# 68' MICRO JOURNAL

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

$2.95USA

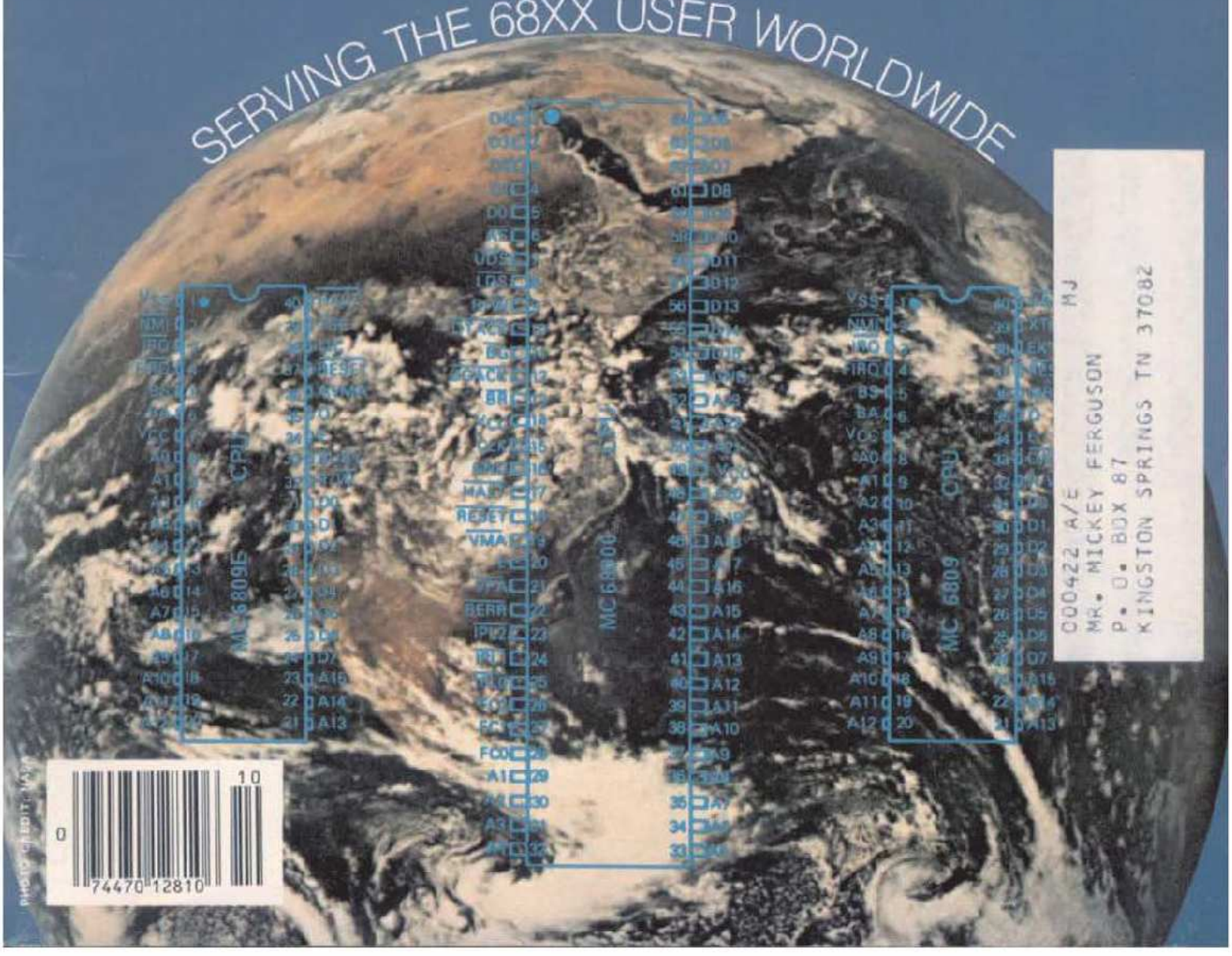
**68000**

68000 USER Notes p. 22
RDA and the 68000 p.17

**6809**

"C" USER Notes p.12
OS/9 USER Notes p.7
FLEX USER Notes p.5
Also: Basic OS-9, PL/9, CoCo Reviews

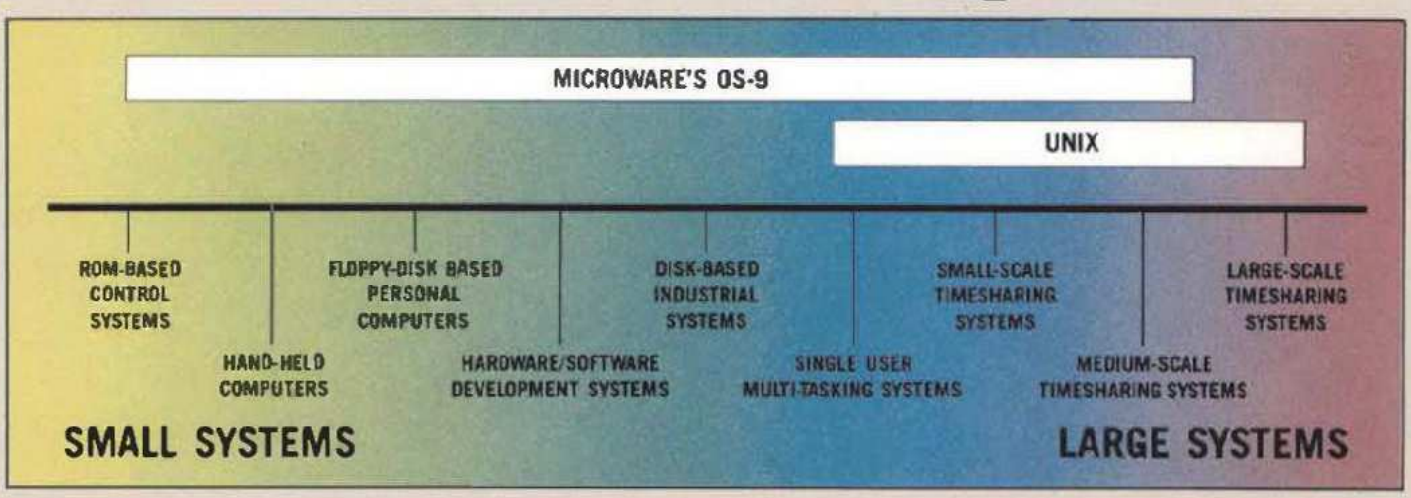
VOLUME VII ISSUE X • Devoted to the 68XX User • October 1985
"Small Computers Doing Big Things"




CO0422 A/E MJ
MR. MICKEY FERGUSON
P. O. BOX 87
KINGSTON SPRINGS TN 37082

0 74470 12810 10

# 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 operatingsystem 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

AUTHORIZED MICROWARE DISTRIBUTORS

**AUSTRALIA**
MICROPROCESSOR CONSULTANTS
16 Bandera Avenue
Wagga Wagga 2650
NSW Australia
phone: 069-31-2331

**ENGLAND**
VIVAWAY LTD.
36-38 John Street
Luton, Bedfordshire
England LU1 2JE
phone: (0582) 423425
telex: 825185

**JAPAN**
MICROWARE JAPAN LTD.
3-8-9 Baraki, Ichikawa
Chiba 272-01, Japan
phone: 0473 (28) 4493
telex: 788-299-3122

**SWEDEN**
MICROMASTER SCANDINAVIAN AB
Sit Persgatan 7
Box 1309
S751-43 Uppsala
Sweden
phone: 018-138595
telex: 76129

**SWITZERLAND**
ELSOFT AG
Bankstrasse 9
5432 Neuenhof
Switzerland
phone: (41) 056-862724
telex: 59136

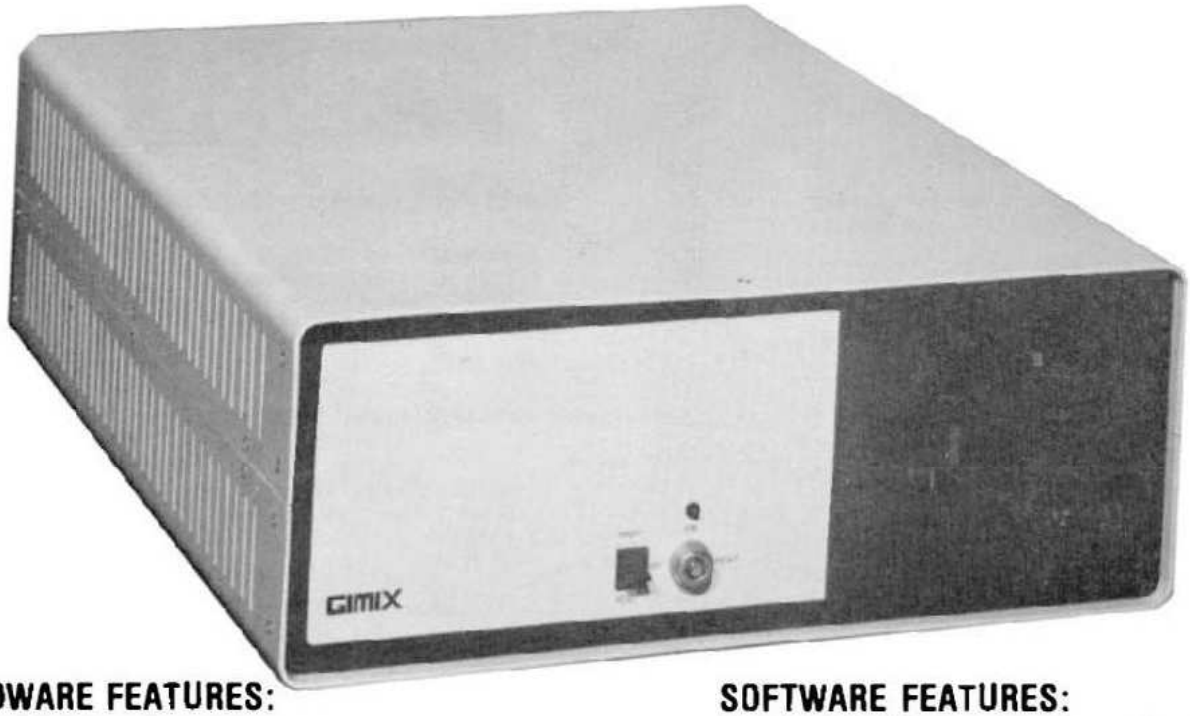
**USA**
MICROWARE SYSTEMS CORPORATION
1866 NW 114th Street
Des Moines, Iowa 50322
USA
phone: 515-224-1929
telex: 910-520-2535
FAX: 515-224-1352

**WEST GERMANY**
DR. KEIL GMBH
Porphystrasse 15
D-6905 Schriesheim
West Germany
phone: (0 62 03) 67 41
telex: 465025

OS-9 is a trademark of Microware and Motorola. Unix is a trademark of Bell Labs.

# GMX 68020 DEVELOPMENT SYSTEM

A Multi-user, Multi-tasking software development system for use with all 68000 family processors.




## HARDWARE FEATURES:

- The GMX-020 CPU board has: the MC68020 32-bit processor, a 4K Byte no wait-state instruction cache, high-speed MMU, and a full-featured hardware time of day clock/calendar with battery back-up. It also provides for an optional 68881 floating point co-processor.
- 1 Megabyte of high speed static RAM.
- Intelligent Serial and Parallel I/O Processor boards significantly reduce system overhead by handling routine I/O functions. This frees up the host CPU for running user programs. The result is a speed up of system performance and allows all terminals to run at up to 19.2K baud.
- The system hardware will support up to 39 terminals.
- Powered by a constant voltage ferro-resonant power supply that insures proper system operation under adverse AC power input conditions.
- DMA hard disk interface and DMA double density floppy disk controller are used for data transfers at full bus speed. The DMA hard disk drive controller provides automatic 22-bit burst data error detection and 11-bit burst error correction.
- A selection of hard disk drives with capacities from 19 to 85 Megabytes, removeable pack hard disk drives, streaming tape drives, and floppy disk drives is available.

## SOFTWARE FEATURES:

The UniFLEX VM Operating System is a demand-paged, virtual memory operating system written in 68020 Assembler code for compactness and efficiency. Any UniFLEX system will run faster than a comparable system written in a higher level language. This is important in such areas as context switching, disk I/O, and system call handling. Other features include:

- compact, efficient Kernel and modules allows handling more users more effectively than UNIX systems, using much less disk space.
- UNIX system V compatibility at the C source code level.
- C Compiler optimized in 68020 code (optional).
- Record locking for shared files.
- Users can share programs in memory.
- Modeled after UNIX systems, with similar commands.
- System accounting facilities.
- Sequential and random file access.
- Maximum record size limited only by the disk size.
- Multiple Level Directories.
- Up to 4 Megabytes of Virtual Memory per user.
- Optional Languages available are: C, BASIC, COBOL, FORTRAN, LISP, PROLOG, SCULPTOR, and ASSEMBLER. In development are ADA, PASCAL, and FORTH.

Included with the UniFLEX Operating System are a Utilities package, editor, relocating assembler, linking loader, and printer spooler. Options inlcude a fast floating point package, library generator, and a sort-merge package.

The GMX version of the MOTOROLA 020 BUG is included with the system.

UNIX is a trademark at A.T. & T.
ADA is a trademark of the U.S. Government.
UniFLEX is a trademark of Technical Systems Consultants, Inc.
GMX and GIMIX are trademarks of GIMIX, Inc.

GIMIX, Inc., a Chicago based microcomputer company established in 1975, has produced state of the art microcomputer systems based on Motorola 6800 and 6809 microprocessors. GIMIX systems are in use in Industry,, Hospitals, Universities, Research Organiations, and by Software Developers. GIMIX was awarded the prestigious President's "E" Certificate for Exports in 1984.

GIMIX INC. 1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609 • (312) 927-5510 • TWX910-221-4055

# '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 - COS1 - 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

**Editorial Staff**

| | |
|---|---|
| Don Williams Sr. | Publisher |
| Larry E. Williams | Executive Editor |
| Tom E. Williams | Production Editor |
| Robert L. Ney | Technical Editor |

**Administrative Staff**

| | |
|---|---|
| Mary Robertson | Office Manager |
| Penny Williams | Subscriptions |
| Christine Kocher | Accounting |

**Contributing Editors**

| | |
|---|---|
| Ron Anderson | Norm Conmo |
| Peter Dibble | William E. Fisher |
| Dr. Theo Elbert | Carl Mann |
| Dr. E. M. Pass | Ron Voigts |
| Philip Lucido | |

**Special Technical Projects**
Clay Abrams K6AEP
Tom Hunt

## CONTENTS

Vol.VII, Issue X October 85

FLEX USER Notes..... 5 Anderson
OS9 USER Notes..... 7 Dibble
C USER Notes..... 12 Pass
ADA And The 68000..... 17 Elbert
Basic OS-9..... 19 Voigts
68000 USER Notes..... 22 Lucido
Pleasant PL/9..... 24 Lester
CoCo USER Notes..... 25 Mann
OS-9 Setime..... 26 Groves
COBOL Name & Address System. 29 Martin
Bit Bucket..... 37
Disk Sector Interleave..... 38 Taylor
Rambling Around NCC..... 41 DMW
CoCo PASCAL OS/9 Review..... 52 Voights
Classified Advertising..... 53

**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

**Copyrighted 1985 by Computer Publishing Inc.**

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 $9.50 a Year. Other Foreign Add $12 a Year for Surface, Airmail Add $48 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, .pg 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 advertisers please contact 68' Micro Journal advertising department for current rate sheet and requirements.

**Classified Advertising**

All classified ads must be non-commercial. Minimum of $9.50 for first 20 words and .45 per word after 20. All classifieds must be paid in advance. No classified ads accepted over the phone.

# ..HEAR YE.....HEAR

# OS-9™ User Notes

By: Peter Dibble
As Published in 68 Micro Journal

The publishers of 68 Micro Journal are proud to announce the publication of Peter Dibbles **OS9 USER NOTES**.

**Information for the BEGINNER to the PRO, Regular or CoCo OS9**

**Using OS9**
HELP, HINTS, PROBLEMS, REVIEWS, SUGGESTIONS, COMPLAINTS, OS9 STANDARDS, Generating a New Bootstrap, Building a new System Disk, OS9 Users Group, etc.

**Program interfacing to OS9**
DEVICE DESCRIPTORS, DIRECTORIES, "FORKS", PROTECTION, "SUSPEND STATE", "PIPES", "INPUT/OUTPUT SYSTEM", etc.

**Programming Languages**
Assembly Language Programs and Interfacing; Basic09, C, Pascal, and Cobol reviews, programs, and uses; etc.

**Disks Include**
**No typing all the Source Listings in.** Source Code and, where applicable, assembled or compiled **Operating Programs.** The Source and the Discussions in the Columns can be used "as is", or as a "Starting Point" for developing **your OWN** more powerful Programs. Programs sometimes use multiple Languages such as a short Assembly Language Routine for reading a Directory, which is then "piped" to a Basic09 Routine for output formatting, etc.

**BOOK** Typeset -- w/ Source Listings (3-Hole Punched; 8 x 11) **$9.95**

**Deluxe Binder - - - - - - - - - - - $5.50**

All Source Listings on Disk
1 8" SS, SD Disk - - - - $14.95
2 5" SS, DD Disks - - - $24.95

Shipping and Handling; $3.50 per Book, $2.50 per Disk Set
* All Currency in U.S. Dollars Foreign Orders Add $4.50 S/H
If paying by check - Please allow 4-6 weeks delivery

## Continually Updated In 68 Micro Journal Monthly

VISA

Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN. 37343

(615) 842-4600
Telex 558 414 PVT BTH

TM - OS9 is a trademark of Microware Systems Corp. and Motorola Inc.
TM - 68' Micro Journal is a trademark of Computer Publishing Inc.

