the program is run the following output results (The line numbers are not printed):

1. options on stdin are standard
2. this is ok
3.
4. options on stdin are clear
5. this is ok
6. this should not print
7. stderr has been opned to /TERM
8. stdin options are still clear
9. this is ok
10.

This proves that clearing the line editing option on stdin also cleared it on stderr. Why does this happen? The last test provides a clue. If stderr is opened explicitely then the problem does not occur. Thus I deduce that when the SHELL sets up the standard paths, it must use the I$DUP call not the I$Open. As a result all three paths share the same path descriptor and changing the setting of one changes them all.

Greg Morse

GREG MORSE
13 Jan 1985

```
         ifpl   use /d0/defs/os9defs
         use /d0/defs/os9defs
         endc
      mod modsiz,modnam,prgrm+objct,reent+1,xeqstr,datend
modnam   fcs "tstWrLn"
*
      org 0       begin data area
iconopt rmb   32
stakspt rmb 200
datend equ .
*
helmsg fcc "this is ok",$0A,$0D,"this should not print "
      fcb $0A,$0D
Lhelmsg equ *-helmsg
stdomsg fcc "options on stdin  are standard",$0A,$0D
lstdomsg equ  *-stdomsg
clrmsg fcb $0A,$0D
      fcc "options on stdin  are clear",$0A,$0D
Lclrmsg equ *-clrmsg
pthmsg  fcc "Stderr has now been opened to /term",$0A,$0D
      fcc "stdin options still clear",$0A,$0D
Lpthmsg equ *-pthmsg
conopt  fdb 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
stdpth fcs "/TERM"
*
xeqstr equ *            exec entry point
      leax   iconopt,u  save the options
      lda #0              on stdin
      ldb #SS.opt            so they can be restored
      os9 I$getstt              later
* print first test message
      lda #2          stderr path
      leax stdomsg,pcr
      ldy  #lstdomsg
      os9 I$WRITE
      leax    helmsg,pcr
      ldy    #lhelmsg
      os9 I$WritLN     should end at 1st CR and does
* now we clear options on the STDIN path
* as we will see this affects the STDERR path also
      leax   conopt,pcr   all zero options
      lda #0              on stdin
      ldb #SS.opt
      os9 I$Setstt
      leax clrmsg,pcr     tell about clearing options
      ldy #lclrmsg
      lda #2
      os9 I$WRITE
      leax   helmsg,pcr   print same msg as before
      ldy #lhelmsg
      lda #2              on stderr
      os9 I$Writln    should end at first CR but doesn't
* now try with stderr opened explicitly
      lda #2
```

```
      os9 I$Close
      leax    STDpth,pcr
      lda #WRITE.
      os9 I$open      should use path #2
      bcc tst10
      os9 F$Perr
tst10 leax pthmsg,pcr
      ldy #lpthmsg
      os9 I$Write
      leax helmsg,pcr
      ldy #lhelmsg
      os9 I$WritLn    should end with first cr and does
* reset the options on stdin
      lda #0
      ldb #ss.opt
      leax    :conopt,u
      os9 I$setstt
      os9 F$exit
      emod
modsiz equ *
```

WINDRUSH
Micro Systems Ltd

SCREDITOR III VERSION NUMBERS

Dear Don,

With reference to our letter WCD/MB dated 13/12/86 regarding the changes in our upgrade policy. Please note that we have both FLEX and OS9 versions of SCREDITOR III. These products have fundamentally different source code modules and therefore have different version numbers. This information was inadvertently omitted from our previous letter. The appropriate versions are as follows:

SCREDITOR III (FLEX) 1.25D

SCREDITOR III (OS9) 1.20B

The upgrade charge for OS9 and FLEX versions will be the same i.e. £25.00 disk only and £45.00 for disk and manual.

CHANGE IN UPGRADE POLICY

Dear Don,

Until we've held back doing this as long as we could but with the steady increase in international postal charges for the past two years we can refrain no longer.

Effective January 1, 1987 our upgrade charges will be as follows:

1. Upgrade disk only for any Windrush product £25.00
2. Upgrade disk and manual for any Windrush product other than PL/9 or SCREDITOR III. £35.00
3. Upgrade disk and manual for PL/9 or SCREDITOR III £45.00

The above prices include air mail postage.

Current version numbers are:

GRACE 2.01

SPACE 2.51

ASMOS 2.30

C-BUG 9.6.88

Access 'C' 25.2:0

PL/9 ..... 6.23

SCREDITOR III 1.20B

To take advantage of this service the user must:

1. Return the ORIGINAL Master disk (not a copy).
2. Enclose a cheque, Money Order or credit card authorisation.

The disk should be sent fully insured for the full price of the product as per our advertisement. We will not be held responsible for disks that fail to reach us. The customs declaration (green sticker) should read "Goods Of UK Origin".

Any application for an upgrade that is not accompanied by the original master disk will be returned without action.

MOTOROLA INC.

Microprocessor Products Division
3501 Ed Bluestein Blvd
Austin, Texas 78721

For further information contact:

Editor Contact: Val Bauer

512 928-6804

Reader Contact: Dean Mosley

512 440-2839

Press Information

MOTOROLA INTRODUCES M68000 SERIES DMA CONTROLLERS

Motorola Microprocessor Products Group introduces their complete line of high-performance DMA controllers to support the M68000 MPU Family--the MC68440, MC68442, and MC68450. These DMA devices complement the performance and architectural capabilities of the M68000 MPU Family by providing a means to move large blocks of data within these systems in a quick, efficient manner with minimum intervention by the MPU itself; thus, these devices aid in high throughput maintenance.

The MC68440, MC68442, and MC68450 are directly compatible with the M68000 bus architecture and provide full support for all bus exception conditions defined for the M68000 Family. These devices are also compatible with the VMEbus and VERSAbus asynchronous bus standards.

The MC68440 is a two-channel high-performance DMA device that supports a 24-bit linear address space. The MC68442 is an extended version of the MC68440, providing a full 32 bits of linear addressing and an additional function code output. The MC68450 is a four-channel DMA controller which is upward pin and register compatible with the MC68440.

The programmable function code outputs of these devices can be used to specify M68000 compatible bus cycle type information on each DMA bus cycle. These DMA devices can thus directly interface with memory management devices such as the MC68451. The additional function code output of the MC68442 can be used to distinguish between DMA and CPU bus cycles.

The MC68440, MC68442, and MC68450 allow four different modes of DMA request generation. These modes include the externally-generated cycle steal and burst modes, as well as the internally-generated maximum rate and limited rate auto request (LRAR) modes. In the LRAR mode, the amount of the total available bus bandwidth utilized by a DMA channel is programmable. This feature allows the DMA's bus utilization to be tailored to the available bus bandwidth in a given system. These DMA devices support a variety of explicitly or implicitly addressed peripherals with flexible handshaking conventions.

Both the MC68440 and MC68442 provide two high-performance, 16-bit DMA channels and have the capability of operating on an

8-bit data bus in conjunction with the MC68008 MPU. These devices will be available in 8, 10, and 12.5 MHz versions yielding maximum peripheral-to-memory rates of 4, 5, and 6.25 Megabytes/second, respectively.

The MC68450 is a four-channel, 16-bit DMA controller that supports a 24-bit linear address space. It is a pin and register compatible superset of the MC68440. In addition to two more DMA channels, the MC68450 supports greater flexibility in the DMA transfer operations. It also fully supports array chained and linked array chained DMA transfer operations.

The MC68450 also supports peripheral generated channel start pulse and a cycle steal with hold mode. The cycle steal with hold feature can be used to minimize the bus arbitration overhead in the limited rate auto request transfer mode. This part can accommodate byte, word, odd aligned word, and long word operand transfers. The MC68450 will be available in 8 and 10 MHz versions providing 4 and 5 Mbyte/sec maximum transfer rates, respectively.

The MC68440, MC68442, and MC68450 are available in a 68-lead pin grid array package. The MC68442 is priced $59.69 in the PGA. The MC68440 is also available in a 64-pin DIP, and is priced $39.69 for plastic. The MC68450 is also available in a 64-pin ceramic DIP and is priced $78.43. All these prices are for 100 piece quantities. Sampling is now, while production quantities will be available 1Q 85.