More help than any other thing I have ever read, except 68 Micro Journal, of course. Come on with the C book on Norm Commo's column. BD

# FLEX™ USER NOTES
## THE 6800-6809 BOOK

SWELL! - Best buy EVER in a 6800/6809 book. JTW

**By: Ronald W. Anderson**
As published in 68 MICRO JOURNAL™

The publishers of 68 MICRO JOURNAL are proud to announce the publication of Ron Anderson's **FLEX USER NOTES,** in book form. This popular monthly column has been a regular feature in 68 MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. Now all his columns are being published, in whole, as the most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

As a SPECIAL BONUS all the source listing in the book will be available on disk for the low price of: FLEX™ format only — 5" $12.95 — 8" $16.95 plus $2.50 shipping and handling, if ordered with the book. If ordered separately the price of the disks will be: 5" $17.95 — 8" $19.95 plus $2.50 shipping and handling.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks.

| | |
|---|---|
| LOGO.C1 | File load program to offset memory — ASM PIC |
| MEMOVE.C1 | Memory move program — ASM PIC |
| DUMP.C1 | Printer dump program — uses LOGO — ASM PIC |
| SUBTEST.C1 | Simulation of 6800 code to 6809, show differences — ASM |
| TERMEM.C2 | Modem input to disk (or other port input to disk) — ASM |
| M.C2 | Output a file to modem (or another port) — ASM |
| PRINT.C3 | Parallel (enhanced) printer driver — ASM |
| MODEM.C2 | TTL output to CRT and modem (or other port) — ASM |
| SCIPKG.C1 | Scientific math routines — PASCAL |
| U.C4 | Mini-monitor, disk resident, many useful functions — ASM |
| PRINT.C4 | Parallel printer driver, without PFLAG — ASM |
| SET.C5 | Set printer modes — ASM |
| SETBAS1.C5 | Set printer modes — A-BASIC |
| | (And many more) |

**Over 30 TEXT files included in ASM (assembler) — PASCAL — PIC (position independent code) TSC BASIC-C, etc.

NOTE: .C1,.C2, etc. = Chapter 1, Chapter 2, etc.

This will be a limited run and we cannot guarantee that supplies will last long. Order now for early delivery.

Finally I don't have to type in all that source code, a GREAT, GREAT idea. Thanks and good luck, you guys are the best! TOA

Foreign Orders Add $4.50 S/H

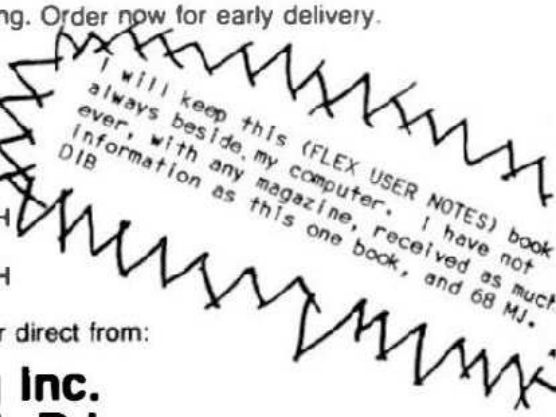
Softcover — Large Format

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H

I will keep this (FLEX USER NOTES) book always beside my computer. I have not ever, with any magazine, received as much information as this one book, and 68 MJ. DIB



See your local S50 dealer/bookstore or order direct from:

**Computer Publishing Inc.**
**5900 Cassandra Smith Rd.**
**Hixson, TN 37343**
**(615) 842-4601**

**TELEX 558 414 PVT BTH**

™FLEX is a trademark of Technical Systems Consultants

# FLEX User Notes

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

### Editors

Here are some random thoughts precipitated by the search for a suitable screen editor for the company's Tandy 1200-HD. We now have purchased three screen editors and have one "freeware" version. My search for one that suits me is not yet at an end.

I have some ideas about what an editor should be and what it should not be. First of all, like any computer program it should serve me, not the other way around. The editor should make creation and changing of a text file or a program as easy as possible. It should be able to handle paragraphs of text as well as formatted program listings. It should minimize the keystrokes necessary to perform any function. I am going to live with it for a long time so I don't mind if it takes a while to learn all of its features, as long as I can learn the most used ones quickly. It should be convenient and logical to use.

Two of our recent purchases require the user to "insert" blank lines when he bumps the end of the text so that there are more lines available to be edited. That in my opinion is absolute nonsense. The editor ought to know there is no more text below the present point, and it should handle that without the user ever knowing about it. One of our 6809 editors works that way. You can just move the cursor down 12 lines below the last line of text, go to the middle of the screen and type whatever you want. The software inserts blank lines (CR) and spaces over to the cursor before inserting the text that you typed. Such operation ought to be standard with editors.

Of the four editors that we purchased, two will do on-screen formatting of paragraphs. Neither adequately handles embedded printer codes for underline or boldface. The two that do not do formatting on screen are usable by a text processor that does handle embedded codes conveniently. It should be possible to have both formatting and embedded codes.

One of the four is very complete, very large, and complex to use. It requires many more keystrokes to accomplish the same things as some of the others. (Though, to be fair, you can create your own key macro definitions to make one control key do just what you need.) It is the most useful of the four. It has a little command language built in by which you can program any key to perform a string of functions. For example, it lacks what I would call auto indent mode. When I write a program I like the mode in which the cursor aligns itself with the first non-blank on the preceding line when you type a CR. With the editor in question, you can program a function key or a control key to do a CR, move the cursor up one line, move right until it finds the first non-space, and move down a line. That is exactly what Auto Indent mode should do. Now if you want to stay at the same indent level, you start typing. If not you tab or backtab one or more indent levels to go where you want to be.

One of the editors was advertised as "top quality software" at a very reasonable price. The price was fine, but that is where the problems began. The license agreement was totally absurd. "You promise not to copy this software." That is what it said, no exceptions. I had the choice of sending it back unopened, but I read the manual a while and it contained explicit instructions to make a backup copy of the supplied disk (which was copy protected by means of a KEY that had to be read from the original disk in order to run the software). The instructions for running it from or with a hard disk were obscure, but I finally figured out that I could operate it from a hard disk, but it does require the original disk to be in drive A on the Tandy in order to run it. While I have no quarrel with protection schemes, the Ad said nothing whatever about protection. The license agreement was contradicted in the instructions. The manual contained a list of files that were supposed to be on the disk. Several of those were not, and consequently I could not install any other printer than the one for which it is originally set up.

I've complained loud and long about that one in a letter to the supplier, but it is too early to have received any sort of response. What I am getting at, is that we FLEX (and OS-9) users are still pretty well off. When those other guys tell you "look at all the software we have available, compared to what you have." Just say, that what they say is true, but quantity and quality are two different things.

When the other guys argue that they can run with larger memory space, remember that it takes them twice as much memory to run the same program as we can run in our 6809 systems. By actual count in my recent translation of JUST to "C", McCosh "C" for the 6809 compiled just under 8K bytes of output code, and Lattice C on the Tandy compiled very nearly 15K for the same program (five lines different out of 650 or so). That very capable editor mentioned above is over 100K of code. Of the four, the smallest pushes 40K.

That brings me to thoughts of my editor project. It is nearing completion, though much field testing is required before I am willing to say that it is bug free. Presently the output is just over 13K (of PL/9 output code). I figure that will translate into 20K of 6809 'C' code and about 40K of 8086 'C' code.

I have several ideas about speeding up the operation here and there, particularly when going to a line at the other end of the file or searching the entire file for a string match. There are a few places where some Assembler code might speed things up by a factor of two or three. I also have some ideas for additional features to be added. I think, however, the project is due for some settling time as I use the editor and find further bugs (found one this morning and fixed it by adding one line of code.) I've found in general that the hardest part of the debug was not finding the code that caused the problem but determining when the problem occurred, that is, what circumstances caused its occurrence. Just as a matter of interest to some of you, I now have an estimated 300 hours or so in the project. I intend to translate the final result into 'C', not so much for an exercise but to make it more portable.

### Responses

The responses continue to come in with respect to my recent plea for some feedback. I received one more rather nice "nasty" letter last week, and in the same mail one that was most supportive of what I have been doing. The writer of the second letter said that he hoped I wouldn't start filling the column with long explanations of how to use standard FLEX routines since he agreed with me that they are all spelled out in the Programmer's Guide.** I appreciate the support, but I also see that many of you (I would guess the number of responses has reached about 25 at this point), do want some articles about FLEX. One writer suggested that I go back through

the history of FLEX and document some of the early problems with it. (I'm sorry, but I can't imagine that anyone would care about or benefit from such a historical review).

To summarize the responses, I'd say about 15 of the 25 were neutral, five or so were very supportive, five were critical and one was downright nasty. I guess I can't win them all. Along that line I note that some people tend to read things into what I say. Support for the discussions of the basics of FLEX was just about unanimous. Several people picked up on my suggestion of including discussion of proper use of subroutines and sub procedures. While I certainly don't consider my programs to be "textbook" examples of that, I think I do a better job than many, and perhaps the old saying that "those who can't can teach someone else" might hold true in this case. I do sincerely thank all of you who responded positively or negatively (even "nastily").

I'd like to discuss the term "hacker" just once more, and then I'll drop my lone crusade. One of the writers said that I surely must be a hacker at least to some degree. That got me to thinking about the term a little more. If he meant that I get in and tinker with the hardware and software of the operating system, he is absolutely right. I think the reason I don't consider myself a hacker is that I don't enjoy such activity but do tackle it when necessary. For example, I found a clock calendar board in my mail a year or so ago, sent by a friend who had made it out of an old MP-C board. A clock board is not very useful unless you link it to FLEX so that FLEX doesn't ask for the DATE but reads it from the calendar board. It is also nice to have a utility to tell you the current time and date. I got into FLEX and spent a long evening disassembling and chasing subroutine jumps to subroutine jumps to subroutine jumps until I found the code that points to the date prompt and puts it on the screen. I was able to patch FLEX to jump to my routine to read the date and print the date and time on the terminal. If the Clock board is absent, it asks for the date normally. Nice end result, but an assured pain to carry out! A true hacker would relish the challenge. I don't think I qualify.

That, I think is what separates a true hacker from the rest. One admitted hacker told me in a letter that he would hope that he would never sink to doing anything practical with his computer ever! Some of us hack, but we don't "love it". I guess I am in that category. Enough on that subject except to say that I have noted lately that most hackers are very helpful to other "computerists" who are not so advanced in their capabilities. I know of several who have helped people on a continuing basis to make improvements in their systems. Kent Meyers, Joe Aulicino and Bob Jones all deserve special mention for their helpfulness to other SS-50 bus users!

### FLEX

In keeping with my recent series on FLEX, I thought I would mention one of the things FLEX users have perceived as a shortcoming for some time. (This is the topic I mentioned last month, saying I would get into it in a future column.) FLEX provides a sector on each disk called the System Information Record. Among other things this sector contains information regarding how many tracks the disk contains, how many sectors on each track, and most importantly the first and last sector in the "Free Chain", the sectors that have either not been used since the disk was formatted, or those that have been returned to the free chain by deleting the file that they contained. When you open a file on a disk this information is read to determine where the file should be written to the disk. Now, suppose you change disks in that drive. While files are open, FLEX does NOT again read the System Information Record (SIR). When you close the file or continue to write to it, FLEX will write the sector that it had intended to write on the original disk even though that sector might be right smack in the middle of a valid file on the new disk. Of course this causes all sorts of nasty things to happen, among which are the ruining of files, trashing of directories, fouling up of the free chain etc. The rule therefore is that you should not ever change disks in a drive while files are open on that drive.

To every rule there is an exception. This problem first became a concern when Peter Stark tried to put his spelling checker's dictionary on multiple disks. He found that he couldn't change disks and have things operate as planned. Of course, Peter tracked down the problem and figured out a way to force FLEX to read the SIR of the new disk when it was inserted. Therefore with Pete's Spell 'N Fix software you can change disks with files open. If you are deeply interested, a back issue of '68' has the article by Peter on how he did it. The point of this, however, is to be careful about changing disks while running such things as database programs. A favorite trick of mine is to be editing a file such as these notes or a letter to a friend, and then unthinkingly decide to do something else and absent mindedly switch disks, forgetting to exit the edit session. Then after switching disks, I see that I am still editing and exit the edit, thereby ruining the disk I just put in the drive. (Of course I don't mean that the disk is physically ruined, just the data files on it).

### Sorts

I just thought I'd put in a little note here. Two or three responses have come to my little discussion of sorting a few months back. Dan Farnsworth sent me some sort programs (in Assembler, of course), a second reply indicated that it would be nice to have a discussion of the full Shell Metzner sort some time, and a third indicated that the inclusion of that simple discussion had more or less made it worthwhile to have subscribed to '68'. The writer indicated that he had been using a bubble sort "in my Fortran programs" for several years, aware that there were others around but not aware that they were significantly better. Now that my big EDIT project is nearing completion, I'll get a BASIC and a Pascal version of the full sort ready and include it along with a further discussion here in the near future.

### Codes

It occurs to me that some of you newcomers to computing might have been confused by my discussion above referring to ASCII 1 as opposed to Hexadecimal 1. By the ASCII code, I mean the code that, when sent to a printer or terminal that accepts the standard ASCII code, causes the character '1' to be printed or displayed. ASCII stands for American Standard Code for Information Interchange. While it is not my purpose to get into representations of numbers in different number bases, (nearly every programming book ever written contains a chapter or two on this subject), let me just say that the following are all equivalent:

| | |
|---|---|
| 01000001 | Binary |
| 101 | Octal |
| 65 | Decimal |
| 41 | Hexadecimal |

These are all different ways of representing the same binary sequence and they represent the ASCII code for the letter 'A' (capital, that is). The Binary representation shows each Binary digiT (BIT). The Octal code groups the binary digits in threes from the right to the left, so the value is 01 000 001 which you will recognize as the binary representations of 1 0 and 1 respectively. Hexadecimal groups the binary digits in fours as in 0100 0001 which are the representations in binary of 4 and 1 respectively, Hexadecimal codes can represent 16 values with each digit. Since there are only ten decimal digits, the excess six values are represented by the letters A through F-- A representing the value 10 through F representing the value 15.

In the "golden days" of Minicomputers (most notably DEC PDP 8 and 11) Octal codes were the general rule. Since the coming of 8 bit microcomputers, with their 8 and 16 bit values, Hexadecimal has become the rule.

### An Apology

Along with the responses to my request for feedback a couple of months ago, have come several letters asking specific questions... looking for advice and help with computer related problems. Time has not permitted me to respond to all of these requests. Some of them were of sufficient general interest that I will include some discussion of them in this column (if I have not already done so). One reader wrote to ask how to convert his SWTPc system to run OS-9. Since I have never had OS-9 running on my system I could be of little help. Another computer user (apparently in a University) asked if I knew of any way to implement an up/down counter for the SS-30 or SS-50 bus to be used to count logic level pulses from an instrument. Unfortunately that was all the information given. I wrote back and asked a dozen questions. What is the maximum pulse rate from each source? Is there to be one count input and an up/down logic input or are separate up and down count inputs needed? What is the maximum count to be stored in the counter? Does the computer have to do something else while the counter is operating? Depending on the answers, I might recommend anything from a large board full of TTL reversible counters with a PIA or two to read the count, to a simple use of two handshake inputs on a PIA already installed in the system, and a software counter.

If you are going to ask me for help, please give me all the information you can. We engineers are always asking how much, how fast, what else, when, and what if!!

At any rate, I am sorry that I couldn't answer all the letters personally.

** Only those purchasing FLEX from TSC normally received the 'Programmers Guide'. My estimate is that about 80% or better is SWTPC FLEX, which is different than the TSC version and also did not come with the Programmers Guide. The guide had to be purchased separately. My guess is that less than 10% of the users bought it.

DMW

- - -

# OS-9 User Notes

Peter Dibble
19 Fountain Street
Rochester, NY 14620

I'm typing this column in the Lake District of England. The surroundings are special, so is the computer I'm using to type and edit this.

I bought a NEC 8401 to use for just this kind of thing. It is about the size and weight of a notebook, with an 80 by 16 LCD screen. It uses a CMOS Z80 with 64K of RAM and 96K of ROM. (GOSH! PETER, UGH!! - DMW) The ROM holds CPM, some CPM utilities, Wordstar-to-go, Calc-to-go, a trivial database system called filer, and a terminal emulation program called Telcom. I've given Wordstar and Telcom a pretty good workout without finding any problems.

For those who are disappointed that I slipped away from pure OS-9 this month, apologies. This computer (NEC calls it a Starlet) is so nice I thought you'd like to hear about it. I looked for a portable computer that ran OS-9. The joy of learning new operating systems and editors every time I turn around has begun to fade. Nobody knew of one. Too bad. At least Wordstar doesn't turn out to be very different from Dynastar.

### Atari

For a long time now I've been reading about the new 68000-based Atari. It has been delayed and delayed, but I've finally read a letter from a regular person who bought one in a regular store (in Canada). If there are a few of them in Canada now it seems reasonable to hope that there may be a few in the United States by the time you read this.

thought it might be one of the new machines running OS-9 that Ken Kaplan told me to expect this year. I called him up all excited, but he told me that the Atari would use Digital Research's GEM environment. The plan was to put GEM in ROM; this would make it very difficult to patch OS-9 in later. Too bad. It sounds like the Atari ST just might be the Mac "for the rest of them." It would have been nice to have OS-9 in on that.

### The AT&T Unix Machine

AT&T is selling a 68010-based desktop computer for which I believe there is a good chance we'll see a version of OS-9. It sounds like the AT&T Unix machine is fast when running simple Unix benchmarks, but things aren't so good when some of its special features are considered. It supports windows, which are evidently disturbingly slow, and has a front end for many system functions which is said to be friendly but slow.

Evidently AT&T has moved away from their open philosophy which has been such a good thing for Unix. The details of the hardware are only available to OEMs and VARs. You have to be a VERY big customer of AT&T to qualify in either of these categories. The letter I have from AT&T says that they will have a source package available soon, It doesn't say how complete that package will be or what it will cost.

I haven't seen either the Atari or the AT&T yet. I've read about them and heard about them from friends who have used them. I'll keep you posted as I learn more ... Especially if I hear about OS-9 for either computer.