MOTOROLA ANNOUNCES A 64K BIT DYNAMIC RAM, THE MCM6164BP15

Reader Contact: Betty Prince
512 928-6700

Motorola Memory Products Division announces availability of the MCM6164BP15, a 64K Bit HMOS Dynamic Random Access Memory (RAM). This yield-enhanced high-speed version of the popular MCM6665A, features a smaller die size and laser redundancy.

The MCM6164BP15 incorporates a maximum 150 nanoseconds (ns) access time with low maximum power dissipations of 302.5 milliwatts (mW) in the active mode and 22mW standby, optimizing its advantage in high density mainframe memory systems.

With a 128-cycle 2 millisecond (ms) Refresh, and fast Page Mode Cycle time of 155ns, the MCM6164BP15 maintains upward pin compatibility from the 16K MCM6116 RAM and MCM6517 RAM, as well as being fully TTL compatible. The device, containing $\overline{RAS}$-only Refresh mode and $\overline{CAS}$ controlled output, operates from a single + 5 volt (V) (± 10%) power supply. By multi-plexing row-and-column-address inputs, the MCM6164BP15 can be packaged in a standard 16-pin dual-in-line plastic package (DIP).

The MCM6164BP15 is available now, and pricing in quantities of 1 - 24 is $5.74 each; $5.57 each for 25 - 249; and $5.18 each for 250 - 299. Contact your distributor or local Motorola sales office for further information.

MOTOROLA INTRODUCES THE MC68881

HIGH PERFORMANCE FLOATING PROCESSOR...

THE ONLY FULL FUNCTION FLOATING POINT DEVICE AVAILABLE

Products Group has expanded its high performance family of Microprocessor Units (MPUs) and peripherals with the introduction of the MC68881 Floating Point Coprocessor (FPCP). The MC68881 is a high performance, single chip HCMOS VLSI floating-point coprocessor. The FPCP is primarily intended to operate as a closely-coupled coprocessor with the full 32-bit MC68020 MPU, but it can also operate as a memory-mapped peripheral in systems based on any other M68000 Family MPU.

High performance, broad based functionality, and flexibility facilitate use in a wide range of applications. The MC68881 can be used in performance of graphic work stations, Computer Automated Design (CAD), and Computer Aided Engineering (CAE) systems. High performance with reduced size and cost will enhance scientific analysis systems, medical equipment, and robotic and numerical control applications. The MC68881 places the power of a scientific computer with complex functions on a single chip, reducing the size and cost of desktop systems.

The MC68881 performs floating point math calculations in strict accordance with the most recent revision of the Institute of Electrical and Electronic Engineers (IEEE) Floating Point Specification (P754 Draft 10.0). This conformance guarantees the support of all required operations, data types, rounding modes, and rounding precisions. The MC68881 is the only floating point chip on the market today that can make this guarantee.

The MC68881 calculates basic math functions (add, subtract, multiply, and divide) twice as fast as any other single-chip math processor. In addition to the operations defined by IEEE, the MC68881 performs a full selection of transcendental and non-transcendental functions. These operations include, but are not limited to, root values, trigonometric functions, exponentials, hyperbolics, and logarithmics. All functions are calculated to 80 bits of precision in hardware. The design of the FPCP eliminates inefficiencies associated with executing software "envelopes" via a hardware implementation.

In 32-bit MC68020 based systems, the MPU and the FPCP communicate via the Coprocessor Interface, which is a standard feature of the MC68020 architecture. The MPU and the FPCP share the tasks of interconnect. The MPU passes coprocessor instructions to the MC68881. This flexible interconnect scheme is a combination of standard signal lines and coprocessor instructions. No special signals are required. The Coprocessor Interface is transparent to the system programmer, as coprocessor instructions are written as a part of the main program instruction stream. The MC68881 can operate concurrently with the main processor, thereby freeing the CPU (Central Processing Unit) for other tasks. The MC68881 will support future generations of M68000 family processors by utilizing the standard Coprocessor Interface.

Floating point functions can be added as memory mapped peripherals to other M68000 MPUs including the Reduced Bus MC68008, the 16/32-bit MC68000, the Virtual Memory MC68010, and their Expanded Virtual MC68012. This is accomplished by providing instruction sequences that emulate the protocol of the Coprocessor Interface.

The MC68881 architecture includes a 67-bit Arithmetic Unit, Barrel Shifter, and eight 80-bit general purpose registers. Using the 2 micron HCMOS process, this high density device contains an effective transistor count of approximately 155,000 on a chip of 270 x 330 millimeters, and low power dissipation of less than 1.0 Watt maximum. Standard clock frequencies are 12.5 megahertz (MHz) and 16.67 MHz. It is packaged in a 68-lead Pin Grid Array with a 1 inch square footprint.

Sampling to Alpha and Beta site customers to ensure complete operational integrity began in December, 1984, and will continue through February, 1985. General samples of the MC68881 at 12.5MHz will be available beginning in March, 1985, limited quality production scheduled to begin in July, 1985. Sample piece price is $375.00.

Press Release!

LLOYD I/O is pleased to announce the availability of our improved X-BASIC(tm) compiler which supports the TRS XBASIC set of commands for the price of $199.00 for OS9 and FLEX.

X-BASIC version 1.2 supports random files using virtual arrays and record I/O, in addition to sequential files. PRINT USING has also been implemented. X-BASIC supports six data types including strings, floating point numbers, and four sizes of integers (8, 16, 32, and 64 bits). X-BASIC has been improved in the area of handling expressions with spaces allowed and automatic data type conversions. Several new functions, loop structures, blocked IF-ELSE-ENDIF, and nest level indenting have been added to make programming easier. The MID$() = assignment enables assigning strings to the middle of an existing string. PRECISION= sets the BCD floating point mantissa length between 3 and 99 digits. The date/time variables YEAR, MONTH, DAY, HOUR, MINUTE, and SECOND return their appropriate values when evaluated. X-BASIC is rich in all types of functions, predefined variables, and constants. Included on the disk are several sample programs which the user may evaluate.

Previous customers should send in their disks for update. Include a check for $15.00 in US funds drawn on a US bank unless it has been less than 60 days since purchase or previous update.

X-BASIC is available from LLOYD I/O at (503) 666-1097, 19535 NE Glisan Street, Portland, OR 97230 USA. X-BASIC is also available through these fine distributors:

	FLEX	OS9
North America:	S.E. MEDIA 5900 Cassandra Smith Rd. Hixson, TN 37343	S.E. MEDIA ---
Phone:	1-800-338-6800	
England:	WINDRUSH MOATSTAD LABORATORIES NORTH WALSHAM NORFOLK, ENGLAND NR28 9SA	VIVAWAY LTD 36-38 JOHN STREET LUTON, BEDFORDSHIRE ENGLAND LU1 2JE
Phone:	(692) 404086	(0582) 423425
Germany:	LAUER COMPUTER GmbH Im Schwarzenstein 34 D-5521 Irrel West Germany	KEIL SOFTWARE Forphyrstraße 15 D-6903 Schriesheim West Germany
Phone:	0 65 25 - 2 99	(0 62 03) 67 41
SW Pacific:	PARIS RADIO ELECTRONICS P.O. BOX 380 Darlinghurst, NSW 2010 Australia	PARIS RADIO ---
Phone:	(612) 344 9111	

PRODUCT ANNOUNCEMENT

Wellwritten Enterprises is pleased to announce the availability of our disk subsystem for 6809 computers using the SS-50 bus. The subsystem includes a top-quality 27 MB (formatted) hard disk drive, hard disk controller, interface card (requires SS-30 slot), all cables, and a minifloppy diskette of source assembler language interface programs for your choice of FLEX9™, OS9/1™, or OS9/11™. The drive and controller may be mounted inside a computer chassis (like a Smoke Signal Chieftan™) or in a seperate cabinet with power supply (like the ELEKTRA HD-5W from AAA Chicago for $199). The introductory price is $2850 (check or money order). For those wishing to supply their own ST410 or ST506 hard disk drive, the above described subsystem minus the 27 MB drive is available for $600. The SS-30 interface card and software are available for $200. Oklahoma residents must add sales tax. Please describe your hardware, port size, and version of operating system when ordering. Call (405) 364-6856 for more detailed information.

Dear Don,

"Old SS-30's never die. They just get passed on to a new generation of idealists." That remark by Terry Haas of Chicago, Illinois, was inspired by my purchase from him of a Gimix 80x24 video board to go with the SWTPc 6809 system I just bought from Ron Anderson of Ann Arbor, Michigan. The DC-4 disk drive controller was purchased from Joe Aulicino of Brooklyn, New York. The SWTPc began life in 1976 as a 6800 system, which must qualify it as one of the oldest working microcomputers. Of course, it has been upgraded somewhat to its present MP-09 6809 processor board, three 16K and one 8K memory boards, MP-C and MP-S serial, and MP-L parallel ports. With the drop in RAM prices, I'm looking forward to replacing the four power hogs with a 256K board.

My first personal computer, a Radio Shack Color Computer, was purchased a few years ago and developed into a pretty powerful system composed of stuff from a dozen manufacturers. The third party suppliers made it possible for a lot of us CoCoNuts to expand our computing horizons far beyond Tandy's vision. It was largely the writing of Ron Anderson and others in *68' Micro Journal* that led me to decide on staying with the 6809 when I moved on from games and graphics. Even more important than the power of the 6809 was the level of support I knew I could count on from 6809 users, suppliers, and magazines.

Sincerely,

Center for Creation Studies
Liberty Baptist College
Lynchburg, VA 24506

Lane P. Lester, Ph.D.
Director

MICROMOTION
12077 Wilshire Blvd #506
Los Angeles, CA 90025

FOR IMMEDIATE RELEASE

ATTN: NEW PRODUCT EDITOR

February 8, 1985

Linda Kahn - Publicist
(213) 478-7598

MASTERFORTH NOW AVAILABLE FOR MACINTOSH USERS

OFFERS FORTH-83 PORTABILITY AND TOOL BOX SUPPORT

Los Angeles, CA, Feb. 8 -- MicroMotion is pleased to announce the release of MasterForth for the Macinto h computer. MasterForth is a state-of-the-art implementation of the Forth programming language. MasterForth provides a complete programming environment for the Macintosh, including a 68000 macro-assembler and support for the mouse, the finder, menus, and the graphics toolbox. Relocatable utilities and transient definitions make it possible to run substantial software packages even on a 128k Macintosh. The string package and resident debugger are standard features. MasterForth exactly matches the Forth-83 standard dialect described in Mastering Forth (Brady, 1984), which is included in the package.

MasterForth version 1.0 also is available for the IBM PC, the Apple II series, CP/M, and the Commodore 64. Software can be written on one system and run on all the others. MasterForth retails for $125.00. Several optional extensions are available.

MicroMotion has been providing Forth software and services since 1980. MicroMotion is located at 12077 Wilshire Blvd. #506, Los Angeles, CA 90025, (213) 821-4340.

# # # # #

(213) 821-4340

EASI **Engineering & Architectural Systems International**

FOR IMMEDIATE RELEASE — Editor: Contact Dr. L. Louis Chu at 415/563-9335 for more information.

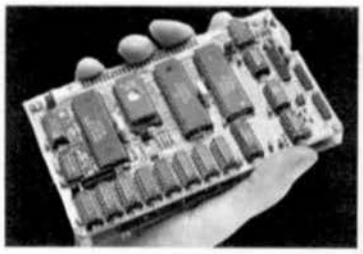
PUBLICATION - "CASE ONE"

CASE ONE is a bimonthly newsletter on the usage and understanding of computer applications amongst civil engineers, contractors, and architects. CASE ONE features informative articles on how to buy software and hardware, practical tips on effectively using computers, and valuable information on current trends in computer technology. CASE ONE also emphasizes the utilization of the state-of-the-art microcomputer software such as spreadsheets and database management programs for technical applications. Price: $30.00/year. FREE sample copies are available for interested readers. Send inquiries on company letterheads to: EASI Inc., 2030 Union Street, Suite 380, San Francisco, CA 94123. 415/563-9335. 1985. ISSN: 0743-1732.

PUBLICATION/COMPANY LITERATURE - "BITS"

Business, Industrial, & Technical Software Catalog & Buyer's Guide

BITS is a catalog and buyer's guide of computer software for the construction industry. The 44-page catalog is published by Engineering & Architectural Systems International Inc. (EASI). BITS features practical articles of interest to computer-using construction professionals, in addition to comprehensive descriptions of over one-hundred computer programs in the areas of general business and scientific applications, computer aided design and drafting, engineering graphics, critical path scheduling, estimating, construction management, structural, geotechnical and mechanical engineering, surveying and hydraulics/hydrology. Priced at $6.00 per copy, BITS will be sent to interested readers FREE. Send inquiries on company letterheads to: EASI Inc., 2030 Union Street, Suite 380, San Francisco, CA 94123. 415/563-9335.



Available Assembled and Tested

Compact Flexible 6809 Computer

The new ST-2900 system — a complete 64K small business or hobbyist computer is only one of its many possible configurations. Among its features are:

- Small enough to hold in your hand! (Eurocard size: 3.9" x 6.3")
- Two board "system" for greater versatility than single board computers.
- CPU Board — powerful 6809E processor, 16K or 64K RAM, 2K-8K EPROM, 2 RS232 serial ports with software programmable baud rates, 16 bit counter/timer. Run the CPU board all by itself, or plug your own custom board or our FDC board into the expansion connector.
- FDC Board — double-sided/double-density floppy disk controller with adjustment free digital data separator and write precompensation, 2-8 bit parallel ports, 2-16 bit counter/timers, prototyping area.
- Available as bare PC boards or **fully assembled and tested** boards. All have solder mask both sides plus silkscreened component overlay.

- OS-9 for only $49?

Well, not quite. But that's all you pay for our OS-9 Conversion Package which lets you use the low cost Radio Shack CoCo version of OS-9 on our ST-2900 system. Save $131 off the suggested list price of OS-9! **No programming is involved.** Supports CoCo OS-9 and standard OS-9 format disks.

• CPU bare board plus EPROM	$45	FLEX Conversion Package	$29
FDC bare board	$38	OS-9 Conversion Package	$49
CPU + FDC board set assembled and tested			$329

• Add $5 shipping/handling ($10 overseas). These prices are in U.S. funds. Canadian orders: call or write for prices. Terms: money order, certified check, VISA.

(FLEX is a trademark of Technical Systems Consultants. OS-9 is a trademark of Microware and Motorola)

SARDIS TECHNOLOGIES

Write for free brochure and complete price list.

(604) 255-4485 (4-5 pm PST)

2261 E. 11th Ave. Vancouver, B.C., Canada V5N 1Z7

Give Your OS-9 System The Power It Deserves!
Total Management Planning Software presents

The TMP POWER SERIES

New! Uniflex Version!

The TMP FREEFORM FILER
The Premier Text Filing Program!

UNIQUE CAPABILITIES:

- **THE OBJECTIVE:** For those who want to randomly store information, but retrieve it quickly, the *FREEFORM FILER* is designed to bridge the gap between word processing and traditional database management programs.
- **THE KEYWORD SYSTEM:** As you enter or edit your text, you can select any word as a "KEYWORD" or "KEYPHRASE." Each "CARD" can contain as many as 117 KEYWORDS, and up to nine pages of text similar to a 3" x 5" card, with no field restrictions. Each "FILE DRAWER" can contain up to 32,767 pages!
- **THE SEARCH:** Search for the **Card Title, KEYWORD,** or a *combination* of both. Plus, *"WILD CARD SEARCHES"* save time!
- **THE RESULTS:** List the titles of cards found, print, or write the cards to a disk file for later printing or use with a word processor.

GIVE YOU MORE ABILITIES:

- ★ **BUSINESS:** Appointments, office phone and address indexes, inventory, service calls, vendor lists, and sales orders.
- ★ **WRITERS AND RESEARCHERS:** For indexing, cross-referencing or cataloging and legal and medical research.
- ★ **PERSONAL:** The possibilities are endless . . . art and coin collections, bill paying, tax records, and home inventory.