### Small 68Ks

Richard Don at Gimix takes great pleasure in teasing me. He'll call me up (the man's telephone bill must be close to the national debt) and tell me about the latest projects at Gimix -- after making me promise not even to talk to myself about them. Then I walk around bursting to tell people about the 68020-based S50

*Meet people making it happen in OS-9. The movers and shakers who are helping OS-9 become the fastest growing operating system for the 6809 & 68000 in the world.*
*Lively and informative round-table discussions will cover the design and use of Microware Software. We'll also discuss OS-9's dynamic growth from where we are today to where we may be in the future.*
*The exhibit area will feature booths from many of the leading suppliers of OS-9 compatible hardware and software. It's a great opportunity to increase your skill and knowledge in the latest microcomputer software technology. Plan to attend — Register Today!*

***Seminar only $150 Hotel Package* $350***
***Location Marriott Hotel, Des Moines, IA***
***Don't Miss It — Pre-Register Now!***
***Call 515-224-1929 or Write***
***MICROWARE SYSTEMS CORPORATION***
***1866 N.W. 114th St. • Des Moines, IA 50322***

microware®

**Hotel package includes 3 nights, single occupancy at the Marriott Hotel and registration fee.*
*OS-9 and BASIC09 are trademarks of Microware and Motorola*


computer or whatever. Just before I left for England he called and gradually revealed that Gimix was in the final stages of development on a 68020 single-board computer small enough to screw onto a 5 inch disk drive. Of course I couldn't talk about it. When I came back from England I found a letter from Richard. It says "Now this is official." I imagine a copy of the Preliminary specs will by in the Bit Bucket of this issue.

Richard thinks of the Gimix 68020 SBC (Single Board Computer) as a complement for their large 68020 system. Having thought about it while watching Swaledale sheep graze I believe it will be the perfect complement for the **Hazelwood UniQuad. Both boards are meant to attach to 5 inch floppy drives. Both run OS-9 68K. The preliminary information I have from Gimix doesn't include a price, but if it costs between two and four thousand dollars for the board a complete system will fall just one solid jump over the UniQuad (at $1000 with lots of software).

I picture the Hazelwood UniQuad as the starting point for 68K computing. The Gimix Micro-20 (that's what they call it) is the upgrade path. It should run any software purchased or developed for the UniQuad. If I'm reading the specs right, even the peripherals will be compatible**.

With a little care the compatibility can work both ways. A 68020 is a lot of computer, too much for many applications. The 68008 is plenty for most applications and less expensive. There's a small problem: the 68008 isn't fully compatible with the 68020; Motorola enhanced the instruction set of the 68020. However, if you don't use the enhancements, the 68020 remains a fine microprocessor and your software will run on any 68K family microprocessor.

Object code compatibility between the 68020 and other 68000 family computers is likely to become quite an issue for us. Distributing two sets of software, one for the the 68020 and the other for all other 68000s isn't going to make software vendors happy. I expect that most people will just sell software for the 68000 and let those with 68020s do without some small performance gains. Still, in some cases the differences might be important. Here's my suggestion: For application programmers: write for the 68000 and use system libraries as much as possible. Only use 68020 enhancements if they make an important difference. For system programmers: use every feature you can find. Spend plenty of effort creating libraries that are tailored to each system.

I think OS-9's standard function library modules will smooth everything over. They are separate modules that are part of the operating system. The services of the module are accessible through trap instructions to any program. For example OS-9 68K includes two math library modules. Programs that need to do any complicated math invoke library functions from the math library. The library module isn't linked to the calling program; it's located dynamically at run time. If you have a system with a floating point coprocessor, replacing the math library modules with new modules that make use of the coprocessor will switch all your programs over to the faster math without any other modification.

In the case of 68000 versus 68020 the difference is less clear, but the way of hiding the incompatibilities is the same.

### Processes

Processes are among the topics that I particularly like. You've probably noticed that I keep coming back to them. This month I'd like to talk about the way that

processes are realized in OS-9. Process descriptors, process queues, the proc command, and assorted other bits of OS-9 esoterica will come into this discussion.

Each process in OS-9 is represented by a process descriptor. Each process descriptor describes the entire "state" of a process. Two parts of the state are high level; process descriptors are collected in queues (lists) of active, sleeping, and waiting processes, and there is a byte in the process descriptor where OS-9 saves details about the treatment of the process.

The state byte expresses seven attributes of a process. A running process can be "Timed out" which means that it has run out of its time slice. It can be in need of a DAT update. It can be "condemned" (to die), or dead (but not yet gone). A process can be in OS-9 system state (processes that aren't running usually are). It can be in a timed sleep (or in an untimed sleep), or it can be suspended.

Active processes are just waiting for a turn at the processor. OS-9 worries about their priorities and gives each one a fair share of the processor cycles.

For sleeping processes OS-9 watches the clock so it can change them into active processes after the right number of clock ticks.

Waiting processes stay dormant until something specifically wakes them up. Waiting processes are waiting for a child process to terminate; when that happens the waiting process is made active. A signal can also be used to activate a waiting process.

Sometimes the signal that starts a process going is well hidden. Read and write service requests will often leave a process in an untimed sleep then start them again without any obvious signals. The signal in this case is hidden. The interrupt-service part of the device driver sends a wakeup signal to the sleeping process.

Starting a sleeping (or waiting) process doesn't take very long, but it is more complicated than starting an active process. A recent addition to OS-9 is the suspended state.

Suspended processes are stuck in limbo between active and sleeping. They are stored in the active process queue, but the suspended attribute prevents them from being run. They can be activated by simply changing the process status byte. The suspend state is only used by processes that are waiting for I/O, but it isn't limited to that use by any restriction on the suspend state. It was invented as a way to speed up I/O and hasn't found its way into other uses yet. It may not even been incorporated into all device drivers yet.

The three major states (active, sleep and wait) are reflected in the "process queue" that the process is kept in. There are pointers to the Active Process queue, the Waiting Process queue, and the Sleeping Process queue in the system direct page. They are called D.AProcQ, D.WProcQ, and D.SProcQ respectively.

Three queues are used for efficiency. At each clock tick OS-9 runs through the Sleeping Process queue updating counters and moving processes that have slept long enough to the active process queue. It doesn't spend time looking at active and waiting processes. When OS-9 wants to start a new process it just picks the first process on the Active Process queue. Activating a process requires OS-9 to put it into the Active Process queue at a position corresponding to its priority. When a process terminates, OS-9 must search the Waiting Process queue for other processes waiting for it. In each case OS-9 has restricted the number of processes under consideration by keeping processes with different states in different queues.

The Proc Command

I have never been very pleased with the proc command. An easy problem to fix is that the process state is not given in an understandible form. I can never remember what the bits mean. Another complaint isn't easy to deal with. I use the proc command to find out the process number of run away processes so I can kill them. A looping program doesn't usually shut down my system. Time slicing gives other programs time to run, but for some reason proc doesn't work well at all when a program is looping.

My theory is that it requests each process descriptor from 2 through 255 with a separate service request. Most of those process descriptors aren't there (I seldom run more than 20 processes at a time) so the proc command prints a few lines then pauses while it checks for about 200 process descriptors that aren't there. I don't know that that's what proc does, but it is how it behaves.

It's not very important, but relationships between processes could be made easier to see than they are.

Let's see about fixing proc. It is easy to give the process state in English. Arranging them so the relationships are visible isn't very hard either. Speeding the command up is hard. Every way to find out what process numbers are in use is hidden in the system address space. Even in level one the information is in system control blocks that are liable to be changed in any revision of OS-9.

The easy way to find out what process ids are in use is to get a copy of the process descriptor block table. We can get that from a user program by copying it from the system address space. For real elegance we can get it with a special SVC.

The pointer to the process descriptor block table is D.PrcDBT in the system direct page. It points to a 256-byte table where each byte corresponds to a process id. If the byte is nonzero the process id is in use.

The program, ps, included with this column reaches into the system address space to get the process descriptor block table and uses it to determine which process descriptors to request from OS-9. I've written the program in C and used some fancy formatting on the output. The result isn't as pretty as I would like, but I find it useful. You can use ps just as you use proc.

You can also add to it. I've never used the stack pointer value in the output from proc so I didn't include it in ps. You can add it. The DAT image would be interesting (but big). Chasing down the file names attached to the paths in the process descriptor would take you all over the system address space (and get you in deep trouble at the next update), but would be very useful.

There are two command line options. One is the traditional "a" which makes ps display all processes. It will normally only display information about processes owned by the user running it. The other command line option is "a," for short; This makes ps display much less information about each process. The command line format is pretty loose.

```
OS9: ps a a
```
will do the same thing as:
```
OS9: ps as
```

If your system uses a DAT size other than 4096 you may have to make some changes, but give it a try the way it stands first.

- - -

** Editor's Note: It should be noted that both of these systems are being mentioned, without our actually knowing if they perform as advertised (or claimed by manufacturer) or if they are actually supplied as advertised, included all the software, cables, documentation, etc. as promised. We have had some complaint, along these line, about one of these systems. However, since it is not advertised in 68 Micro Journal, we have not seen one, either for normal product evaluation or for advertising approval. I would caution you to investigate before spending your money, when considering any new purchase.

Also speed is an important factor to be considered. In the case of the 68008, it is in many respects slower than your 6809 (and even some 6800) systems. As for the 68020 *GIMIX system, it is probably 30 times faster then the 68008. Of course, this would depend on how much of the "enhanced" 68020 code you use. As time passes I feel that you will see a lot more 68020 code, and a lot less 68008 code. Sorte like 6800 - 6809. Some experts seem to feel that the 68008 is on its way out, and will be the "forgotten" 68XXX.

As to the price difference; it is a proven industry fact that the price of chips goes down with production. The difference now in the price difference of a 68008 and a 68020 system is the CPU cost. The price of the 68020 is bound to fall as has all other devices. I can remember when the 6809 was nearly $100.00. Now you can buy all you want at $4.95. Also the price difference between the different systems could well be that one may use dynamic (cheap) RAM and the other static (more expensive) RAM. And that one has practically four (4) times more RAM available than the other.

*We have seen the GIMIX 68020 run in the mainframe system at NCC, July 1985. It is FAST! We will have an evaluation and test GIMIX 68020 Single Board Computer, for review as soon as they become available. Also it is felt that a full case, power supply, disk drives (including wincheater) will also be available from a second source to make the SBC a very powerful (and inexpensive) desk top unit. The price will be far less than most think. But most important; WE HAVE SEEN THE GIMIX SYSTEM RUN - IT IS FOR REAL.

```
1  #include <stdio.h>
2  #include <os9.h>
3  #include <module.h>
4
5  #define FALSE 0
6  #define TRUE 1
7  #define D_PRCDBT 72 /* address of process descriptor block table */
8  #define DATBSIZE 4096 /* Size of a DAT block */
9
10 static direct char all=FALSE, shortlines=FALSE; /* Options */
11 static direct int MyID;
12
13 main(argc, argv)
14     int argc;
15     char **argv;
16     {
17         char *option;
18         int pid;
19
20 /*----------------------------------*
21  *  Evaluate command line options   *
22  *----------------------------------*/
23         option = **argv;
24         while(argc > 1) {
25             switch(*option) {
26                 case '-': /* all processes */
27                 case 'A':
28                     ++all;
29                     break;
30                 case 's': /* short output */
31                 case 'S':
32                     ++shortlines;
33                     break;
34                 case '\0':
35                     option = *argv++;
36                     --argc;
37                     continue;
38                 default:
39                     break;
40             }
41             ++option;
42         }
43
44 /*----------------------------------*
45  *  Get Process descriptor block    *
46  *  table from the system address   *
47  *  space.                          *
48  *----------------------------------*/
49         GetProcTbl();
50 /*----------------------------------*
51  *  Print process information       *
52  *----------------------------------*/
53         if(!all) MyID = getuid();
54         for(pid=2; pid < 256; ++pid)
55             PrintProc(pid, 0);
56
57         exit(0);
58     }
59
60 static char ProcTable[256];
61
62 /*----------------------------------*
63  *  Get a copy of the process descriptor*
64  *  block table from the system address *
65  *  space.  Put the copy in ProcTable   *
66  *----------------------------------*/
67 GetProcTbl()
68     {
69         int DATImg[2]; /* could get away with 1 */
70         char SysDP[(D_PRCDBT+2)];
71         char **pp;
72         char *PTPtr;
73
74         DATImg[1] = 1;
75         DATImg[0] = 0;
76
77         CopyMem(SysDP, NULL, DATImg, D_PRCDBT+2,
78                 "Copy of System DP failed");
79
80         pp = (char **)(SysDP + D_PRCDBT);
81         PTPtr = *pp;  /* PTPtr contains a pointer to the PDB Table */
82
83         /* The Proc Desc Block table should come within the
84            first two DAT blocks in the system address space */
85
86         if((int)PTPtr > (DATBSIZE * 2)) {
87             fprintf(stderr, "Descriptor table out of range\n");
88             exit(1);
89         }
90         CopyMem(ProcTable, PTPtr,
91             DATImg, (sizeof ProcTable),
92             "Copy of ProcTable failed");
93         return;
94     }
95
96 /*----------------------------------*
97  *  Invoke the OS9 F$CpyMem service *
98  *  request.  If there is an error  *
99  *  Give the error message in msg   *
100 *  and exit with the error number  *
101 *  returned from F$CpyMem.         *
102 *----------------------------------*/
103 CopyMem(to, from, DATImg, size, msg)
104     char *to, *from;
105     int DATImg[], size;
106     char *msg;
107     {
108         struct registers reg;
109
110         reg.rg_a = (int)DATImg >> 8;
111         reg.rg_b = (int)DATImg & '\xFF';
112         reg.rg_x = from; /* offset */
113         reg.rg_y = size ; /* length */
114         reg.rg_u = to;
115         if(_os9(F_CPYMEM, &reg) == -1) {
116             fprintf(stderr, "%s\n", msg);
117             exit(reg.rg_b);
118         }
119         return;
120     }
121
```

```
122 /*-----------------------------*
123  *  Return TRUE if process pid  *
124  *  exists in ProcTable.        *
125  *-----------------------------*/
126
127 ProcExists(pid)
128     int pid;
129     {
130         return(ProcTable[pid]);
131     }
132
133 /*-----------------------------*
134  *  Mark process pid "not       *
135  *  present" in ProcTable.      *
136  *-----------------------------*/
137 ClearProc(pid)
138     int pid;
139     {
140         ProcTable[pid] = FALSE;
141         return;
142     }
143
144 /*-----------------------------*
145  *  A structure describing the  *
146  *  interesting parts of a      *
147  *  process descriptor.         *
148  *-----------------------------*/
149 typedef struct pdesc {
150     char ID, PID, SID, CID,
151          *SP, Task, PageCnt;
152     int User;
153     char Priority, Age, State;
154     struct pdesc *QNext;
155     char IOQP, IOQN;
156     mod_exec *PModule;
157     char *SWI, *SWI2, *SWI3;
158     char Rec_Signal;
159     char *SigVec, *SigData;
160     char DeadlockID;
161     } PROCDESC, *PROCDPTR;
162 #define PDAT 64
163 #define PDESCSIZE 512
164
165 static direct char FirstLine=TRUE;
166
167 PrintProc(pid, leadingblanks)
168     int pid;
169     int leadingblanks;
170     {
171         struct regsiters reg;
172         char *GetMName(), *ExpndState();
173         register PROCDPTR pptr;
174         char Sibling;
175         static char buffer[PDESCSIZE];
176         static char ctlStr[50];
177
178         if(!ProcExists(pid)) return;
179
180         /*-----------------------*
181          *  Get the process      *
182          *  descriptor for pid   *
183          *-----------------------*/
184         reg.rg_a = pid;
185         reg.rg_x = buffer;
186         if(_os9(F_GPRDSC, &reg) == -1) {
187             fprintf(stderr, "Error in GPrDsc %d\n", pid);
188             exit(reg.rg_b);
189         }
190
191         pptr = buffer;
192
193         if(all || (MyID == pptr->User)) {
194             if(shortlines) { /* Short form output */
195                 if(all) {
196                     sprintf(ctlStr, "%%%ds%s", leadingblanks,
197                             "%-4d%-4d %s\n");
198                     printf(ctlStr, "", pid, ~\xFF~ & pptr->User,
199                         GetMName(buffer + PDAT, pptr->PModule));
200                 } else { /* don't bother to print user ids */
201                     sprintf(ctlStr, "%%%ds%s", leadingblanks,
202                             "%d %s\n");
203                     printf(ctlStr, "", pid,
204                         GetMName(buffer + PDAT, pptr->PModule));
205                 }
206             } else { /* Long form output */
207                 if(FirstLine) { /* print heading */
208                     FirstLine = FALSE;
209                     printf("%s%s\n", "PID User Pty Age Sig",
210                             "     State                Mem  Name");
211                 }
212
213                 sprintf(ctlStr, "%%%ds%s", leadingblanks,
214                         "%-4d%-4d%-4d%-4d%-4d%-20s %6d %s\n");
215                 printf(ctlStr, "", pid, ~\xFF~ & pptr->User,
216                         ~\xFF~ & pptr->Priority, ~\xFF~ & pptr->Age,
217                         ~\xFF~ & pptr->Rec_Signal, ExpndState(pptr->State)
218                         pptr->PageCnt*256,
219                         GetMName(buffer + PDAT, pptr->PModule));
220             }
221
222             ClearProc(pid);
223             Sibling = pptr->SID; /* save Sibling in non static store */

224             if(pptr->CID) /* print info about children */
225                 PrintProc(pptr->CID, leadingblanks + 1);
226             if(Sibling) /* print info about sibling */
227                 PrintProc(Sibling, leadingblanks);
228         }
229
230     }
231
232 #define MBUFSIZE 256
233 /*-----------------------------------*
234  *  The primary module's name can only  *
235  *  be found through the module header.  *
236  *  Get the module header and follow     *
237  *  the pointer to the module name.      *
238  *  Change the name string to C string   *
239  *  format and return a pointer to it.   *
240  *  This function may have difficulties  *
241  *  with module names more than 20       *
242  *  characters long.                     *
243  *-----------------------------------*/
244 char *GetMName(DAT, ModOffset)
245     char *DAT;
246     mod_exec *ModOffset;
247     {
248         static char buffer[MBUFSIZE+1];
249         mod_exec *ModuleH;
250         char *NameOffset;
251         char *ptr;
252
253         CopyMem(buffer, ModOffset, DAT, MBUFSIZE, "Error in get name");
254         ModuleH = (mod_exec *)buffer;
255         NameOffset = ModuleH->m_name;
256         if((unsigned)NameOffset > MBUFSIZE - 20) {
257             CopyMem(buffer, (char *)ModOffset + (unsigned)NameOffset,
258                     DAT, MBUFSIZE,
259                     "Error in get name 2");
260             NameOffset = 0;
261         }
262         NameOffset = buffer + (unsigned)NameOffset;
263         buffer[MBUFSIZE] = ~\0~;
264         for(ptr = NameOffset; *ptr > 0; ++ptr);
265         if(*ptr) {
266             *ptr++ &= ~\x7f~;
267             *ptr = ~\0~;
268         }
269
270         return NameOffset;
271     }
272
273 /*-----------------------------------*
274  *  Generate a string that describes    *
275  *  a process state.                    *
276  *-----------------------------------*/
277 #define S_SYS 128
278 #define S_SLEEP 64
279 #define S_TIMOUT 32
280 #define S_IMGCHG 16
281 #define S_SUSPEND 8
282 #define S_CONDEM 2
283 #define S_DEAD 1
284 char *ExpndState(state)
285     int state;
286     {
287         static char s[41];
288
289         s[0] = ~\0~;
290         if(state == 0)
291             return(s);
292
293         strcpy(s, "<");
294         if(state & S_SYS) strcat(s, "sys ");
295         if(state & S_SLEEP) strcat(s, "slp ");
296         if(state & S_TIMOUT) strcat(s, "timeout ");
297         if(state & S_IMGCHG) strcat(s, "img ");
298         if(state & S_SUSPEND) strcat(s, "suspend ");
299         if(state & S_CONDEM) strcat(s, "condem ");
300         if(state & S_DEAD) strcat(s, "dead ");
301         s[strlen(s) - 1] = ~\0~;
302         strcat(s, ">");
303         return(s);
304     }
```

# "C" User Notes

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

## INTRODUCTION

This chapter is intended to introduce the reader to some of the techniques used by C compilers in translating C programs to assembly language, which may then be used to produce machine code for a particular computer.

The topic of compilation is so broad that only a small part of it may be discussed here. Many books are available for those interested in a better understanding of compilation techniques, in general. The techniques used by a particular compiler may often be determined only by inference from the analysis of outputs generated by particular inputs. Thus, it is often easier to study the theory than the application, in terms of compilers.

## THEORY

Almost all of the C compilers on micro-computers are more or less based upon Ron Cain's Small C. Notable exceptions include Leor Zolman's BDS-C for CP/M systems and a few of the Full C compilers (although most have ultimate roots in Ron Cain's Small C).

Ron Cain's Small C (known here for short as Small C) establishes a model of computation based upon a hypothetical computer with two 16-bit registers, one of which is also partially 8-bit byte-addressable, and an 8-bit byte-addressable memory.

Addresses are 16 bits in length, limiting the computer model to addressing 65536 bytes. The computer has a stack without severe restrictions, but with limited access. The instruction set is very small. Essentially, Small C uses a highly-simplified model of the 8080 for its code generation.

This conceptual modelling allows Small C to generate code which executes properly on the 8080 and Z-80 computers, Integers and pointers are stored and processed as 16-bit signed and unsigned integers, respectively. Characters are stored as signed 8-bit bytes and processed as 16-bit signed integers. Since Small C does not support them, no provision need be made for long integers, short or long floating-point variables, structures, unions, etc., as supported by the Full C compilers.

The extreme simplification makes the Small C compilers much smaller and easier to use than the Full C's and also has the potential of generating better machine code. In reality, the code produced by most Small C compilers for the 6809 would be ranked by knowledgeable assembly language programmers as quite poor. A few offer peephole optimizers to clean up the worst of the bad code, but the remaining code is still generally poor. Many of the code generation and other problems with the 6809 C compilers are caused by attempting to emulate the 8080 emulating the C model machine.

Most of the Small C derivatives for the 6809 follow this computational model, although which set of 6809 registers are used varies by implementation. The major variations concern the use of the primary 16-bit register (the other is secondary, or scratch) and whether char variables are sign-extended or zero-filled when converted to type int.

Most implementations use the "d" register as a primary register, primarily because of its split role as the "a" and "b" 8-bit registers, although at least one uses the "x" register as a primary register. Introl C extends the "d" register into the "u" register for long int and float variables.

Several use the "y" register to point to the global variable table, although at least one uses the "u" register for this purpose and uses the "y" register as a secondary register. Several implementations use the "u" register as a stack frame pointer indicating the location of the return address to a function, but one sets up two stacks, one for function parameters and return addresses, and one for local variables.

Therefore very few generalizations about implementations are possible, especially in the area of register usage.

## COMPARISON

To illustrate the generation of machine code by 6809 C compilers, consider the following simple statement:

```
i = 0;
```

in which the variable "i" is assumed local type "int". An assembler language programmer would probably write code similar to the following to accomplish the actions represented by the C statement above:

```
ldd #$0000
std 2,s
```

assuming that "i" represented a location in the stack with positive offset of two. This requires two instructions, 5 bytes of code, and nine machine cycles to execute. Of course, this code is not unique, and might be improved with a little thought.

Everhart's Small C generates the following code for the same statement:

```
leay 2,s
tfr y,d
psha d
ldd #$0000
std ,s++
```

under the same assumptions about the location of "i". This requires five instructions, 11 bytes of code, and 27 machine cycles to execute.

Dugger's Small C generates the following code for the same statement:

```
leax 2,u
pshu x
ldd #$0000
std ,u++
```

which requires four instructions, 9 bytes of code, and 16 machine cycles to execute. Note the use of the "u" register to point to local variables.

Wordsworth's Small C generates the following code for the same statement:

```
ldd #$0000
std 2,u
```

which requires two instructions, 5 bytes of code, and nine machine cycles to execute. This Small C uses "u" to point to local variables, as does Dugger's.

Dyna-C generates the following code for the same statement:

```
ldd #$0000
std 2,s
```

which is identical in size and speed to the original example and Wordsworth's Small C.

Introl C generates the following code for the same statement:

```
clra
clrb
std 2,s
```

which requires three instructions, 4 bytes of code, and ten machine cycles to execute. Note the trade-off in machine cycles for bytes of code with other examples.

For comparison purposes, UNIX C generates the following code for the same statement in the same context:

```
clr r4
```

which requires one instruction, 2 bytes of code, and one machine cycle to execute. UNIX C optimizes the use of registers, placing variables in registers when it is more efficient to do so, even if the "register" declaration is not specified.

It is evident that every C compiler reviewed has a unique manner of handling even the simplest statement. It should be very clear that the more complex constructs would be handled in even more unique manners.

## FUNCTION CALLS AND DEFINITIONS

Since the C language depends so heavily upon function calls, most C implementations attempt to optimize the code generated to call and to establish functions. However, there are several conflicting goals which must be resolved.

Compatibility is a major goal of many C compilers. The UNIX C compiler places function arguments into the stack in inverse order, although it evaluates them in left-to-right order. On computers with only elementary stack manipulation, such as the 8080, the overhead required to evaluate the parameters and then to reverse their order may be considered excessive by the compiler-writer.

In particular, in Ron Cain's Small C compiler, function arguments are pushed onto the stack as they are encountered between parentheses. For example, the following C statement:

```
function (X,Y,Z());
```

would produce the following 8080 code:

```
LHLD X
PUSH H
LHLD Y
PUSH H
CALL Z
PUSH H
CALL function
POP B
POP B
POP B
```

The compiler cleans up the stack after the call using a simple algorithm to use the least number of bytes.

Almost all Small C-derivative compilers (6809 or not) follow the same scheme for function calls. Also, in Ron Cain's Small C, local variables are allocated stack space, as required, and are assigned the current value of the stack pointer (after the allocation) as their address. For example,

```
int X;
```

would produce

```
PUSH B
```

which merely allocates room on the stack for two bytes. References to the local variable X will be made to the stack pointer +0. No particular initialized value may be assumed for local variables.

If another declaration is made, such as the following:

```
char array[3];
```

the code produced would be the following:

```
DCX SP
PUSH B
```

in which case array[0] would be at SP+0, array[1] would be at SP+1, array[2] would be at SP+2, and X would be at SP+3. Local declarations are allocated only as much stack space as is required, including an odd number of bytes. Function arguments always consist of two bytes each. If an argument is type "char" (8 bits), the most significant byte of the two byte value is a sign-extension of the lower byte.

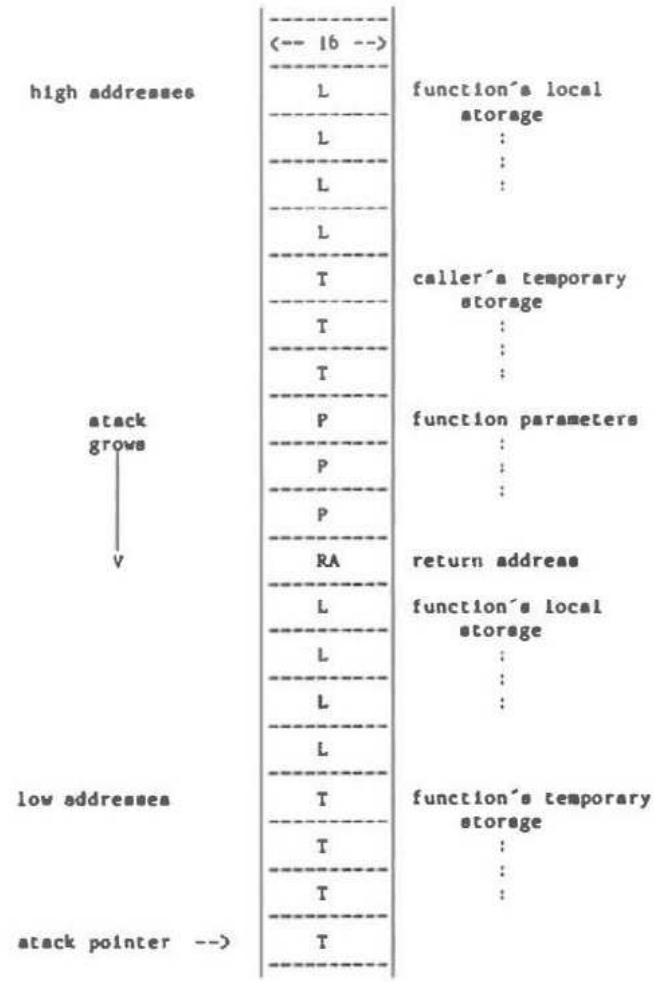
Most Small C-derivatives (6809 or not) follow the same scheme for function establishment.

In most of the 6809 Small C adaptations, function calls are implemented by branch to subroutine instructions. A return from subroutine is generated at the end of each function, as well as for explicit "return" statements.

Before any function call, the parameters (if any) are evaluated from left to right and pushed on the stack. Characters are widened to 16 bits (with sign extension) when they are placed on the stack. After the call, code is generated to remove the parameters from the stack by adjusting the stack pointer. Functions which return values usually do so by placing the value in the "d" register, but implementations vary in their manners of returning values.

At the beginning of each block, code is generated to reserve storage for local variables. To do this, the stack pointer is adjusted downward by the number of bytes necessary for local storage. Locals are referenced by offsets from the stack pointer. At the end of each block, or when breaking out of a block, the stack pointer is adjusted upward to remove the local variables.

Following is the stack frame configuration for a typical function call:



```
                    |----------|
                    |<-- 16 -->|
                    |----------|
high addresses      |    L     |  function's local
                    |----------|      storage
                    |    L     |        :
                    |----------|        :
                    |    L     |        :
                    |----------|
                    |    L     |
                    |----------|
                    |    T     |  caller's temporary
                    |----------|      storage
                    |    T     |        :
                    |----------|        :
                    |    T     |        :
                    |----------|
     stack          |    P     |  function parameters
     grows          |----------|        :
       |            |    P     |        :
       |            |----------|        :
       |            |    P     |
       |            |----------|
       V            |    RA    |  return address
                    |----------|
                    |    L     |  function's local
                    |----------|      storage
                    |    L     |        :
                    |----------|        :
                    |    L     |        :
                    |----------|
                    |    L     |
                    |----------|
low addresses       |    T     |  function's temporary
                    |----------|      storage
                    |    T     |        :
                    |----------|        :
                    |    T     |        :
                    |----------|
stack pointer  -->  |    T     |
                    |----------|
```

As in Ron Cain's Small C implementation, function arguments are pushed onto the stack as they are encountered between parentheses. By definition, parameter passing is "call by value".

For example,

```
func(a,b,c);
```

is compiled into 6809 code similar to the following:

```
ldd a,pcr
pshs d
ldd d,pcr
pshs d
ldd c,pcr
pshs d
lbsr func
leas 6,s
```

Each local variable is allocated as much stack space as it requires, and is then assigned the current value of the stack pointer as its address. For example,

```
int x;
```

would produce

```
leas -2,x
```

which allocates 2 bytes on the stack for the variable "x". The following declaration:

```
char y[5];
```

would produce

```
leas -5,s
```

The address of "y" is the first of the five reserved bytes.

The Small C compiler allocates the space required by each variable declaration. Function calls always allocate two bytes per variable. All "char" variables of 8 bits are sign-extended to 16 bits before being placed into the parameter list in the stack.

The calling function is responsible for establishing the stack in terms of function arguments and return address. The called function is responsible for allocation of stack space for local variables and for ensuring that the stack pointer is correctly restored before the return.

The exact means of pushing values into the stack, calling the function, and correcting the stack pointer are usually very similar to the model above, although there are some variations in implementations among the Small C derivatives on the 6809 and major variations among the Full C compilers. The model just provided describes Everhart's and Wordsworth's schemes almost exactly, and is very similar to Dyna-C's scheme.

In contrast to the function call method just described is the method used by Introl C. When a function is called in this implementation, the second through last parameters are first pushed on the stack in the reverse order of their specification. The first parameter is loaded into the D accumulator. Char values are converted to iot when passed as a parameter. A branch to subroutine is then used to call the desired function. After the function returns, the area in the stack used for parameters is freed.

For example, the following function call:

```
f(a,b,l+2)
```

would generate 6809 code with the following interpretation:

```
push the value of l+2
push the value of variable b
load the value of variable a into d
call function
deallocate 4 bytes from the SP
            (total pushed parameter size)
```

When the function is entered, the stack frame is configured as follows:

```
        stack contents            offset

        the value of (l+2)         SP+4
        the value of b             SP+2
SP -> return address               SP+0
```

If a function requires auto storage locations, it allocates them below the return address of the stack frame described above.

Suppose the function f() has the following declaration:

```
f(x,y,z)
iot x,y,z;
{
   iot b;
   char a;
   :
   :
}
```

The function would expect its parameters to be in the stack frame as described above. The function will often save parameter one (passed in the D register) in the stack just under the return address. After entering the function, the stack pointer would be modified to allow

the storage of a and b below the return address of the stack frame. The new stack frame would have the following configuration:

| | stack contents | offset |
|---|---|---|
| | the value of z | SP+9 |
| | the value of y | SP+7 |
| | return address | SP+5 |
| | the value of x | SP+3 |
| | variable b | SP+1 |
| SP -> | variable a | SP+0 |

Note that char variables use only one byte as auto variables. The only case in which they are allocated two bytes is when they are passed as parameters. The function has the responsibility of "cleaning up" after itself by removing the allocation of variables a and b from the stack. Allocating memory from the stack is accomplished by subtracting the desired number of bytes from the SP and using the area between the new SP and the old SP. Deallocating memory from the stack is the opposite: add the number of bytes to deallocate to the SP.

The Microware/McCosh C compilers use a stack frame format different from any of those described above. It reverses the order of the parameter values, as performed by the Introl C compiler, but it does not place the last parameter in the "d" register.

Suppose the function f() has the declaration provided above.

The function expects its parameters to be in the stack frame as described below. After entering the function, the stack pointer is decremented to allow the storage of a and b below the return address and the U and Y registers. The stack frame would then have the following configuration:

| | stack contents | offset |
|---|---|---|
| | the value of z | SP+13 |
| | the value of y | SP+11 |
| | the value of x | SP+9 |
| | return address | SP+7 |
| | U-register | SP+5 |
| | Y-register | SP+3 |
| | variable b | SP+1 |
| SP -> | variable a | SP+0 |

Unlike the case in Introl and the 6809 Small C's, Microware/McCosh parameters occupy only the stack space necessary for the corresponding types. Thus, if x, y, or z in the example above had been declared type char, they would have occupied only one byte of stack space, and the successive parameters would have been shifted down in the stack.

In all C implementations, the stack pointer must point to the return address in the stack when the function is complete. It must also point to an area large enough to hold all of the local variables of a series of functions at their deepest nesting level, allow room for the parameters and return addresses, leave space for any temporary variables that might be used on the stack, and allow room for saving the system state if the programs are to be run in an interrupt environment.

## OPTIMIZATION OF CODE

Since many Small C compilers on the 6809 are derivatives of the original 8080 Small C, much of their code generation is oriented toward the 8080 model of the C model machine discussed earlier. This provides the potential for two levels of inefficiency in code generation.

The most obvious level is in the almost literal 8080 emulation exhibited by several of the compilers, such as in Everhart's Small C. Recall that five instructions, 11 bytes of code, and 27 machine cycles are required in Everhart's C compiler to perform what other Small C compilers could perform in two instructions, 5 bytes of code, and nine machine cycles.

Because of the limited stack manipulation capabilities of the 8080, allocation of local variable stack space is usually accomplished by one PUSH B instruction for each int variable or array element. Adjustment of the stack pointer after a function has been completed is usually accomplished with the corresponding POP B instructions for each local variable and parameter.