NOTE: The *FREEFORM FILER* integrates data into the *"POWER MANAGER."*

'68' MICRO JOURNAL Said: "TMP FREEFORM FILER is a flexible Program that can be used for a multitude of tasks without requiring a Computer Science background . . . a very useful Program."

Requires 128K, OS-9 Version, $195; Uniflex version, $295.

The TMP POWER MANAGER
. . . Best In Its Class!

POWERFUL CAPABILITIES:

- More *characters per record (7500)* than any other program in its class!
- Each database can contain 32,767 records, with up to 150 fields in each record and up to 50 characters in each field.
- Powerful *Sort and Report Generation capabilities.*
- *SORTING* on any field or on a combination of fields.
- Intricate math *CALCULATING* between fields.

GIVE YOU MORE ABILITIES:

★ *POWER MANAGER* can create *CUSTOMIZED LETTERS, INVOICES, COLUMNAR REPORTS, OR LABEL FORMATS* for mailing!

★ Our Users put *POWER MANAGER* to work for them to do: . . . customer mailings . . . past due notices . . . invoicing . . . sales analysis . . . inventories . . . credit, insurance and employee records, client profile reports, tracking stock portfolios, and much more.

► ***THE BOTTOM LINE:*** The *POWER MANAGER* is the best in its class! **Requires 128K, $365. (OS-9 Only)**

The TMP POWER PLANNER
. . . Unequalled in Speed and Power!

★ The *POWER PLANNER* is an electronic spreadsheet with extra Speed and Power due to a unique feature called "circular referencing" that recalculates only the related cells in the spreadsheet.

★ Data and formulas can be entered *IN ANY ORDER.* And, you don't have to keep recalculating for the right answer as in other spreadsheets.

★ *SPECIALIZED REPORTS* can be easily created by overlaying any number of screens and automatically updating one spreadsheet with another!

★ Other features include "Snap-Shot" printing, full 13-digit precision, standard arithmetic and trig functions, and a worksheet that will display up to 24 rows by 255 columns.

★ The Speed and Power of the *POWER PLANNER* make it a natural for Budgeting, Cash Flow Comparisons, Sales Forecasts, Profit/Loss Projections, and all kinds of Financial Analysis and "What If" Calculations.

Requires 64K, $250 (OS-9 Only).

The TMP FRONTEND . . . A Powerful Menu Program

★ The FRONTEND allows the user to call up TMP or other programs such as application programs, editors, shell scripts, or commands, from a menu system. Capacity is seven menu screens, each with up to 36 menu options, producing 252 options. Can be called from non-TMP programs.

Requires 64K, OS-9 Version $50, Uniflex Version, $75.

e TMP LABELER

★ Lets you make large quantities of labels in seconds, with options for automatic numbering and selecting the number of copies of each label. Variable pitch and 8-line screen. Great for Serial Numbers, Inventory, or making a quick Shipping Label. **$75, Uniflex Only.**

► **ORDERING INFORMATION: TMP SOFTWARE**
2431 E. Douglas • Wichita, Kansas • 67211
► **OR CALL TOLL-FREE: 1-800-25 -1382 Ext.47**
We accept VISA, MC, AMEX, money orders and checks.

NOTE: The parent company of TMP Software, The United Software Co., is now the distributor and support organization for TMP Software.

Classified Advertising

SSB 2MHz 6809 4Meg Hard, DSDD floppy 96K, OS-9, Introl-C, BASIC09, much more. Beat offer. Also DSDD drive, 16K, SS-50 Chasis, SWTPC boards, Terminals, MX-80. Robert (414)755-6982

***

SS50 6809 ayatem, includes F&D BMB1, Data Systema 6809 & 64K DRAM cards, F&D floopy disk interface, 2 aerial I/O cards, and two 40 track disk drives. All hardware has complete doc. Runs FLEX9 OS. Also includes Screditor III, Spreadsheet program and lots of utilities. Jim (201) 299-9499 or write POB 497C, Covent Station, N.J. 07961. Any reasonable accepted.

TELETYPE Model 43 PRINTER - with serial (RS232) interface, and full ASCII keyboard. **LIKE NEW** - New coat $1295.00-ONLY **$759.00** ready to run.
Call Tom - Bob, CPI (615) 842-4600

***

For Sale: Motorola 128K Memory Boards, removed from SWPTC S/09 $795.00, SWPTC 8212 Terminals Demoatrators $795.00, Hazelwood Dynamic 64K Memory Boards $395.00 Call ask for Tom (615) 842-4600

***

Wanted to Buy: Used 6800 & 6809 FLEX software. Commercial and public domain. Send description and asking price to: John Current, P.O. Box 273 Honolulu, HI 96859

***

Selling GIMIX 6809 computer with clock, DMA disk controller, 56K RAM. FLEX-9, OS-9 software. Only $1295.
John Pomeroy (216) 372-4457.

***

2 Tandon SS/SD or DD drives-15 hrs total time, currently in use. Larry Hughes, Prestona Burg, Ky 41653 (606)886-6049

COMPILER EVALUATION SERVICES
By: Ron Anderson

The S.E. MEDIA Division of Computer Publishing Inc., is offering the following 'SUBSCRIBER SERVICE':

COMPILER COMPARISION AND EVALUATION REPORT

Due to the constant and rapid updating and enhanchment of numerous compilers, and the different utility, appeal, speed, level of communication, memory usage, etc., of different compilers, the following services are now being offered with periodic updates.

This service, with updates, will allow you who are wary or confused by the various claims of compiler vendors, an opportunity to review comparisons, comments, benchmarks, etc., concerning the many different compilers on the market, for the 6809 microcomputer. Thus the savings could far offset the small cost of this service.

Many have purchased compilers and then discovered that the particular compiler purchased either is not the most efficient for their purposes or does not contain features necessary for their application. Thus the added expense of purchasing additional compiler(s) or not being able to fully utilize the advantages of high level language compilers becomes too expensive.

The following COMPILERS are reviewed initially, more will be reviewed, compared and benchmarked as they become available to the author:

PASCAL "C" GSPL W IMISCAL PL/9

Initial Subscription - $39.95
(includes 1 year updates)
Updates for 1 year - $14.50

S.E. MEDIA - CPI
5900 Cassandra Smith, POB 794
Hixson, TN 37343
615 842-4601

FEATURES THE POWERFUL, THIRD GENERATION, MOTOROLA 6809 PROCESSOR!

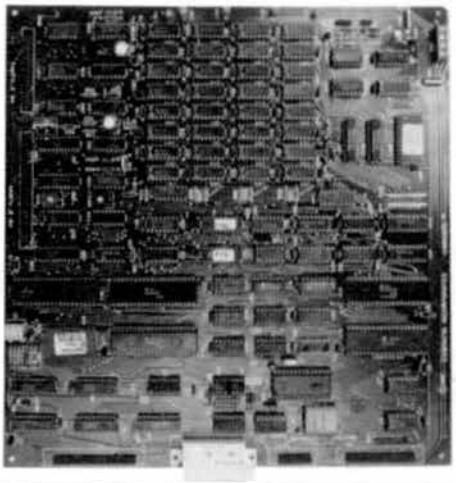
THE 6809 "UNIBOARD"™

SINGLE BOARD COMPUTER KIT

PERFECT FOR COLLEGES, OEM'S, INDUSTRIAL AND SCIENTIFIC USES!

64K RAM! DOUBLE DENSITY FLOPPY DISK CONTROLLER!

New! Lower Price!



BLANK PC BOARD
$99.95
WITH PAL'S, AND TWO EPROMS.
FOR 5-1/4 OR 8 INCH SOURCE DISKETTE ADD $10.

$249.00
COMPLETE KIT! FULLY SOCKETED.

PRICE CUT!!

THE COMPACTA UNIBOARD™: Through special arrangement with COMPACTA INC., we are proud to have been selected the exclusive U.S. Mfg. of their new 6809 UNIBOARD™ COMPUTER KIT. Many software professionals feel that the 6809 features probably the most powerful instruction set available today on ANY 8 bit micro. Now, at last, all of that immense computing power is available at a truly unbelievably low price.

ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A FREE COPY IS AVAILABLE UPON REQUEST.