Everhart C (at least partially) maintains this limitation into the 6809 code with the use of one leas -2,s for each PUSH B rather than the use of one equivalent leas -m,s instruction.

Rather than attempting to correct the code generation algorithms, Everhart and Wordsworth Small C systems provide "peephole optimizers" to locate the worst and most frequently-occurring inefficient code sequences and change them into shorter, more-efficient code sequences. These optimizers were covered in an earlier chapter.

Carl Kreider has provided a better peephole optimizer for Everhart's C to clean up additional, longer sequences. Everhart's optimizer removes redundant stack manipulation and modifies two common sequences of code, achieving a ten to thirty percent reduction in code. Kreider's improvements net an additional five to fifteen percent reduction. His optimizer converts the following sequences:

optimizations

| from | to |
|---|---|
| leay p,s<br>tfr y,d<br>pshs d<br>ldd [,s++] | ldd p,s |
| leay p,s<br>tfr y,d<br>pshs d<br>ldd #n<br>std [,s++] | ldd #n<br>std p,s |
| leay p,s<br>tfr y,d<br>pshs d<br>leay q,s<br>tfr y,d<br>pshs d<br>ldd [,s++] | leay p,s<br>pshs y<br>ldd q,s |
| leay p,s<br>tfr y,d<br>pshs d<br>ldb [,s++] | ldb p,s |
| leay p,s<br>tfr y,d<br>pshs d<br>ldb #n<br>stb [,s++] | ldb #n<br>stb p,s |
| leay p,s<br>tfr y,d<br>pshs d<br>lbsr label<br>std [,s++] | lbr label<br>std p,s |

```
leay p,e        ldd p,e
tfr y,d         addd #n
peha d          std p,s
ldd [,s]
addd #n
std [,e++]

leay p,s        ldd p,s
tfr y,d         subd #n
paha d          etd p,s
ldd [,s]
subd #n
std [,s++]
```

A more subtle source of inefficiency concerns the original C model of computation. Since it lacks a condition codes register (called a flags register on the 8080), extra sequences of code are required to capture the results of comparisons and other operations. For example, a 6809 assembler language programmer might use a sequence similar to the following to compare two sixteen-bit numbers for equality:

```
        ldd i
        cmpd j
        bne label
        : (i=j here)
        :
label
        :
```

whereas a 6809 assembler language programmer restricted to the hypothetical C model computer instructions might use a sequence similar to the following:

```
        ldd i
        cmpd j
        bne label1
        ldd #$0001
        bra label2
label1  ldd #$0000
label2  cmpd #$0000
        beq label3
        : (i=j here)
        :
label3
        :
```

Few of the Small C derivatives on the 6809 have totally broken free of the constraints on register usage imposed by the original C model, probably because of the conceptual difficulties of formulating a new model and rewriting the compiler sufficiently well to improve on the original model, restricted as it is.

Beyond the peephole optimizers and general code cleanup are the global code optimizers used by some of the Full C compilers. Rather than emitting target-computer assembler language directly, they emit an intermediate language representation of the program. This representation is usually stated in terms of an operation and two or three pseudo-addresses, often called a "tuple" or "quadruple". From this restructured version of the original C program, many levels of optimization may be performed, including automatic register assignment and reassignment, the removal of constant expressions from loops, the deletion of redundant computations, etc.

The order in which the optimization phases may be performed by a full optimizer is as follows:

1. intermediate language to quadruples
2. basic block analysis
3. flow analysis
4. jump optimization
5. alias analysis
6. basic block optimization
7. data flow analysis
8. loop analysis
9. loop optimization
10. register analysis
11. quadruple to final language

Although no known 6809 C compilers are capable of full optimization, several (including Introl C) perform some optimization phases listed above. UNIX C contains a full optimizer, as does ZILOG C/8000.

Since the code generated by most current 6809 Small C compilers is of generally low quality, the use of compilers with full optimization capabilities is almost certain to increase. As shown by the case of Introl C, global optimization, even on a limited scale, is well worth the time, effort, and cost.

Because of the potentially extremely high demands on memory space and CPU time, optimization should be selectable from none, thru local and partial global, to full global. Properly performed, local (peephole) optimization will always improve the code in terms of instruction, byte, or cycle count. Global optimization relies on algorithms based at least partially on statistical analysis and may not produce better code than that produced from programs written by experienced C programmers. When debugging C programs, all (or all global) optimization should normally be suppressed because it will probably require more system resources to compile the program than to execute it one time, and because the generated code may very well not be recognizable on a statement-by-statement basis.

## CROSS-COMPILATION

Just as cross-assemblers may be used to produce object code for a different type of computer than that used to assemble the program, cross-compilers may be used to produce code for a different type of computer than that used to compile the program.

The organization of several of the C compilers, including Ron Cain's Small C and the UNIX Portable C, greatly facilitates developing such cross-compilers by systematically segregating the language scanning, code emission, and function library functions.

Modular cross C compilers should be reasonably simple to develop, given a properly-structured C compiler base. This would be a boon to the writers of software for single-board computers based upon such processors as the Motorola 6804/6805, Zilog Z8, or Intel 8048/8049/8051 series, thus freeing them from assembler language coding for non-critical applications. Introl offers several such cross-compilers for various machines, including the 6805.

## SUMMARY

This chapter presented several of the techniques and considerations in the use of C compilers in translating C programs to an assembler or object language. This target language may be for the same or different class of computer. The code produced may be further refined with optimization operations.

**SUPPORT YOUR ADVERTISERS**

# ADA[R] And The 68000

by
Theodore F. Elbert
The University of West Florida
Pensacola, Florida 32514

PART 6 - ADA'S SUBPROGRAMS

## ADA'S SUB PROGRAMS

Subprograms are found in virtually every programming language, so much so that the basic machine code instruction set of nearly every computer contains instructions dealing with subprograms. In most languages, however, subprograms are considered useful because they permit an algorithm that is used repeatedly in a program to be coded only once as a subprogram, rather than repeatedly each time it is used. In the Ada language, subprograms also provide a means of applying the software engineering design features of modularity, information hiding, and abstraction of the algorithm implemented by the subprogram. Ada's subprograms comprise two forms that differ principally in the manner in which they communicate with the calling environment. These forms are

procedure--communicates with the calling environment through the use of parameters

function— receives information from the calling environment through the use of parameters, but can return information only as a value of the function name.

In general, subprogram parameters can be of any Ada data type, as can the result returned by a function call. Ada subprograms may also communicate with their calling environment through the use of global variables, but such communication is usually considered to be poor programming practice.

As with the other Ada program units--packages and tasks--subprograms have a two part format consisting of the specification part and the body, with the following functions:

-specification--gives the name of the subprogram, together with the parameters and their associated data types. For a function, the data type of the result is also given.

-body --contains the implementation of the algorithm represented by the subprogram. The body contains the executable code of a subprogram.

The subprogram specification contains all information necessary for use of the subprogram. That is, a programmer developing code for the calling environment needs only the following information:

-the subprogram name
-the names and types of the parameters
-for a function, the type of the result
-the effect of the algorithm

The first three items are contained in the subprogram specification, while the last requires only a description of the algorithm--not a detailed knowledge of the implementation. It is helpful to draw an analogy to the integrated circuit package, for which the pinout arrangement represents the interface with the user, while the physical properties of the chip itself constitutes the implementation. The user of the integrated circuit package must know the interface details--the pin connections--and the effect of the implementation, but he need not be concerned with the detail of the implementation--in this case, the physical properties of the chip. Just as the integrated circuit package is an abstraction of the micro-circuitry on the chip, so also is the specification of a subprogram an abstraction of the implemented algorithm. This same concept applies, and to an even greater extent, to packages and tasks.

Each subprogram parameter, in addition to being of a specific data type, also has an associated mode. The parameter modes are:

-in —information flow is from the calling environment to the subprogram only. Parameters specified as in parameters are treated as constants within the subprogram; their value cannot be changed.

-out --information flow is from the subprogram to the calling environment only. Parameters specified as out parameters must have a value assigned within the subprogram.

-in out--information flow is both from the calling environment to the subprogram, and from the subprogram to the calling environment. Values of in out parameters can both be used and assigned within the subprogram.

While procedures may have any combination of in, out, and in out parameters, functions are restricted to in parameters. Also, the in mode is the default.

The specification of a simple procedure is shown below:

```
procedure SWAP_BITS(BIT_1, BIT_2 : in out BOOLEAN);
```

The two parameters are of the type BOOLEAN, and they are in out parameters. This specification, together with the description of the algorithm, as given below,

"Procedure SWAP_BITS interchanges the values of the two parameters BIT_1 and BIT_2."

suffice to permit a user to avail himself of the services provided by the procedure. No knowledge of the

implementation detail contained in the procedure body is necessary. The separate compilation feature of the Ada language permits compilation of subprogram specifications, so that the specification given above can be compiled into a library unit and can be referenced--for compilation purposes--by other program units.

Calls to subprograms are rather standard; the procedure call is an executable statement and consists of the procedure name followed by the parameter list. A function call, on the other hand, is an expression and therefore must appear on the right hand side of an assignment statement, or in some other context that implies the evaluation of an expression. A call to procedure SWAP_BITS might be

```
SWAP_BITS(A,B);
```

where A and B are the actual parameters to be associated with formal parameters BIT_1 and BIT_2. This particular form uses positional association of the actual and formal parameters; that is, actual parameter A is associated with formal parameter BIT_1 because both occupy the first position in the parameter list. The Ada language also supports named association in subprogram calls. For example, the call to procedure SWAP_BITS could be written as

```
SWAP_BITS(BIT_1 => A, BIT_2 => B);
```

or even as

```
SWAP_BITS(BIT_2 => B, BIT_1 => A);
```

since position is of no consequence when named association is used.

Function calls have similar properties. The specification

```
function ROUND(VALUE : FLOAT) return FLOAT;
```

declares a function with one parameter of mode in. The value returned by the function is of type FLOAT. A call to function round might take the form

```
INDEX := ROUND(NUM);
```

using positional association, or perhaps

```
INDEX := ROUND(VALUE => NUM);
```

using named association. The function call is an expression on the right hand side of the assignment statement.

While a subprogram specification may be separately compiled and then referenced by other program units, the body of the subprogram must eventually be compiled. In the case of procedure SWAP_BITS, the body might be implemented as

```
procedure SWAP_BITS(BIT_1, BIT_2 : in out BOOLEAN) is
    TEMP : BOOLEAN;
begin
    TEMP  := BIT_1;
    BIT_1 := BIT_2;
    BIT_2 := TEMP;
end SWAP_BITS;
```

The procedure body has a declarative part and a sequence of statements. Declarations made in the declarative part of a subprogram body have a scope that extends only the end of the subprogram body. Thus, the variable TEMP declared above cannot be referenced by any other program unit. This feature is in consonance with the concept of the subprogram specification as the interface between the subprogram and any other program unit. That is, the only communication between a subprogram and the calling environment is through the parameter list. As indicated previously, global variables may also provide communication, but their use is normally considered poor programming practice.

All subprograms in an Ada program are recursive; that is a subprogram may call itself any number of times. Subprograms are also reentrant, meaning that more than one concurrently executing task may be executing in any given subprogram simultaneously.

The generic units of the Ada language comprise generic subprograms and generic tasks. The specification of a generic subprogram has an additional part called the generic formal part, and this formal part may contain generic formal parameters. Without going into great detail, an example of a generic procedure can be used to illustrate the concept. The specification and body of generic procedure SWAP are given below:

```
generic
    type ELEM is private;
procedure SWAP(VAL_1, VAL_2 : in out ELEM);

procedure SWAP(VAL_1, VAL_2 : in out ELEM) is
    TEMP : ELEM;
begin
    TEMP  := VAL_1;
    VAL_1 := VAL_2;
    VAL_2 := VAL_1;
end SWAP;
```

The similarity to procedure SWAP_BITS is immediately noted, as is the addition of the generic formal part following the reserved word generic. Both the specification and the body may be separately compiled. When this subprogram is compiled, the precise data type implied by the identifier ELEM is not known. Rather, this information is supplied by a process called instantiation--the creation of an instance of the generic subprogram. An instantiation of procedure SWAP is effected by a single Ada declaration. For example,

```
procedure SWAP_BITS is new SWAP(BOOLEAN);
```

or equivalently

```
procedure SWAP_BITS is new SWAP(ELEM => BOOLEAN);
```

creates (or instantiates) a subprogram fully equivalent to that whose declaration was given earlier. The effect here is that the generic formal parameter ELEM is replaced by the actual parameter BOOLEAN within the instantiated subprogram. Furthermore, the generic subprogram can be instantiated any number of times with virtually any data type supplied as the actual parameter. For example, the instantiation

```
procedure SWAP_FLOAT is new SWAP(ELEM => FLOAT);
```

instantiates a procedure that will exchange values of type float. Nor is the actual parameter limited to scalar data types. For example, if type DATA is some complex data structure--an array of records, for example--then the instantiation

```
procedure SWAP_DATA is new SWAP(ELEM => DATA);
```

creates a procedure that will interchange the values of two objects of type DATA.

Generic formal parameters can be data types, as in the example of generic procedure SWAP, or they can be variables or even subprograms. The generic facilities of the Ada language offer promise for advances in reuseable software. But it is in the generic package, rather than in the generic subprogram, in which the greatest promise lies.

**NEXT: Ada's Packages.**

*Ada is a trademark of the U.S. Department of Defense.

# Basic OS-9

**Ron Voigts**
2024 Baldwin Ct.
Glendale Hts., IL 60137

## POKIN' AROUND PASCAL

This is an exciting month for me. Among other things, I got something I have always wanted. I can now run OS-9 Pascal on the Color Computer. Owning a Pascal compiler has been a secret passion of mine for years. I'm not 100% sure of the reason. Maybe it's because Pascal does not encourage using line numbers in programs. Perhaps its structured program statements like WHILE...DO or REPEAT...UNTIL attracts me. And there is its free-format style and block structures. Whatever the reason, Pascal is here for Coco OS-9.

The Pascal is available from Radio Shack for the CoCo, but it is actually the Microware OS-9 Pascal version 2.0 in CoCo OS-9 format. So whether you use RS's version or Microware's, you'll be able to run the same programs on your OS-9 system. In fact you should be able to take a program from almost any source, compile it, and run it. The only problems that you might run into would be in using a UCSD version of Pascal since OS-9 Pascal is based on the ISO Standard. I picked up a few Pascal books from the library, tried some of the programs in them and they worked.

Pascal is not like other languages such as Basic and Fortran. It must be written in a particular fashion. There are three parts to any program. They are, in the order that they must appear in the program, the

**program heading, the**
**declaration section, and the**
**executable section.**

The program heading consists of the word "program", a name for it, and a list of external data files to be used. The program called "graph" would be started:

```
program graph(input,output);
```

The files declared here are "input" and "output" which are the Standard Input and Standard Output Paths in OS-9.

The "declaration section" is where variables are created. All variables must be declared before using them. In a language like Fortran or BASIC you can create new variables on the fly. Not so with Pascal; add a variable without declaring it and the compiler will let you know about it (in computerese, we would say, in this case, that the compiler will "complain vigorously"). In the variable declaration section you also state the type of variable you are using. Here is an example of how variables might be declared:

```
VAR
   sum: real;
   average: real;
   count: integer;
```

If you don't like the standard variable types, you can create you own with the "TYPE" statement. With it, arrays, complex variables, subranges, and even new variables, can be created. Here I created some new variable types:

```
TYPE
   daysoftheweek=(sun,mon,tue,wed,thur,
                  fri,sat);
   weekdays=mon...fri;
VAR
   day: daysoftheweek;
   workday: weekdays;
```

These are totally new 'variable types' that did not exist prior to creating them. In languages like BASIC there is no way to do this. One other thing, in Pascal you can have constants. They are declared:

```
CONST
   pi=3.14;
   space=' ';
```

When the program is compiled all occurrences of the constant name are replaced with the actual value. 'Constants' are very handy for defining "Computer System Specific" values, because they allow the programmer to simply change that constant and re-compile the program to make it run on a different Computer System. For example, if you are writing a program that will be using the Display for some of its output, you might define a couple of 'Constants' as follows:

```
CONST
   DisplayWidth=32;
   DisplayDepth=16;
```

for the CoCo's normal 32x16 Display, and then change these to "80" and "24" and re-compile the program for a program that would run on a standard 80x24 Display without having to go through the whole program (which may consist of thousands of lines of code) to change every number which pertains to the Width and Depth of the Display.

One other thing is in the "declaration section". It contains the procedures and functions that are used by the main program. Pascal requires that they come before the program (Pascal insists that ANYTHING that is going to be used in the program must be defined before it can be used, and to keep the Compiler simple, it expects these definitions to be in a certain order). You are telling the compiler ahead of time what will be used. Procedures are subprograms that pass parameters in the calling list, like:

```
checksign(number,sign);
```

where "checksign" is some procedure that has been defined earlier. "Sign" might be some indicator to whether "number" is positive or negative. A function is a subprogram whose name contains the returned information. A call to a function might appear:

```
a:=reciprocal(number);
```

This functions takes "number" and returns to "a" its reciprocal.

Editors NOTE: The terms "Procedure" and "Function" are a major source of confusion to anyone reading much of the "Computer Literature" of today. Both refer to a self-contained routine that accomplishes some specific 'job' (to stay away from the word 'function') within a Program, and usually end up as a "subroutine" in assembly language. Generally, the difference between the words "Procedure" and "Function" follow the Pascal terminology in that a "Procedure" performs a function within the Program that does not return some value to the portion of the program that called that procedure. For example, a "Procedure" might Clear the Display Screen; it clears the screen, but does not return something to the statement that called it. On the other hand, a "Function" accomplishes some job and returns the result back to the calling statement in the program. Again, for example, a "Function" might find the Minimum Value of a list of Numbers, and return this number back to the statement that called it. One of the primary exceptions to this interpretation is the "C" Programming Language, where everything is a "function"; so it goes in this industry where the precision of an expression determines whether a Program will run correctly or not! If you are not totally confused by now, you either do not understand the problem or have fought it long enough to have developed your own interpretation of what the two terms SHOULD mean... — rln —

Finally, there is the "executable section". This is the main program. It starts with a "BEGIN" and ends with a "END." (note the PERIOD after the word END; this is a special END in Pascal which means "there ain't no more to this Program"). Everything between these two keywords is executed, except comments which are enclosed by "{ }" or "(* *)". The body of the program is made up of functions, procedures, and Pascal structures. There are 5 types of structures.

The first is:

```
WHILE boolean expression DO
   statement;
```

It executes some statement as long as the boolean expression is true. A simple counter to go from 1 to 10 could be written as:

```
count:=1;
WHILE count<=10 DO
   count:=count+1;
```

Here, "count" is set to "1" and then incremented until it reaches "10". The ":=" is the assignment statement which sets the left term equal to what is on the right. (Again note the precision in terminology of Pascal -- the ":=" indicates that the variable on the left becomes the value of the expression on the right. In BASIC, we say that "A=A+1", yet we have been taught from the first grade that "3 is NOT equal to 3+1". In Pascal, ":=" is an assignment, while "=" really means "is equal to".) The while structure may never execute if the boolean expression is initially false.

A structure that works like the previous one is:

```
REPEAT
   statement
UNTIL boolean expression;
```

But it has a different twist. It executes as long as the boolean expression is false and it executes at least one time since the test is at the end. The counter could be rewritten:

```
count:=1;
REPEAT
   count:=count+1
UNTIL count>10;
```

This does the same thing as before. A big difference is that even if "count" were greater than "10" it would still get incremented at least once.

The third is:

```
IF boolean expression THEN
   statement;
```

Another form of it is:

```
IF boolean expression THEN
   statement
ELSE
   alternate statement;
```

In either case, the statement following the THEN is executed when the boolean expression is TRUE. When the ELSE is used, the alternate statement is executed if the expression is FALSE. The counter from before could be incremented as before , until it reaches 10 and then reset to 0 with the following line.

```
IF count<10 THEN
   count:=count+1
ELSE
   count:=0;
```

At each pass "count" is incremented by 1, until it reaches 10. On the next pass it is reset to 0.

The FOR is next. Its syntax is

```
FOR variable:=start- TO stop-expressions DO
   statement;
```

The TO can be replaced with a DOWNTO if you choose to go from a higher expression to a lower. This one will increment "count" on its own, but we can have it print it's value as it goes along.

```
FOR count:=1 TO 10
   writeln(count);
```

Now we can see what count is as it executes.

The final structure is

```
CASE expression OF
   item1:statement;
   item2:statement;
   { etc. }
END;
```

This structure executes a statement based on the what the expression refers to in the list. If the expression refers to "item1", then the statement after it executes. If it is "item2", then its expression executes. We can now increment "count" selectively.

```
CASE i of
   1: count:=count+1;
   2: count:=count+2;
   3: count:=count+3;
END;
```

This example could probably be written as count:=count+i, but it does make the point.

Now you've got some of the Pascal's basics out of the way. Actually there is so much more, but that will come with time, experience and learning. A few minor things you should know. Pascal is a 'free format' language. Each statement is terminated with a semicolon. The following two statements are identical.

```
IF a=b THEN c:=5;
```

or

```
IF
a=b
THEN
c:=5;
```

Neither statement may be how you want to enter it, but Pascal will take them both ways. The other thing is that wherever a statement is needed, a compound statement can be used. A compound statement starts with a "BEGIN" and stops with an "END" (Note - there is NO period after the "END" for a block; "END." is reserved for the end of the Program ONLY). If we wanted to execute a few statements after the THEN in the previous example, we could enter:

```
IF a=b THEN
BEGIN
   c:=5;
   writeln(a, b, c)
END;
```

Starting with the BEGIN and stopping at the END is considered to be one statement. In fact the END is a continuation of the statement "writeln(a, b, c)", so no semicolon is needed between them, although the compiler wouldn't mind if a semicolon happens to appear between the last statement and the "END".

## LEONARDO'S RABBITS

This month I thought it would be fun to use Pascal to solve a math puzzle. The puzzle was first published in the 13th century by a mathematician named Leonardo of Pisa (was EVERYONE named "Leonardo" in that era, or was he the major "publisher" of that time??). He was known by another name, but I'll leave that for later. He asked, "A man put a pair of rabbits in a place enclosed by walls on all sides. How many rabbits will be produced in 1 year if a pair can produce a new pair every month and it takes 2 months before a new pair can produce offspring?" The two months is needed since it takes 1 month for a pair to mature and another to produce young.

Back in those days Leonardo didn't have OS-9, or a computer to help him. He had to think this out. In the first month, a new pair is born to the original pair of rabbits. In month two, the original pair produces another pair, plus the new pair from month 1 matures. In month three, the two mature pairs produce two more pairs, plus the pair from month 2 matures. This goes on for 9 more months! The simpler solution is in this months listing. I solved the problem with Pascal. You have most of the information from this months column to understand it. A few things are worth noting. I used variable names that were descriptive. The OS-9 Pascal compiler only understands the first 8 characters of a variable name, but it is still legitimate to use longer names if necessary. Just make certain that the first 8 characters don't conflict with any other variable names. I've capitalized keywords and put the rest in lowercase.

The words "writeln" and "write" are not keywords, but are special procedures defined by the system, so they are in lowercase. It doesn't matter to Pascal whether we use uppercase or lowercase letters. It understands either equally well. Also, in the line with "writeln", variables have a colon and number after them. This specifies a field in which the value is to be written. "Count:5" means the value of "count" is to be written in a field of 5 characters wide with the number being right justified. Give the program a try.

If you don't have Pascal and are dying to know the answer, here is what would be printed:

| MONTH | MATURE RABBITS | NEW RABBITS |
|---|---|---|
| ******* | ********** | ********** |
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| 2 | 2 | 1 |
| 3 | 3 | 2 |
| 4 | 5 | 3 |
| 5 | 8 | 5 |
| 6 | 13 | 8 |
| 7 | 21 | 13 |
| 8 | 34 | 21 |
| 9 | 55 | 34 |
| 10 | 89 | 55 |
| 11 | 144 | 89 |
| 12 | 233 | 144 |

The number of rabbits in month 12 is 610

Notice that two identical sequences appear under "mature rabbits" and "new rabbits". Our friend, Leonardo, discovered this special number sequence and hence it was named after him. His last name was Fibonacci and the number sequence is called the Fibonacci Series. The sequence starts with two 1's and then every subsequent number is the sum of the previous two numbers. So,

```
1 + 1 = 2
1 + 2 = 3
2 + 3 = 5
3 + 5 = 8
etc.
```

A fun program you might want to try is one that will generate the Fibonacci Series. The method you use would be similar to the "rabbit" program except you only want to print the outcome at the end of each addition. See what you can do.

### XSCREEN, THE EXTRA SCREEN

One problem with running OS-9 on the Color Computer is its display. It is only 32x16 characters in size. That is not much. If you enter a program like this months, you'll find that a line may wrap around a few times, making things hard to read. In addition, the Coco does not have true upper and lower case. So if you keyed in this months program you'd see black letters on a green background and green letters in little black boxes. Not an easy screen to read!

One solution is XSCREEN. It is a high resolution screen that allows you to go 51, 64, or even 85 characters per line. You can choose from a green and black, or a buff and black display. There is also light characters on a dark background, or dark on a light background. All you do is place XSCREEN in you commands directory, /d0/cmds. Whenever you need a high resolution screen just enter XSCREEN and (poof!) it is ready to go. The program is menu driven, so you can select the type screen you want with just a few keystrokes when you start it. Later you can change it to a different configuration by returning to the menu.

If you are interested in XSCREEN, it is available from:

Microtech Consultants Inc.
1906 Jerrold Avenue
St. Paul, MN 55112
(612) 633-6161

The cost is $19.95, plus shipping and handling charge. They also have some other OS-9 products like XWORD, a word processor, and XTERM, a communications program.

That wraps up another month. If you get a chance look into Pascal. It is a fun and rewarding language, and is often used in Computer Literature as a vehicle for describing algorithms, etc., since it breaks large problems up into easy to understand smaller sections.

Learning to program in Pascal will make you a better Programmer in ANY Language, since it forces you to understand both the Problem AND the Solution before you begin writing Code. In future columns I will include some Pascal programs. But there will also be C Language, Basic09, and assembly language programs as well. So keep reading "Basic OS-9". Bye for now!

```
{ This program will solve the math puzzle proposed
  by Leonardo of Pisa in the 13th century.  His
  puzzle was:
  "A man put a pair of rabbits in a place enclosed
  by walls on all sides.  How many rabbits will be
  produced in 1 year if a pair can produce a new
  pair every month and it takes two months before
  a new pair produce offspring?" }



VAR
   maturerabbits      : integer;
   newrabbits         : integer;
   totalrabbits       : integer;
   temporary, count   : integer;

BEGIN
   { These are the initial conditions }
   maturerabbits:=1;        { one pair }
   newrabbits:=0;

   { A header to tabualte the data }
   writeln;
   writeln('              MATURE       NEW');
   writeln('   MONTH      RABBITS     RABBITS');
   writeln('  *******  **********  **********');

   { Calcualte the results  and print it }
   count:=0;
   WHILE count<=12 DO
   BEGIN
      writeln(count:5, maturerabbits:11,
              newrabbits:11);
      temporary:=newrabbits;
      newrabbits:=maturerabbits;
      maturerabbits:=maturerabbits+temporary;
      count:=count+1
   END;

   { Print the final tally }
   writeln;
   totalrabbits:=maturerabbits+newrabbits;
   write(' The number of rabbits in month 12 is ');
   writeln(totalrabbits:6)
END.
```

# 68000 User Notes

Phillip Lucido
2320 Saratoga Drive
Sharpsville, Pa 16150

### Re-revisiting the Time Program

On closer examination, I've discovered another small problem with the listing of the time program from this column in the July '85 issue. It should be fairly obvious to those typing it in, unless you don't know assembly language. Anyway, on page 23, the last line of assembly code, labelled gprdsc2, is a movem.l. The next line, on the next page, is #endasm. There should be a single line between these:

```
gprdsc2  movem.l  (a7)+,d1/a0
         rts
#endasm
```

As I said, this should have been clear to those of you who typed the program in. If you didn't see the missing line (you know what I mean), then this could very well be the reason for your program behaving in a very strange manner.

### Computer Burnout
### or
### What My Summer Vacation Did To Me

The past month or two has been a curious time for me. I got back, in early June, from several weeks spent in Washington, D.C, and St. Louis, carried my Macintosh from the car to the computer room in my house, and proceeded to do -- absolutely nothing! Not some half-hearted imitation of nothing, either. This was the real thing. I didn't plug my Helix back in for a full two and a half weeks, and only then because I was feeling guilty about being hopelessly late with last month's column.

My treatment of the Mac has been even worse. It remained packed in its carrying bag for at least a full month. It was finally plugged back in when I wanted to use MacPaint for some graphing. It wasn't until yesterday, though, a full month and a half after getting back from vacation, that I hooked the printer back up to the Mac. I still haven't hooked up the second disk drive!

OK, just what gives here, Phil? Have I been hit by that dreaded demon, Computer Burnout? Dumb question - of course I have. A better question is why, and when does it end?

I've been working with computers for about nine years now, since high school. In all that time, my attitude toward computers has always been that of the hacker. Not the general media's distorted compu-crook, out to invade the credit bureaus of America, but the true hacker who plays with computers because they are entertaining. They stretch the mind in a special way, better than any crossword puzzle or board game.

Ah, but here's the rub. For the past four years, I've been working full time as a system programmer, writing some dreadfully huge programs in 8080 and Z-80 assembler. It has been getting increasingly hard to come home from sweating out a program at work, and turn on my own computer system to play around doing pretty much the same thing I do at work. All it took was an extended vacation to bring my home computer activity to a complete standstill.

Right about now, you're probably asking yourself, "So why is he telling me this?" Partly, this is a space-filling expedient. If I've been burned out for the past month, I'm unlikely to have much to talk about. But more importantly, this experience got me thinking about our mutual hobby.

When I first became active with micros, they were small, cantankerous little beasts. A 'user-friendly' machine was one that didn't go up in smoke when you turned it on the first time after putting it together. Those systems may have been difficult to use, but they stretched your mind to the limit. Because of my involvement with computers, I learned programming, digital (and a little analog) electronics, hardware construction techniques, and my own peculiar form of touch-typing which would horrify any high school typing teacher.

Now look around you. Of the three computer systems currently in my house, none are kit-based. All have disk drives, with the largest having a 15 megabyte hard disk. Only the smallest, a CoCo, has only 64K of memory. My Mac has 512K of RAM, and my Helix has a full megabyte. Forget those silly comparisons that say "this computer on a desk is as powerful as a roomful of computers from past decades." Those early machines couldn't touch mine for ease of use and single-user power.

Of course, just because the machines have gotten bigger and powerful, that doesn't mean you can't still play around with them. Many of you out there are undoubtedly doing just that, writing software or modifying your systems for your own enjoyment. But I've been at this for so long now, that I just can't seem to get the same enjoyment from writing small programs. I've gotten deep within my machine, doing things like disassembling the operating system to see what makes it tick, writing several disk drivers and other operating system level programs, and writing dozens, maybe hundreds, of utilities. Finally, it seems to have gotten to the point that whenever I start some project, it's either something too much like a previous project to keep my interest, or it's too large a project to tackle, after working all day on equally large projects for work.

I'd be curious in seeing some response from you out there. How long have micros been a hobby for you? If your job also involves working with computers, do you manage to keep the two separated, or does the work begin to intrude on the fun?

Sorry for taking up space with this admittedly rambling discourse. This column necessarily follows my own computer experiences, though, and this is certainly one of the major experiences lately.

Anyway, I have found something of a solution which is likely to invigorate my interest in computers. Since my full-time job writing software seems to be the problem, I'll just quit my job! Actually I'm leaving my present job with an eye toward consultant work and time spent working on computers programs that interest me, in place of working two straight years on one gigantic problem. If relying on my own ingenuity for an income doesn't get me going again, nothing will (impending bankruptcy has a way of doing that).

### The Mail Brings a Reprieve

Thank goodness! Just when I was beginning to despair of filling up this column, I get mail. First, I received a letter from Kirk Anderson, who first wrote about a year ago. Second, I have received fliers with information on new 68000 products from two companies.

Mr. Anderson was one of the earliest purchasers of OS-9/68K.

He starts out by mentioning that he now has Version 1.1, and is finally very pleased with what he has bought. This certainly matches my own experiences. As I've mentioned in previous columns, OSK seems to have settled down to a stable form, with useful programming tools and far fewer bugs than previous versions.

There is a question in the letter. Sored, the OSK screen editor, looks for a terminal configuration file, /dd/sys/termset, which is to hold a description of the escape sequences used for the terminal. The question has to do with the "/dd" part of the file's name. What is it, and why doesn't Sored just look in the current directory?

Actually, the use of a /DD device is an attempt by Microware to solve a thorny problem with the way in which OS-9 refers to separate devices. In a hierarchal (tree-structured) directory system, an operating system needs some way in which to find standard information, such as configuration files. Placing a copy of the files in the current working directory is not a particularly good idea, because the working directory can change so often, and multiple copies of the configuration files in each possible working directory can waste a lot of disk space. Putting the files in the current execution directory is not much better, since that directory is really meant for programs, not for description files.

Unix, with a hierarchal directory, gets around the problem nicely, but in a way that isn't directly applicable to OS-9. In Unix, there is a single root directory named "/", which can be used as a base for finding all other files in the entire system. In OS-9, though, each physical disk device has its own root directory, such as /H0, /D0, or /R0 for a hard disk, floppy disk, or ram disk, respectively.

Some OS-9 software which needs to find a standard file ignores the problem, using a fixed device like /D0 or /H0. If you must place the file on some other device, then you must change the actual program code, or specify the location of the standard file using a command line option. Neither choice is very convenient. This sort of solution is also often seen in assembly language source text using standard definition files.

Finally, there is the /DD device. The DD stands for "Default Disk". This does not refer to any particular physical disk or device. Instead, depending upon your system configuration, /DD is expected to be some device which is always available, and which holds all of the various standard directories and files like /DD/SYS/termset. Generally, /DD is an alias for some disk in the system, such as /H0 or /R0. To set up the alias, a device descriptor for /DD is created which matches the descriptor for the aliased device in all respects except for the actual device name. For instance, on my system, the /DD descriptor is a copy of the /R0 descriptor, so both refer to the 256K RAM disk. If I did not have enough memory for a full sized RAM disk, then I might use a /DD which was an alias for /H0 or /H1 instead.

Now, when OS-9 software needs to find a standard file, it just looks on device /DD. It is not important to the software that this is actually a RAM disk, or hard disk, or whatever. All that is important is that the file is globally available in a known location, no matter what the current directories may be.

One point - if /D0 is a RAM disk, then obviously all of the standard files which may be required must be copied from a true disk when the system is booted. Thus, while OS-9 programs can refer to a general device /DD, the Startup file must refer to a particular device like /H0 or /D0.

To sum up: /DD, the default device, is a convention used by OS-9/68K in order to locate standard files, like configuration files, assembler system definitions, or C language #include files. To use the convention, the standard files in your system should all be located on one particular disk, such as /H0, /D0, or /R0. Create a device descriptor for /DD, which matches the descriptor for the device holding the standard files, in all respects except for the changed name. If your system, as shipped, does not come with a /DD descriptor in the OS9Boot file, then create your own, place it in the default execution directory, and use **load** and **ini8** commands in your startup file to get the descriptor into memory.

There was one more question in Anderson's letter. He noticed that I seemed to have wholeheartedly converted to C (not surprising in view of my impending burnout - I probably hadn't the patience for assembler). He is using assembler for just about everything, mostly because he is interested in what's going on in the guts instead of quickly pumping out code. What he'd like to know, then, is what other people are using. Well, how about it? I've never gotten a good idea of how many of you are out there. Here's your chance. Do you program in C, assembler, Basic09, Pascal, or something else? Do you program for the fun of it, or is the idea to quickly produce a working program?