FEATURES:

- ★ 64K RAM using 4116 RAMS.
- ★ 6809E Motorola CPU.
- ★ Double Density Floppy Disk Controller for either 5-1/4 or 8 inch drives. Uses WD1793.
- ★ On board 80 x 24 video for a low cost console. Uses 2716 Char. Gen. Programmable Formats. Uses 6845 CRT Controller.
- ★ ASCII keyboard parallel input interface. (6522)
- ★ Serial I/O (6551) for RS232C or 20 MA loop.
- ★ Centronics compatible parallel printer interface. (6522)
- ★ Buss expansion interface with DMA channel. (6844)
- ★ Dual timer for real time clock application.
- ★ Powerful on board system monitor (2732). Features commands such as Go To, Alter, Fill, Move, Display, or Test Memory. Also Read and Write Sectors. Boot Normal, Unknown, and General Flex™.

YOUR CHOICE OF POPULAR DISK OPERATING SYSTEMS:

FLEX™ from TSC	$99
OS9™ from Microware	$199

Specify 5-1/4 or 8 inch

PC BOARD IS DOUBLE SIDED, PLATED THRU SOLDER MASKED, 11 x 11-1/2 IN.

Digital Research Computers
(OF TEXAS)
P.O. BOX 461565 · GARLAND, TEXAS 75046 · (214) 225-2309

TERMS: Shipments will be made approximately 3 to 6 weeks after we receive your order. VISA, MC, cash accepted. Add $4.00 shipping.
USA AND CANADA ONLY

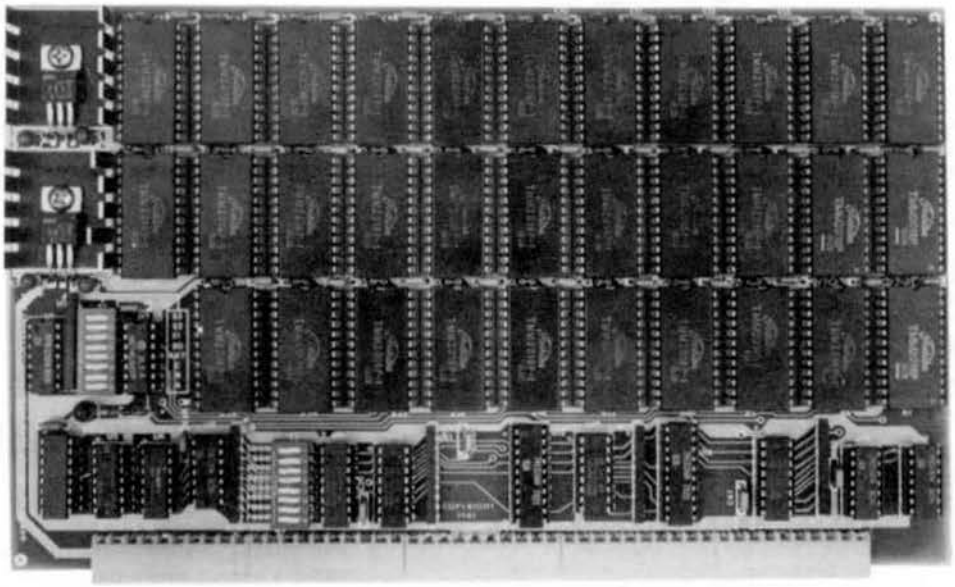
64K SS-50 STATIC RAM

PRICE CUT!!

$149.00 (48K KIT)

NEW!

LOW POWER!



RAM OR EPROM!

BLANK PC BOARD WITH DOCUMENTATION $45

SUPPORT ICs + CAPS -	$18.00
FULL SOCKET SET -	$15.00

ASSEMBLED AND TESTED **ADD $50**

FEATURES:

- ★ Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- ★ Fully supports Extended Addressing.
- ★ 64K draws only approximately 500 MA.
- ★ 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- ★ Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- ★ 2716 EPROMs may be installed anywhere on Board.
- ★ Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- ★ One Board supports both RAM and EPROM.
- ★ RAM supports 2MHZ operation at no extra charge!
- ★ Board may be partially populated in 16K increments.

56K	$169
64K	$199

CLOSE OUT SPECIAL
WE HAVE DROPPED OUR 32K SS-50 STATIC RAM BOARD WHICH USED 2114 LOW POWER RAMS. WE WILL SELL THE REMAINING STOCK OF BLANK PCB'S WITH DATA FOR $17.50 EA. THESE FORMERLY SOLDFOR $50.

16K STATIC RAMS?

The new 2K x 8, 24 PIN, static RAMs are the next generation of high density, high speed, low power, RAMs. Pioneered by such companies as HITACHI and TOSHIBA, and soon to be second sourced by most major U.S. manufacturers, these ultra low power parts, feature 2716 compatible pin out. Thus fully interchangeable ROM/RAM boards are at last a reality, and you get BLINDING speed and LOW power thrown in for virtually nothing.

Digital Research Computers
(OF TEXAS)
P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 225-2309

TERMS: Add $2.00 postage. We pay balance. Order under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50, add 85¢ for insurance.

WINDRUSH MICRO SYSTEMS

UPROM II

PROGRAMS and VERIFIES: 2758, 2508, 2716, 2516, 2732/2732A, MCM68766/6, 2764/2764A, 2564, 27128/27928A, and 27256. I=Intel, T=Texas, M=Motorola.

NO PERSONALITY MODULES REQUIRED!

TRI-VOLT EPROMS ARE NOT SUPPORTED

INTEL's Inteligent Programming (tm) implemented for Intel 2764, 27128 and 27256 devices. Intelligent programming reduces the average programming time of a 2764 from 7 minutes to 1 minute 15 seconds (under FLEX) with greatly improved reliability.

Fully enclosed cable with 5' of flat ribbon cable for connection to the host computer MC6821 PIA interface board.

MC6809 software for FLEX and OS9 (Level 1 or 2, Version 2.2).

BINARY DISK FILE offset loader supplied with FLEX, MDOS and OS9.

Menu driven software provides the following facilities:

a. FILL ..... a selected area of the buffer with a HEX char.
b. MOVE ..... blocks of data.
c. DUMP ..... the buffer in HEX and ASCII.
d. FIND ..... a string of bytes in the buffer.
e. EXAMINE/CHANGE . the contents of the buffer.
f. CRC ..... checksum a selected area of the buffer.
g. COPY ..... a selected area of an EPROM into the buffer.
h. VERIFY ..... a selected area of an EPROM against the buffer.
i. PROGRAM ..... a selected area of an EPROM with data in the buffer.
j. SELECT ..... a new EPROM type (return to types menu).
k. ENTER ..... the system monitor.
l. RETURN ..... to the operating system.
m. EXECUTE ..... any DOS utility (only in FLEX and OS9 versions).

FLEX and OS9 VERSIONS AVAILABLE FROM GIMIX. SSB/MDOS CONTACT US DIRECT.

PL/9

- Friendly inter-active environment where you have INSTANT access to the Editor, the Compiler, and the Trace-Debugger, which, amongst other things, can single step the program a SOURCE line at a time. You also have direct access to any FLEX utility and your system monitor.
- 375* page manual organized as a tutorial with plenty of examples.
- Fast SINGLE PASS compiler produces 8K of COMPACT and FAST 6809 machine code output per minute with no run-time overheads or license fees.
- Fully compatible with TSC text editor format disk files.
- Signed and unsigned BYTES and INTEGERS, 32-bit floating point REALs.
- Vectors (single dimension arrays) and pointers are supported.
- Mathematical expressions: (+), (-), (*), (/), modulus (\), negation (-)
- Expression evaluators: (=), (<>), (<), (>), (>=), (<=)
- Bit operators: (AND), (OR), (EOR/XOR), (NOT), (SHIFT), (SWAP)
- Logical operators: (.AND), (.OR), (.EOR/XOR)
- Control statements: IF..THEN..ELSE, IF..CASE1..CASE2..ELSE, BEGIN..END, WHILE.., REPEAT..UNTIL, REPEAT..FOREVER, CALL, JUMP, RETURN, BREAK, GOTO.
- Direct access to (ACCA), (ACCB), (ACCD), (XREG), (CCR) and (STACK).
- FULLY supports the MC6809 RESET, NMI, FIRQ, IRQ, SWI, SWI2, and SWI3 vectors. Writing a self-starting (from power-up) program that uses ANY, or ALL, of the MC6809 interrupts is an absolute snap!
- Machine code may be embedded in the program via the 'GEN' statement. This enables you to code critical routines in assembly language and embed them in the PL/9 program (see 'MACE' for details).
- Procedures may be passed and may return variables. This makes them functions which behave as though they were an integral part of PL/9.
- Several fully documented library procedure modules are supplied: IOSUBS, BIFIO, HARDIO, HEXIO, FLEXIO, SCIPACK, STRSUBS, BASTRING, and REALCOM.