If anybody else has any new hardware they would like to call attention to, send me a flier. I can't cover hardware in much detail, but I can at least mention it so people know where to go for more info.

### Next Month WILL be Better

So much for a very difficult column. One way or another, I am going to break through this burnout nonsense. I'll be back next month, this time with some useful information of some sort. I have to limit myself to at most one of these silly outbursts per year.

- - -

# Pleasant PL/9

Lane P. Lester
413 Woodland Circle
Lynchburg, VA 24502

As a biology professor, my interest in the use of computers is two-fold. I'm interested in ways that the machines can improve my job satisfaction through the elimination or diminution of boring tasks (management of student grades). I'm also looking for ways to increase my personal productivity (word processing, database management, and research). The second area of interest is that of the hobbyist: these things are fun to run! And as Ron Anderson put it in his June, 1985 column, "Most of the fun of having a computer is programming it to do what you want it to do, not running someone else's software."

Programming involves the use of a language, and if one stays with computers for any length of time, the choice of a language becomes important. Nowadays, most personal computers come with BASIC, so that's the first language that most people learn. Modern BASICs are powerful enough to do just about everything, and with compilers like K8ASIC, lack of speed is no longer a problem for most applications. But just as we are able to find compelling reasons for buying that new piece of stereo equipment, most of us get the itch to learn a new language and are able to provide our spouses, employers, or just ourselves with all kinds of high-sounding advantages for the change.

But what language to get? We 6809ers have a nice selection from which to choose: assembly, Basic09, C, COBOL, Pascal, FORTH, FORTRAN, PL/9, Whimsical, and maybe some others that slip my mind. The June, 1985 issue of BYTE had an article about choosing a programming language, but the advice provided would not be very helpful to the hobbyist who can't afford to change languages with each change in application. It takes time and effort to learn a language well enough to get it to do useful things, not to mention the cost of purchasing the assembler / compiler / interpreter. What the hobbyist needs is a language that will do a lot of things well. On the other hand, I suspect that I'm not the only hobbyist who is interested in the "latest stuff", whether it's hardware or software.

One of the latest languages to appear is Modula-2. It was created by Niklaus Wirth, the author of Pascal, as an improvement and extension of Pascal. I believe there is one or more Modula-2 compilers for the 68000, but it hasn't reached the 6809... yet. When it does, it will be the "latest", but is it a good idea? This series of articles is about PL/9, so I am not going to give a complete discussion of Modula-2, but a few comments might be worthwhile. At least, I'll feel better for having said them.

With BYTE columnist Jerry Pournelle's continuing promotion of Modula-2 and now the special issue of BYTE devoted to that language, there is probably no stopping the avalanche of Mod-mania. However, in reading the articles dealing with this new language I detect a particularly distasteful hidden agenda. There's no disputing the fact that some of the features of Modula-2 are definite improvements over Pascal, primarily those changes that give the programmer more freedom. But I believe that the developers and promoters of this language yearn for the Ante-micro Age when computer scientists were high priests who dealt in secret arts not accessible to mere mortals. How else can we evaluate the reincarnation of octal numbers for ASCII constants and the abbreviation of "<>" to "" and "AND" to "&" while retaining the nuisance of ":="? What need has been demonstrated for the additional vocabulary of "atoms" and "individua"? How much more self-documenting does a program become when we substitute INC(I) for I = I + 1?

Wirth himself appreciates the importance of readability in a program. In the May, 1985 issue of Computer Language, he stated, "What I mean to say is that programs must be understood by humans. Programming should be an act directed not at the machines, but at humans." In that same issue, Donald Knuth, author of The Art of Computer Programming, said, "I'm trying to emphasize that the best way to program, I believe now, is really to concentrate on explaining to a person what the computer is supposed to do rather than explaining to a machine what it's supposed to do."

So which are the readable languages for the 6809? Let me throw caution to the winds and list a few in decreasing order of readability: PL/9 (of course!) > Pascal > C > FORTH > assembly. I've written at least a few programs in all but C, and I've looked at enough C code to be able to place it in the list. As an aside, for transportability I don't think C can be beat, but it's harder to learn, harder to read, and compiles slower than PL/9.

It may seem a contradiction to what I've said above, but I'm glad that I learned assembly as my second language after BASIC (well, FORTRAN was really first, but who cares?). I thought for a while that I was going to have to teach a course in assembly language programming, and that impelled me to work my way through Leventhal's excellent 6809 Assembly Language Programming. Even though I want on to gain an acquaintance with other languages, the familiarity with assembly continues to be a big help, especially with PL/9, since it lets you get as close to the machine as you wish. In the same interview quoted above, Niklaus Wirth said, "For computer science students, I would say it's absolutely essential that they learn about machine structures and assembly coding. Not to become assembly language programmers, of course, but to gain insight." And Donald Knuth expressed a similar idea with, "I think one of the most important things for people to learn is the ability to flash rapidly from the high level to the low level and back."

One of the main things that make PL/9 so pleasant is the freedom it affords the programmer. PL/9 allows you to "flash rapidly from the high level to the low level and back". It has high level words like WHILE, and REPEAT - UNTIL, but it also has low level words like ACCA, which reads from and writes to the A accumulator. There are other kinds of freedom you can enjoy. The co-resident editor and compiler provides freedom from long compile times. The programs produced free you from slow operation and memory - gobbling size. There's freedom in mixing 8-bit and 16-bit numbers. And if you want, you can even call the same memory location by two (or more) different names.

This latter option can really improve the readability of programs. On my CIMIX video board, the same byte serves sometimes as a mode control port and sometimes stores the location of the last line on the screen. So in stuff I write for the board I have

```
AT $E021 BYTE MODE;
AT $E021 BYTE LAST;
```

so that where it makes the most sense I can call the port MODE, and where it refers to the last line I can call it LAST. In vectors, as single - dimension arrays are called, you can even have a name that refers to an embedded part of the vector. An example with which most readers will be familiar is the FLEX File Control Block,

a 320-byte buffer used by FLEX. The second byte of this vector is the error byte, which indicates whether something has gone wrong during a disk operation. You can handle this in two ways, by either placing the FCB at a definite location or by letting PL/9 put it on the stack. The former would look like this:

```
AT $C840 BYTE FCB;
AT $C841 BYTE ERROR;
```

allowing you to refer to FCB for disk operations and to ERROR to see if everything went OK. To put these on the stack, you'd write:

```
GLOBAL BYTE FCB(1), ERROR(319), NEXT_GLOBAL_VARIABLE;
```

That would reserve one byte for FCB and 319 for ERROR. But FCB requires 320 bytes, you say? PL/9 doesn't care what kind of "subscripts" you put on the variables; it assumes you know what you're doing. When FLEX refers to FCB, it'll use the whole 320 bytes without hiccuping over the fact that you only declared one byte for FCB. So you'll be able to use statements like:

```
OPEN_FOR_READ(.FCB);
IF ERROR THEN FLEX;
```

How much more readable can you get?

To close with a little code, I mentioned at the outset that the hobbyist doesn't have the time or money to invest in a lot of languages. He or she needs to get maximum mileage out of the language(s) on hand. The other day I got the urge to be able to control various Epson FX-80 printer modes so that they'd be set for CAT and LIST. I wrote some little programs to do the job, and I gave them the same names as the "dot" commands in my version of Ron Anderson's JUST text processor. LM.CMD, the one that follows, sets the Epson's left margin.

- - -

```
/* LM, Sets Epson FX-80 left margin */
/* Address all of memory as a vector */
AT $0000 BYTE MEM;
/* Pointer to place in command line  */
AT $CC14 INTEGER LINE_BUFFER;
/* Last line terminator encountered */
AT $CC11 BYTE LAST_TERM;

CONSTANT ESC = $1B, CR = $0D;

ORIGIN = $C100; /* FLEX COMMAND AREA */

PROCEDURE PRINTCHAR(BYTE CHAR);
/* Put character in Accumulator A */
ACCA = CHAR;
/* Call FLEX printer routine */
CALL $CCE4;
ENDPROC;

PROCEDURE MAIN: BYTE MARGIN, I;
/* Initialize printer routines */
CALL $CCC0;
/* Get margin value after command */
MARGIN = 0;
WHILE MEM(LINE_BUFFER) <> CR
   .AND MEM(LINE_BUFFER) <> ':
      BEGIN
         MARGIN = MARGIN * 10
            + MEM(LINE_BUFFER) - $30;
         LINE_BUFFER = LINE_BUFFER + 1;
      END;
/* Tell FLEX if there's more on the */
/* command line */
LAST_TERM = MEM(LINE_BUFFER);

/* Get Epson's attention */
PRINTCHAR(ESC);
/* Set margin to value of next byte */
PRINTCHAR('l');
/* This is the byte! */
PRINTCHAR(MARGIN);
```

# CoCo User Notes

by **Carl Mann**
30 Warren Ave.
Amesbury, MA 01913

Midnight. Black clouds skidded across a starless sky. Somewhere in the distance a screech owl answered the howling of a lone dog with a cry that foretold the doom of the world. I rolled over again, muttered something between my teeth about animals and the trouble they can make, and was floating over the brink of slumber's chasm when...

Tap...tap...tap. Never mind. A settling of the house, perhaps, or the refrigerator overflowing its Condenso-Pan again. Goodnight, world. Hello, Void. Lovely to see you again, my friend...

TAP!..TAP!..TAP!!! Hello? That's no refrigerator-that's MY door! The Void vanished into itself as I dialed up a reasonable glow from the bedside dimmer box (my eyeballs send me roses-by-wire at least once a year since I installed the thing) and fumbled for my pajamas. Oh. Wearing them. Okay. Good. The door.

Nobody there! ?Por que? (My CPU sometimes lapses into Spanish when it can't think of the appropriate French phrase. It's a slightly charming feature of the wetware, I think.) I rubbed my eyes with my knuckles. It didn't help. Then I felt it: something cold and not very hard under the tip of my big toe. A package?

A package. I retrieved it and edged the dim light up a few degrees. Judging from the stamps in the corner of the heavy cardboard wrapping, it had come quite a distance. A rubberstamp had been applied to several of the stamps. Blood-red ink on blood-red postage. Dimly I discerned the word, "BELGRAD". What!

I found my Swiss army knife (never leaves my side) and cut the tape and string and tape that bount the parcel together. Inside was a cassette with no label. I sniffed. The case smelled like petroleum. I yawned and went back to bed.

Next morning I did the obvious. I dropped the mysterious tape into CoCo's cassette drive and (on a hunch) typed SKIPF <ENTER>. Sipping on the morning tea, I was rewarded in due course with the filename, "LABLMAKR". I rewound, CLOADed, and LLISTed the result. Interesting stuff!

```
10 CLS:PRINT" THIS PROGRAM MAKES CoCo PRINT"
20 PRINT"text on standard mailing labels.";
30 PRINT"Any number of duplicates may"
40 PRINT"be made by pressing <ENTER> at"
50 PRINT"the end of a run. Tap <SPACE> to";
60 PRINT"advance the label one or more"
70 PRINT"lines as required. Movement to"
80 PRINT"the next label is automatic."
90 FOR X=1 TO 2000:NEXT
100 PRINT:PRINT"            OTHER KEYS:":PRINT"<UP ARW>
ADVANCES TO NEXT LABEL.";
110 PRINT"<DN ARW> =1/2 REVERSE LINEFEED."
120 PRINT"<LF ARW> BACKFEEDS TO PRIOR":PRINT"
LABEL."
130 PRINT:PRINT"     PRESS ANY KEY TO START.";
1000 E$=INKEY$:IF E$=""THEN 1000
1010 CLS:PRINT"    Set BAUD Box for 9600!":FOR X=1 TO
```

```
2000:NEXT
1020 POKE 150,1:POKE 282,255
1030 CLS:PRINT" Vaht you vant me to print for   you today,
aveetie?"
1040 PRINT"furadt line:":LINE INPUT A$
1050 PRINT"eeckondt line:":LINE INPUT B$
1060 PRINT"thurdt line:":LINE INPUT C$
1070 PRINT"fourdth line:":LINE INPUT D$
1080 INPUT"Tab?";A
1090 PRINT#-2,TAB(A);A$
1100 PRINT#-2,TAB(A);B$
1110 PRINT#-2,TAB(A);C$
1120 PRINT#-2,TAB(A);D$
1130 PRINT#-2:PRINT#-2
1140 CLS:PRINT"You vant maybe anuzza vun, mein komrade?"
1150 E$=INKEY$:IFE$=""THEN 1150
1160 IF E$<>"Y"AND E$<>"y" THEN 1170:ELSE RUN1020
1170 IF E$=CHR$(13) THEN1090
1180 IF E$=CHR$(32) THEN PRINT#-2:GOTO1150
1190 IF E$=CHR$(94) THEN FOR L=1 TO 6:PRINT#-
2:NEXT:GOTO1150
1200 IF E$=CHR$(10) THEN PRINT#-
2,CHR$(27)+CHR$(68);:GOTO1150
1210 IF E$=CHR$(8) THEN FOR L=1 TO 6:PRINT#-
2,CHR$(27)+CHR$(10);:NEXT:GOTO1150
1220 IF E$<>"N" AND E$<>"n" THEN1150
1230 PRINT"No.":PRINT"Bye for now, aveetheart!!":END
```

I wondered whose uncle the old bat was who (which? what?) had dropped all this at my door. Oh, well. Obviously long gone by now. Besides, who am I to look a workhoree in the mouth? I loaded my old Qume printer up with labels, then fired up the program. Was I pleased? Ask my fellow worker John (who ALMOST turned up his nose at CoCo, thinking he'd had the VERY best in computers at Harvard Univerelty... now he runs CoCo at work, too!) what he saw me doing that morning. Now we both are the proud custodians of LOTS of neatly labelled file folders. No more chickenscratch labels in OUR system!

Let the reader beware: this program was targeted onto my personal system. (Who done it? who knows? Hope I don't have to Pay Later...) Other printers may not have fancy reverse and/or half-reverse linefeeds. If yours is of this sort, you should delete program lines 1200, 1210, and the references to these functions in the opening remarks. On the other hand, your printer may in fact support these commands- but through different codes. If this is the case, your printer manual will tell you so. Substitute codes as required.

Line 1020 contains a POKE to Memory Location 150. This POKE forces CoCo to send data to the printer at 9600 bits per second. CoCo's normal speed is 600 CPS, or BAUD, as it is called. To avoid the BAUD rate speedup, delete the POKE from yourversion of the program.

The other POKE in Line 1020 insures that the label will be printed in upper case capital letters. To see what else can happen, change it to POKE 282,0. Or you can press <SHIFT 0> to get the same result.

Have trouble making this bit of Transylvanian technology work? Write me c/o 68 Micro Journal. Send along a printed listing of the program, and tell me what your printer is and is supposed to do. Also tell me what the program does wrong. A SASE will speed my reply. I'll answer any mail, though.

---

# OS-9 SETIME Module


OE Grovee
10207 Gillette
Lenexa Kan. 66215


**Editors Note: --- Oops! ! !**

One of the strong points of 68 Micro Journal has been the accuracy with which we have been able to print Program Listings by transferring the Source directly from disk to the Magazine. Of course, there is some Hardware BETWEEN the Disk Source and the Printout for the Magazine, and this is what let us down this time. Oral called us right after he received his Aug. '85 Issue of the Magazine and asked, in effect, "Is THAT what I sent on the Disk?". We did some "De-Bugging" and found a Printer causing problems; well, I suppose it was bound to happen sooner or later.

After carefully (we HOPE) comparing the Disk File with the Listing, we decided that we could point out the corrections, since few of them actually affect the CODE itself.

Most of the problems showed up on page 44 of last months Listing. In the remarks at the top of the left Column, the "NOT:" should be "NOTE" and the "alpha-natic" should be "alpha-numeric" in the general opening remarks. About 1/3 of the way down that column following the NOTE on the Robertson board the remark about the 'debugging code' should read "Invoke debugging code".

At the top of the right column on page 44, the remark "reserve at lea(t 200 bytes decimal r the" should read "reeerve at least 200 bytes decimal for the stack...".

Right below that problem, the 17th line from the top of the page, is a labeled statement which says

```
      daytib fcc 'SunMonTueWedThuFriSat'
```

which should read

```
      daytab fcc 'SunMon.....
```

In the remarks following that mis-print, the "Initialize storage area" should be "Initialize storage area".

Moving down to the middle of that same column, on the 46th line of the right column of page 44, the statement

```
      sta dow save diy of week
```

should read

```
      sta dow save day of week
```

Finally, moving to the very last line in the left column of page 45, the "OUTPU " should be "OUTPUT".

Since the ONLY actual "code" affected was the 'daytab' label, and the rest were in remarks that make understanding what is going on easier, we decided to offer this correction rather than completely rerunning all of the Code. Oh yeh! The Printer is headed for the "Doctor's Office". Sorry if it caused you any problems.