'... THIS IS THE MOST EFFICIENT COMPILER I HAVE FOUND TO DATE.'

Quoted from Ron Anderson's FLEX User Notes column in '68. Need we say more?

MACE/XMACE/ASM05

All of these products feature a highly productive environment where the editor and the assembler reside in memory together. Gone are the days of tedious disk load and save operations while you are debugging your code.

- Friendly inter-active environment where you have instant access to the Editor and the Assembler, FLEX utilities and your system monitor.
- MACE can also produce ASMPROCs (GEN statements) for PL/9 with the assembly language source passed to the output as comments.
- XMACE is a cross assembler for the 6800/1/2/3/8 and supports the extended mnemonics of the 6303.
- ASM05 is a cross assembler for the 6805.

D-BUG

LOOKING for a single step tracer and mini in-line disassembler that is easy to use?? Look no further, you have found it. This package is ideal for those small assembly language program debugging sessions. D-BUG occupies less than 6K (including its stack and variables) and may be loaded anywhere in memory. All you do is LOAD IT, AIM IT and GO! (80 col VDUs only).

McCOSH 'C'

This is as complete a 'C' compiler as you will find on any operating system for the 6809. It is completely compatible with UNIX VII and only lacks 'bit-fields' (which are of little practical use in an 8-bit world!).

- Produces very efficent assembly language source output with the 'C' source optionally interleaved as comments.
- Built-in optimizer will shorten object code by about 11%.
- Supports interleaved assembly language programs.
- INCLUDES its own assembler. The TSC relocating assembler is only required if you want to generate your own libraries.
- The pre-processor, compiler, optimizer, assembler and loader all run independantly or under the 'CC' executive. 'CC' makes compiling a program to executable object as simple as typing in 'CC,HELLO.C <RETURN>'.

IEEE-488

- SUPPORTS ALL PRINCIPAL MODES OF THE IEEE-488 (1975/8) BUS SPECIFICATION:
  - Talker - Serial Poll - Single or Dual Primary Address
  - Listener - Parallel Poll - Secondary Address
  - System Controller - Group Trigger - Talk only ... Listen only
- Fully documented with a complete reprint of the KILOBAUD article on the IEEE bus and the Motorola publication 'Getting aboard the IEEE Bus'.
- Low level assembly language drivers suitable for 6800, 6801, 6802, 6803, 6808 and 6809 are supplied in the form of listings. A complete back to back test program is also supplied in the form of a listing. These drivers have been extensively tested and are GUARANTEED to work.
- Single S-30 board (4, 8 or 16 addresses per port), fully socketed, gold plated bus connectors and IEEE interface cable assembly.

PRICES

D-BUG	(6809 FLEX only)	$ 75.00
MACE	(6809 FLEX only)	$ 75.00
XMACE	(6809 FLEX only)	$ 98.00
ASM05	(6809 FLEX only)	$ 98.00
PL/9	(6809 FLEX only)	$198.00
'C'	(6809 FLEX only)	$295.00
IEEE-488	with IEEE-488 cable assembly	$298.00
UPROM-II/U	with one version of software (no cable or interface)	$395.00
UPROM-II/C	as above but complete with cable and S-30 interface	$545.00
CABLE	5' twist-n-flat 50 way cable with IDC connectors	$ 35.00
S-30 INT	SS-30 interface for UPROM-II	$130.00
EXOR INT	Motorola EXORbus (EXORciser) interface for UPROM-II	$195.00
UPROM SFT	Software drivers for 2nd operating system. Specify FLEX or OS9 AND disk size!	$ 35.00
UPROM SRC	Assembly language source (contact us direct)	

ALL PRICES INCLUDE AIR MAIL POSTAGE

Terms: CWO. Payment by Int'l Money Order. VISA or MASTER-CARD also accepted.

WORSTEAD LABORATORIES, NORTH WALSHAM, NORFOLK, ENGLAND. NR28 9SA.

TEL: 44 (692) 404086
TLX: 975548 WMICRO G

WE STOCK THE FOLLOWING COMPANIES PRODUCTS: GIMIX, SSB, FHL, MICROWARE, TSC, LUCIDATA, LLOYD I/O, & ALFORD & ASSOCIATES.

FLEX (tm) is a trademark of Technical Systems Consultants, OS-9 (tm) is a trademark of Microware Systems Corporation, MDOS (tm) and EXORciser (tm) are trademarks of Motorola Incorporated.

DYNACALC®

CoCo's Best & Fastest Spreadsheet System

ACCLAIMED BY THE EXPERTS

"DYNACALC is my choice for a CoCo spreadsheet."
Dan Downard, RAINBOW, September, 1984.

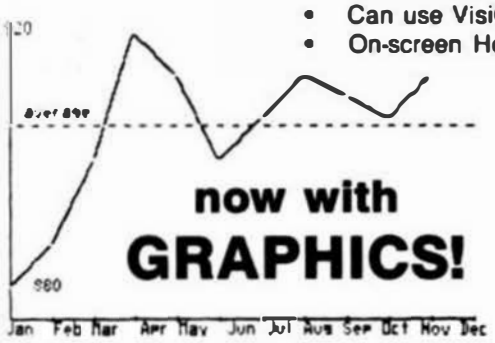
"Eat your heart out, Lotus 1-2-3!"
Scott Norman, HOT CoCo, October, 1984.

NOW ONLY $99.95

Built-in Features:

- 51 x 24 Display with Lower Case
- Super-fast Smart Screen Refresh
- Auto-Repeat Keyboard Driver
- Keysaver (TM) repeats last command x times
- Disk Operating System (works just like ROM DOS)
- Easy communication with BASIC/DOS programs
- Two-way communications with PRO-COLOR-FILE ★Enhanced★
- Outputs to ASCII Word Processors like Telewriter-64
- Fast 16-Digit Arithmetic with Scientific Functions
- Summation, Mean, and Standard Deviation Functions
- Logical Functions with String & Numeric Comparison
- String locate command to navigate large worksheets
- Sort full or partial worksheet by columns or rows
- Line, Bar, Hi/Lo/Close, Circle Graphs
- Full Graphics captioning and overlay facility
- Graphics Drivers for all popular Printers
- Joystick/Mouse Driver for Cursor Movement
- Works with any ROM versions, even JDOS
- 33k Available Worksheet Space
- Up to 256 Columns or 256 Rows
- Can use VisiCalc worksheets & training materials
- On-screen Help Messages

RAINBOW CERTIFICATION SEAL



now with GRAPHICS!

FOR 64K DISK SYSTEMS
available from

COMPUTER SYSTEMS CENTER
42 Four Seasons Center #122
Chesterfield, MO 63017
(314) 576-5020
10 to 6 Mon.-Fri.

NEW ADDRESS

CSC

or your local DYNACALC dealer

Price US$99.95
Outside North America add $10 postage

CANADA- CDN$129.95
RGS MICRO INC.
759, VICTORIA SQUARE 405
MONTREAL H2Y 2J3
TEL.: (514) 287-1563
ORDER LINE ONLY ★ ★ ★
QUEBEC-ONTARIO-MARITIMES
800-361-5338
WESTERN CANADA 800-361-5155

MasterCard VISA

DYNACALC Reg. U. S. Pat. Off.
Lotus and 1-2-3 are trademarks of Lotus Development Corp.
PRO-COLOR-FILE is a trademark of Derringer Software
Telewriter-64 is a trademark of Cognitec
VisiCalc is a trademark of VisiCorp.

68' MICRO JOURNAL ADVERTISERS INDEX

This index is provided as a reader service. The publisher does not assume any liability for omissions or errors.

GIMIX HAS THE 6809 SYSTEM TO SUIT YOUR NEEDS

HARDWARE

All systems feature the **GIMIX CLASSY CHASSIS**; with a ferro-resonant constant voltage power supply, gold plated bus connectors, and plenty of capacity for future expansion.

Static **RAM** and double-density **DMA** floppy disk controllers are used exclusively in all systems.

All systems are guaranteed for 2 MHz operation and include complete hardware and software documentation, necessary cables, filler plates, etc.

Systems are assembled using burned-in and tested boards, and all disk drives are tested and aligned by **GIMIX**.

You can add additional components to any system when ordering, or expand it in the future by adding **RAM, I/O,** etc.

GIMIX lets you choose from a wide variety of options to customize your system to your needs.

SOFTWARE

All **OS-9/FLEX** systems allow you to software select either operating system.

Also included is the **GMXBUG** monitor and, in systems with 128K or more of RAM, **GMX-VDISK** for **FLEX**.

All **GIMIX OS-9** systems include **Microware's Editor, Assembler, Debugger, Basic09,** and **Runb**; and the GMX versions of **RMS** and **DO** for **OS-9**.

All **GIMIX** versions of **OS-9** can read and write RS color computer format OS-9 disks, as well as the Microware/GIMIX standard format.

New and exclusive with **OS-9 GMX III** systems is the **GMX OS-9 Support ROM**, a monitor for **OS-9** that includes memory diagnostics and allows the system to boot directly from either hard disk or floppy.

A wide variety of languages and other software is available for use with either **OS-9** or **FLEX**.

OS-9 GMX III/FLEX SYSTEMS (#79)

The **#79** super system now includes (in addition to the above): the **GMX 6809 CPU III, a 256K CMOS Static RAM Board (#72), and a 3-port Intelligent Serial I/O Processor (#11)**.

The **GMX 6809 CPU III** can perform high-speed DMA transfers from memory to memory and uses memory attributes and illegal instruction trapping to protect the system and users from program crashes. If a user program crashes, only that user is affected; other users are unaware of the problem.

The 3-Port Intelligent Serial I/O Board (#11) significantly reduces system overhead by handling routine I/O functions; freeing the host CPU for running user programs. This improves overall system performance and allows user terminals to be run at up to 19.2K baud.

with dual 40 track DSDD drives	**$5998.79**
with dual 80 track DSDD drives	**$6198.79**
with **#88** dual 8" DSDD drive system	**$7698.79**
with **#90** 19MB Winchester subsystem and one 80 track	**$8898.79**
with a 47MB Winchester subsystem and one 80 track	**$10,898.79**
with a 47MB plus a 6MB removable pack Winchester subsystem and one 80 track drive	**$12,398.79**

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account #73-32033.

OS-9 GMX I / FLEX SYSTEMS #49

The **#49** systems include 64KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	**$3998.49**
with dual 80 track DSDD drives	**$4198.49**
with **#88** dual 8" DSDD drive system	**$5698.49**
with **#90** 19MB Winchester subsystem and one 80 track	**$6898.49**

OS-9 GMX II / FLEX SYSTEMS #39

The **#39** systems include 128KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	**$4498.39**
with dual 80 track DSDD drives	**$4698.39**
with **#88** dual 8" DSDD drive system	**$6198.39**
with **#90** 19MB Winchester subsystem and one 80 track	**$7398.39**

GIMIX DOES NOT GUARANTEE PERFORMANCE OF ANY GIMIX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

EXPORT MODELS: ADD $30 FOR 50Hz. POWER SUPPLIES.

GIMIX, Inc. reserves the right to change pricing, terms, and products specifications at any time without further notice.

ALL PRICES ARE F.O.B. CHICAGO

Contact GIMIX for price and availability of UniFLEX and UniFLEX GMXIII Systems.

NOTE on all drive systems: Dual 40 track drives have about 700KB of formatted capacity; dual 80's about 1,400KB; dual 8" about 2,000KB. The formatted capacity of hard disks is about 80% of the total capacity.

Want to expand your system to a megabyte of Static RAM and 15 users?

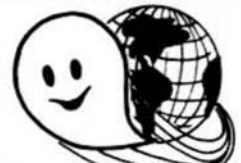
Simply add additional memory and I/O boards. Your GIMIX system can grow with your needs. Contact us for a complete list of available boards and options.

#72 256KB CMOS STATIC RAM board with battery back up	**$1898.72**
#64 64KB CMOS STATIC RAM board with battery back up	**$528.64**
#67 64KB STATIC RAM board	**$478.67**
#11 3 port Intelligent serial I/O board	**$498.11**
#43 2 port serial I/O board	**$128.43**
#42 2 port parallel I/O board	**$88.42**
#95 cable sets (1 needed per port), specify board	**$24.95**

NOW SHIPPING !

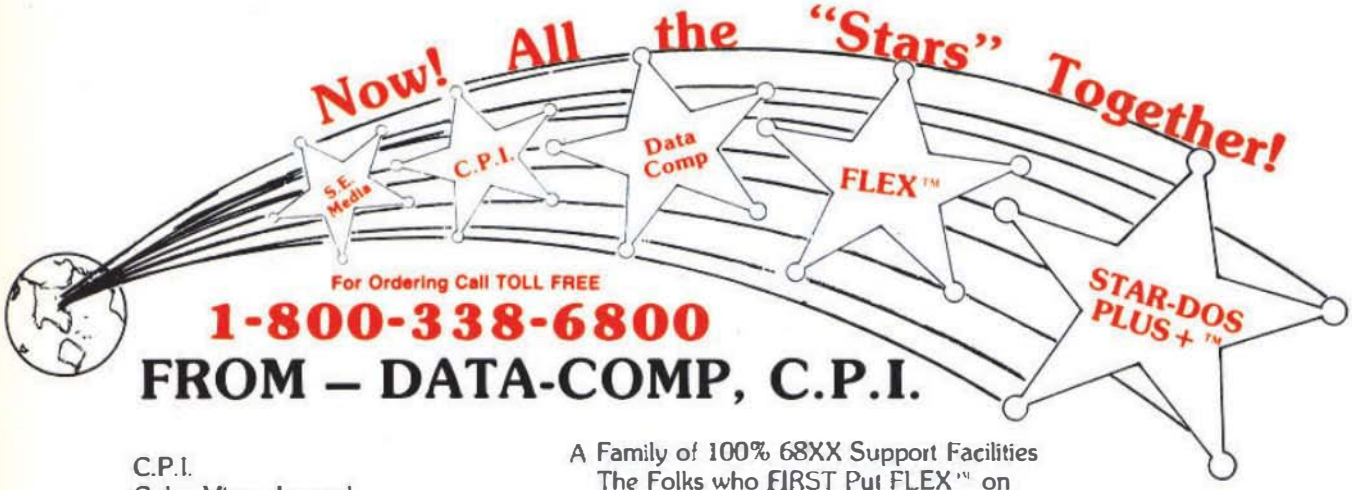
UniFLEX

GMX III Systems



GIMIX INC.

1337 WEST 37th PLACE
CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055



For Ordering Call TOLL FREE

1-800-338-6800

FROM – DATA-COMP, C.P.I.

C.P.I.
Color Micro Journal
'68' Micro Journal
Data-Comp
S.E. Media

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo
Now Offering: *FLEX™ (2 Versions)
AND *STAR-DOS PLUS+™

FLEX-CoCo Sr.
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. $250.00 **Only $79.95**

STAR-DOS PLUS+
- Functions Same as FLEX
- Reads - writes FLEX Disks
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

$34.50

FLEX-CoCo Jr.
without TSC Editor & Assembler
$49.95

PLUS

ALL VERSIONS OF FLEX & STAR-DOS+ INCLUDE

+ Read-Write-Dir RS Disk
+ Run RS Basic from Both
+ More Free Utilities
+ Super 800 Support
+ Free Color Micro Journal 1 yr. sub.
+ External Terminal Program
+ Test Disk Program
+ Disk Examine & Repair Program
+ Memory Examine Program
+ Many Many More!!!

TSC Editor
Reg $50.00
NOW $35.00

TSC Assembler
Reg $50.00
NOW $35.00

DISK SYSTEMS FOR THE COLOR COMPUTER

THESE PACKAGES INCLUDE DRIVE, *CONTROLLER, POWER SUPPLY & CABINET, CABLE, AND MANUAL.

* SPECIFY WHAT CONTROLLER YOU WANT J&M, OR RADIO SHACK.

PAK #1 - 1 SINGLE SIDED, DOUBLE DENSITY SYS.	$389.95
PAK #2 - 2 SINGLE SIDED, DOUBLE DENSITY SYS.	$639.95
PAK #3 - 1 DOUBLE SIDED, DOUBLE DENSITY SYS.	$439.95
PAK #4 - 2 DOUBLE SIDED, DOUBLE DENSITY SYS.	$699.95
PAK #5 - 2 DOUBLE SIDED, DOUBLE DENSITY SYS. THINLINE DRIVES, HALF SIZE	$659.95
COLOR COMPUTER II 64K W/EXT. BASIC	$189.95

CONTROLLERS

J&M DISK CONTROLLER W/ JDOS OR RADIO SHACK DISK BASIC, SPECIFY WHAT DISK BASIC.	$139.95
RADIO SHACK DISK CONTROLLER 1.1	$134.95

DISK DRIVE CABLES

CABLE FOR ONE DRIVE	$ 19.95
CABLE FOR TWO DRIVES	$ 24.95

MISC

64K UPGRADE W/MOD. INSTRUCTIONS, C,D,E,F, AND COCO 2	$ 49.95
HJL KEYBOARDS	$ 69.95
MICRO TECH LOWER CASE ROM ADAPTER	$ 74.95
RADIO SHACK BASIC 1.2	$ 29.95
RADIO SHACK DISK BASIC 1.1	$ 29.95
RADIO SHACK EXT. BASIC	$ 39.95
SCREEN CLEAN CLEARS UP VIDEO DISTORTION	$ 39.95
MEMOREX DISKS 5" SS,DD	$ 24.00
SHIPPING INCLUDED ON DISK PRICES	
DISK DRIVE CABINET & POWER SUPPLY	$ 49.95
SINGLE SIDED, DOUBLE DENSITY 5" DISK DRIVE	$199.95
DOUBLE SIDED, DOUBLE DENSITY 5" DISK DRIVE	$249.95

PRINTERS

EPSON RX-80	$325.00
EPSON RX-80FT	$375.00
EPSON MX-100	$650.00
EPSON FX-100	$799.00
EPSON FX-80	$549.00
EPSON MX-70	$200.00

SERIAL BOARDS FOR PRINTERS

MX-SERIES	$119.95
FX-SERIES	$ 99.95

USA ADD 2% SHIPPING
FOREIGN ADD 5% SHIPPING

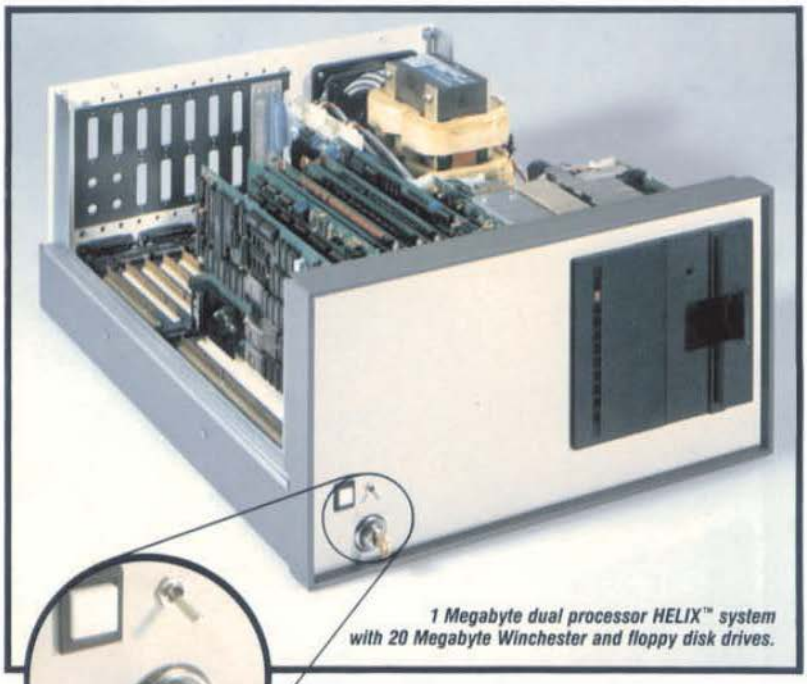
SPECIAL MX-100 $550.00

*FLEX is a Trademark of Technical System Consultants
*STAR-DOS+ is a Trademark of STAR-Kits & Data-Comp

5900 Cassandra Smith Rd. Hixson, TN 37343

68 Micro Journal
ISSN 0194-5025
5900 Cassandra Smith Rd
Hixson, TN 37343

Second Class Postage Paid
At Hixson, TN
And Additional Mailing Offices



1 Megabyte dual processor HELIX™ system with 20 Megabyte Winchester and floppy disk drives.

***ONCE AGAIN HAZELWOOD COMPUTER SYSTEMS** demonstrates its leadership in computer technology by delivering the only computer system capable of switching between either the 6809 or the 68000 processor. Switching is easily accomplished by a simple front panel toggle switch. The reason we can offer this exclusive feature now, is that when our proven 6809 processor board was designed several years ago, we had the foresight to include the bus controls that allow processor switching.*

Hazelwood Computer Systems is also proud to be the first S-50/S-64 bus manufacturer to license and deliver the OS9/68K Operating System from Microware Systems Corporation. OS9/68K is the 68000 version of the popular and powerful OS9 Operating System. Utilizing our proven MC-20 disk controller, OS9/68K can conveniently share a Winchester disk with OS9. Changing from 6809 to 68000 operation is as simple as switching processors and booting the new system from the Winchester disk.

The ease of switching processors and operating systems makes a HELIX™ dual processor system the natural choice for software development. In addition, the advanced design of HELIX™ equipment, emphasizing performance and reliability, makes HELIX™ boards and systems the best value in computing offered anywhere.

System prices vary with configuration. Call for exact pricing.

THE SWITCH IS ON...

The CP-08 processor board utilizes a 68008 processor running at 10 Mhz clock rate. Using proprietary bus synchronization circuitry and single cycle DMA, the CP-08 achieves a marked performance increase over a 2 MHz 6809. Offering absolute compatibility with the 68000 instruction set, the 68008 addresses up to 1 Megabyte of memory. Also included on the CP-08 are up to 4K of ROM, an interrupt timer, and, with battery backup operation, a clock/calendar and 2K RAM. Implemented as a standard S-50 board, the CP-08 brings 68000 operation to S-50 bus computers.

ORDER: CP-08 **PRICE: $595**

The MC-20 Mass Storage Controller board interfaces up to 4 floppy and 8 Winchester disk drives to the S-50/S-64 bus. The MC-20 is an intelligent controller with its own 2 Mhz 6809 processor and 56K RAM. It provides DMA data transfers to a full 24 bit address. All disk operation requests are by logical block number, with the controller performing the necessary track/sector address calculations. Any combination of 5¼ or 8 inch floppy drives can be accommodated with all drive parameters, such as write precompensation, software controlled for each individual drive. Winchester drives are connected via a SASI bus interface. Block address mapping is provided which allows a single drive to be segmented into several logical units. The MC-20 is the controller of the MS-20 Mass Storage Subsystem which includes a 20 Megabyte Winchester drive.

ORDER: MC-20 **PRICE: $695**

OS9/68K

OS9/68K offers increased performance and larger user memory space while retaining all of the features of OS9. Disk file compatibility and operational similarity assures that present OS9 users can easily transfer their operations to the 68000. Included are an editor, assembler, linker, and debugger. A C compiler is available now. BASIC09 and other languages will be avilable soon.

ORDER: OS9/68K **PRICE: $250**

All items available stock to 30 days. Prices subject to change without notice.

HAZELWOOD COMPUTER SYSTEMS

907 East Terra, O'Fallon, MO 63366, **314-281-1055**

HELIX™