--- RLN ---

```
*
 bcs inperr
 lda todyy convert year to hex
 lber bcdhx
 sta todyy
 lbra endwhl
pal0 decb
 bne pall numeep = 1? - do Mooth
 leey todmth,u
 ldb #$12
 lber getnum
*
* * *   conditionally invoke debug code
*
 ifeq debug-2
 pshs x,y,d,cc
 lber shorege
 leax svd,u
 ldy #(15+24+35)
 lber shomem
 puls x,y,d,cc
 endc
*
*
 inc flg
 bcs inperr
 lda todmth
```

```
 lbsr bcdhx
 sta todmth
 bra endwhl
pal1 decb
 bne inperr numsep =2? - do Day of Month
 leay todd,u
 ldb #$31
 bra pal6
*
pa9 decb
 bne inperr at time string? - CASE numsep
 ldb numsep
 bne pa13 numsep =0? - do Hours
 leay todhh,u
 ldb #$23
 bra pal6
pa13 decb
 bne pa14 numsep = 1? - do Minutes
 leay todmm,u
 bra pal5
pa14 decb
 bne inperr numsep = 2? - do Seconds
 leay todss,u
pal5 ldb #$60
pal6 lbsr getnum
*
* * *   conditionally invoke debug code
*
 ifeq debug-2
 pshs x,y,d,cc
 lbsr showregs
 leax svd,u
 ldy #(15+24+35)
 lbsr showmem
 puls x,y,d,cc
 endc
*
*
 inc flg
 bra inperr
endwhl lbra pal
inperr lda #1 send message to stdout
 ldy #35
 leax dtmini,pcr display correct syntax
 os9 i$writln
 comb Bad input
 ldb #bedinp
 rts
*
*    This routine converts a BCD byte to hex
*
*    input                    output
* a  =byte (in BCD)      a     = byte (in hex)
*
*     bcdhx calls:        bcdhx is called by:
*           noone               parse
*
*
bcdhx equ *
 pshs a save LS nibble
 ldb #$f
 andb ,s
 stb ,s
 anda #$f0 keep MS nibble in A
 ldb #160 mult MS nbl X10 & keep it in A
 mul
 adda ,s+ add in LS nibble
 clrb
 rts back to caller
*
*    This routine gets the next character in
*    the input buffer
*
* input                outpu
* x    = buff addr  x     = moved to next char
* u,dp = parm area  u,dp = unchanged
*                   cc    = carry if nonalphno
*                         overflow if nonalpha
*                   a     = character
*
*  nxtch is called by:          nxtch calls:
*        parse                        noone
*        getnum
*        getdow
*
*    Class sets condition codes based on the
*    type of character in the A register.
*
* class is called by:          class calls:
*       getnum                       noone
*       getdow
*
*
nxtch equ *
 lda ,x+
class cmpa #$30
 bcc nxtn
 cmpa #$39
 ble nxtnu
 cmpa #$41
 bcs nxtn
 cmpa #$5a
 bls nxta
 cmpe #$61
 bcs nxtn
 cmpa #$7a
 bhi nxtn
*
*    alpha-numeric
*
nxta andcc #$fc clr C & V
 rts
*
*    numeric
*
nxtnu andcc #$fe clr carry
 orcc #2 set overflow
 rts
*
*  neither
*
nxtn orcc #1 set carry
 rts
* * * * * *
 ifne Rbtan
*
*       This routine expects to find 1 or 2
* alpha characters on the line before a
* delimiter.  It converts them into a hex
* number corresponding to a day of the week (0=
* Sunday). No attention is paid to any but the
* first two characters. Once these have been
* found, the buffer pointer is moved to the
* next delimiter on the line.
*
*  input                    output
* x    = buff pointer      x     = pointer
*                                 to next
*                                 delimiter
* u,dp = parm area         u,dp = unchanged
* a    = first character   d     = destroyed
*                          cc    = carry if
*                                 problem
*
* getdow calls:            getdow is called
*                          by:
*         nxtch                      parse
*
*
getdow equ *
 bsr class what is 1st character?
 bcs gtder not alpha-num
 bvc gtder not alpha
 anda #$5f upcase
 sta svd save 1st letter
 bsr nxtch get next character
 bcc gtder not alpha-num
 bvc gtder no alpha
 anda  #$5f upcase
 pshs a,x save char & buff ptr
 leax deytab,pcr get Day of week table
```

```
 clrb
 clr todow
gtd4 cmpb #21 end of table?
 bge gtderl yes- error
 lda b,x get char from table
 anda #$5f upcase it
 cmpa svd compare with 1st letter
 bne gtd1
 incb keep going
 lda b,x
 anda #$5f
 cmpa ,s compare with 2nd letter
 beq gtd2 got it
 decb
gtd1 addb #3 try again
 inc todow count to next day
 bra gtd4
gtderl pula a,x
gtder comb
 rts
gtd2 pula a,x
gtd5 bsr nxtch move to next sep
 bcc gtd5
 leax -1,x
 clrb
 rts
*
* * * * *    END OF GETDOW ROUTINE
 endc
*
* * * * * * * * *
*    This routine gets the next 0 or 1
*    characters from the buff and
*    converts them to a BCD byte in the
*    indicated address.
*
* input                      output
* x    - buff pointer        x    - moved
*                                  to next
*                                  delimiter
* u,dp - parm area           u,dp - unchanged
* y    - addr to byte in     y    - unchanged
* b    - max legal value     b    - destroyed
* a    - first character     a    - destroyed
*                            [y]  - value in
*                                   BCD format
*                            cc   - carry if
*                                   problem
*
* getnum calls:              getnum is called
*                            by:
*       nxtch                      parse
*
getnum equ *
 pshs d keep both MS nbl & max val
 bsr clase find out what's in A
 bcc gtner not alph-num
 bvc gtner not a number
 ldb #16 move MS nibble to left
 mul
 stb ,s save on stack
 bsr nxtch get LS
 bcc gtn2 only one digit
 bvc gtner not a number
 anda #$f Take off Character part
 ora ,s put into LS nibble
 bra gtn3
gtn2 ldb ,s put it back ...
 lda #16 ...into LS position
 mul
 leax -1,x put buff ptr back
gtn3 cmpa 1,s max value?
 bhi gtner yes-error
 sta ,y keep it
gtn4 lbsr nxtch use only 2 digits
 bcc gtn4
 leax -1,x set buff ptr for next char
 clrb no error
 puls d,pc rtrn
gtner comb set carry
 puls d,pc rtrn
*
*
* * * This part of the program sets the time
*     from the TOD packet
*
doset leax todyy,u get TODset packet
 stx svx keep addr of time packet
*
 ifeq Rbtsn convert to normal time format
 lda todd
 lbsr bcdhx
 sta todd
 lda todh
 lbsr bcdhx
 sta todh
 lda todm
 lbsr bcdhx
 sta todm
 lda tods
 lbsr bcdhx
 sta tods
 endc
*
 os9 f$stim set year
 bcc problem
*
*
 ifne Rbtsn
 tst flg
 beq noset don't change anything but year
 lda #$c1 module type- program
 leax todstr,pcr find TODset module
 os9 f$link
 bcc test ok? set the time
 cmpb #$DD e$mnf-module not in memory?
 bne problem other problem
 lda #$c1 still a program
 leax todstr,pcr
 os9 f$load get from x-dir
 bcs problem can't find it ? quit
test ldx svx get TODset packet
 jsr ,y set time
 os9 f$unlk remove module
 endc
noset clrb normal exit
problem rts return
*
* * * *  end of input and set
*
*
* This part of the program displays the current
*     time and requests user input
*
display equ *
 ifne Rbtsn
 ldb dow get dey of week
 cmpb #6 only 7 days in a week
 bhi rngerr return range error
 leax daytab,pcr
 lda #3 offset 3 bytes per entry
 mul
 abx  add offset to x
 leay dtmstr,u point to message buffer
 ldb ,x+ move day into buffer
 stb ,y+
 ldb ,x+
 stb ,y+
 ldb ,x+
 stb ,y++
 endc
 ifeq Rbtsn
 leay dtmsnr,u
 endc
 leax syeyy,u convert day number to strng
 lda #6 six bytes to convert
 pshs a
 ldb ,x+
dislp bsr hxbcd convert to bcd in D
 bsr binchr convert to-char in D
```

```
std ,y++ store two digits
leay 1,y skip separator
ldb ,x+ get next byte
dec ,s dec counter
bne dislp
puls s
ifne Rbtsn
leax dtmstr,u
endc
ifeq Rbtsn
leax dtmnr,u
endc
lds #1 write time to stdout
ldy #35
os9 i$wrln
bcs problem
ifne Rbtsn
leax okstr,pcr
endc
ifeq Rbtsn
leax oknr,pcr
endc
lds #1
ldy #35

leax buff,u
os9 i$rdln
lbcs problem
tfr y,d
rts
rngerr comb
ldb #badrng value out of range
rts
*
* Convert 1 hex byte to two ascii characters
*    in D
*
binchr pshs b save number
lds #16
mul  move most sig nibble to A reg
adds #$30 make a char
cmps #$39 HEX ?
ble bin1 decmal
adds #7
bin1 puls b do least sig nibble
andb #$f
addb #$30
cmpb #$39
ble bin2
addb #7
bin2 rts
*
* * * *  end display and request user input
*
*
* This part of the program converts the system
*    time packet to TODset format
*
hxbcd clr huncnt start with 0000
 clr tencnt
 clrs
hx0 subd #100 divide by 100
 blt hx1
 inc huncnt
 bra hx0
hx1 addb #100
hx3 subb #10 divide by 10
 blt hx2
 inc tencnt
 bra hx3
hx2 addb #10 restore remainder
 asl tencnt move tens into position
 asl tencnt
 asl tencnt
 asl tencnt
 addb tencnt add ones
 lds huncnt get hundreds
 rts return
*
*
convert lds todyy
 bpl cnvt0 year changed
 lds sysyy
 sts todyy
cnvt0 lds todmth
 bpl cnvt1 month changed
 lds sysmth keep cur month
*
 ifeq Rbtsn
cnvt1 sts todmth
 endc
*
 ifne Rbtsn
cnvt1 ldb #16 shift to left nibble
 mul
 lds todow
 bmi cnvt2 day of week unchanged
 sts dow put new day in
cnvt2 orb dow combine mth & day
 stb todmth
 endc
*
 leax sysdd,u point at rest of packet
 leay todd,u
 lds #4
 pshs s
cvlp tst ,y no change?
 bpl cvlp1
 ldb ,x keep cur byte
 bsr hxbcd convert to bcd
 stb ,y
cvlp1 leax 1,x goto next byte
 leay 1,y
 dec ,s count 5 bytes
 bne cvlp
 puls s,pc return
*
*
* * * *  end convert to TODset format
*
* * * *  E N D  O F  P R O G R A M  * * *
*
 emod
endset equ *
```

# COBOL Name & Address System

by Mike Martin

**Continued from last month.**

```
B100-NEW-DATA SECTION.
    MOVE SPACES TO WS-NEW-RECORD.
B101-FIRST.
    MOVE 5 TO X.
    MOVE 13 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NEW-FIRST-NAME FROM CONSOLE.
    IF WS-NEW-FIRST-NAME = "<" OR "@"
        GO TO B199X.
B102-LAST.
    MOVE 5 TO X.
    MOVE 45 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NEW-LAST-NAME FROM CONSOLE.
    IF WS-NEW-LAST-NAME = "<"
        GO TO B101-FIRST.
    IF WS-NEW-LAST-NAME = "@"
        GO TO B190-CONFIRM.
B103-LINE1.
    MOVE 7 TO X.
    MOVE 13 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NEW-LINE1 FROM CONSOLE.
    IF WS-NEW-LINE1 = "<"
```

```
        GO TO B102-LAST.
    IF WS-NEW-LINE1 = "@"
        GO TO B190-CONFIRM.
B104-LINE2.
    MOVE 7 TO X.
    MOVE 45 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NEW-LINE2 FROM CONSOLE.
    IF WS-NEW-LINE2 = "<"
        GO TO B104-LINE1.
    IF WS-NEW-LINE2 = "@"
        GO TO B190-CONFIRM.
B105-CITY.
    MOVE 9 TO X.
    MOVE 13 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NEW-CITY FROM CONSOLE.
    IF WS-NEW-CITY = "<"
        GO TO B104-LINE2.
    IF WS-NEW-CITY = "@"
        GO TO B190-CONFIRM.
B106-STATE.
    MOVE 9 TO X.
    MOVE 31 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NEW-STATE FROM CONSOLE.
    IF WS-NEW-STATE = "<"
        GO TO B105-CITY.
    IF WS-NEW-STATE = "@"
        GO TO B190-CONFIRM.
B107-ZIP.
    MOVE 9 TO X.
    MOVE 39 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NEW-ZIP FROM CONSOLE.
    IF WS-NEW-ZIP = "<"
        GO TO B106-STATE.
    IF WS-NEW-ZIP = "@"
        GO TO B190-CONFIRM.
B108-PHONE.
    MOVE 9 TO X.
    MOVE 54 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NEW-PHONE FROM CONSOLE.
    IF WS-NEW-PHONE = "<"
        GO TO B107-ZIP.
    IF WS-NEW-PHONE = "@"
        GO TO B190-CONFIRM.
B109-XMAS.
    MOVE 11 TO X.
    MOVE 12 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NEW-XMAS FROM CONSOLE.
    IF WS-NEW-XMAS = "<"
        GO TO B108-PHONE.
    IF WS-NEW-XMAS = "@"
        GO TO B190-CONFIRM.
B110-PARTY.
    MOVE 12 TO X.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NEW-PARTY FROM CONSOLE.
    IF WS-NEW-PARTY = "<"
        GO TO B109-XMAS.
    IF WS-NEW-PARTY = "@"
        GO TO B190-CONFIRM.
B111-BUSINESS.
    MOVE 13 TO X.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NEW-BUSINESS FROM CONSOLE.
    IF WS-NEW-BUSINESS = "<"
        GO TO B110-PARTY.
    IF WS-NEW-BUSINESS = "@"
        GO TO B190-CONFIRM.
B112-BILL.
    MOVE 14 TO X.
    MOVE 12 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    ACCEPT WS-NEW-BILL FROM CONSOLE.
    IF WS-NEW-BILL = "<"
        GO TO B111-BUSINESS

B190-CONFIRM.
    MOVE 16 TO X.
    MOVE 1 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM Z020-EOL-ERASE THRU Z020X.
    DISPLAY 1-0 "Enter 'Y' to confirm update "
        UPON CONSOLE
    ACCEPT WS-RESPONSE FROM CONSOLE.

IF WS-RESPONSE = "<"
    GO TO B112-BILL.
IF WS-RESPONSE = "C"
    GO TO B199X.
IF WS-RESPONSE NOT = "Y"
    MOVE SPACES TO WS-NEW-RECORD
    GO TO B101-FIRST.
IF WS-NEW-FIRST-NAME = "@"
    NEXT SENTENCE
ELSE
IF WS-NEW-FIRST-NAME NOT > " "
    NEXT SENTENCE
ELSE
    IF WS-NEW-FIRST-NAME = "-"
        MOVE SPACES TO WS-NAR-FIRST-NAME
    ELSE
        MOVE WS-NEW-FIRST-NAME TO WS-NAR-FIRST-NAME.
IF WS-NEW-LAST-NAME = "@"
    NEXT SENTENCE
ELSE
IF WS-NEW-LAST-NAME NOT > " "
    NEXT SENTENCE
ELSE
    IF WS-NEW-LAST-NAME = "-"
        MOVE SPACES TO WS-NAR-LAST-NAME
    ELSE
        MOVE WS-NEW-LAST-NAME TO WS-NAR-LAST-NAME.
IF WS-NEW-LINE1 = "@"
    NEXT SENTENCE
ELSE
IF WS-NEW-LINE1 NOT > " "
    NEXT SENTENCE
ELSE
    IF WS-NEW-LINE1 = "-"
        MOVE SPACES TO WS-NAR-LINE1
    ELSE
        MOVE WS-NEW-LINE1 TO WS-NAR-LINE1.
IF WS-NEW-LINE2 = "@"
    NEXT SENTENCE
ELSE
IF WS-NEW-LINE2 NOT > " "
    NEXT SENTENCE
ELSE
    IF WS-NEW-LINE2 = "-"
        MOVE SPACES TO WS-NAR-LINE2
    ELSE
        MOVE WS-NEW-LINE2 TO WS-NAR-LINE2.
IF WS-NEW-CITY = "@"
    NEXT SENTENCE
ELSE
IF WS-NEW-CITY NOT > " "
    NEXT SENTENCE
ELSE
    IF WS-NEW-CITY = "-"
        MOVE SPACES TO WS-NAR-CITY
    ELSE
        MOVE WS-NEW-CITY TO WS-NAR-CITY.
IF WS-NEW-STATE = "@"
    NEXT SENTENCE
ELSE
IF WS-NEW-STATE NOT > " "
    NEXT SENTENCE
ELSE
    IF WS-NEW-STATE = "-"
        MOVE SPACES TO WS-NAR-STATE
    ELSE
        MOVE WS-NEW-STATE TO WS-NAR-STATE.
IF WS-NEW-ZIP = "@"
    NEXT SENTENCE
ELSE
IF WS-NEW-ZIP NOT > " "
    NEXT SENTENCE
```

## K-BASIC UPDATES

**K-BASIC updates are now available. If you purchased K-BASIC prior to July 1, 1985 and wish to have your K-BASIC updated, please send $35 enclosed with your master disk to Southeast Media.**

TELEX 558 414 PVT BTH

(615)842-4600

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 BASIC, XBASIC, and XPC Source Code Files

**K-BASIC** now makes the multitude of TSC BASIC Software available for use under OS-9. Transfer your favorite BASIC Programs to OS-9, compile them, Assemble them, and RUN!! -- usable, multi-precision, familiar Software is running under your favorite Operating System!

**K-BASIC** (os-9 or FLEX), **including** the Assembler

**$199.00**

# SCULPTOR

**Microprocessor Developments Ltd.'s Commercial Application Generator** Program provides a **FAST** Commercial Application Development tool unavailable to the OS-9 and UniFLEX User before. Develop any Commercial Application in 20% of the normal required time; gain easy updating or customizing. PLUS, the Application can also be run on MS-DOS and Unix machines! Sculptor handles input validation, complex calculations, and exception conditions as well as the normal collecting, displaying, reporting, and updating information in an orderly fashion. Key fields to 160 bytes; unlimited record size; file size should be held to 17 million records. Utilizes ISAM File Structure and B-tree Key files for rapid access. Input and Output communication with other programs and files plus a library of ISAM routines for use with C Programs. Run-time included w/ the Development package; a compiled Application only needs a Run-time License. Additional charge for Networked Units. Prices for **Development Package/Run-time.** Discounts available for purchases of 5 **or more** Run-time Packages.

UniFLEX -- $1450.00/337.00
OS-9 Lvl II -- $995.00/200.00

Unix -- up to $2535/675 -- Call for info!
IBM PC & compats -- $635.00/170.00

**BasicO9 Tour Guide**

by Dale Puckett -- An excellent Book on using OS-9. Oriented towards using the powerful **BasicO9 Programming Language**, it also contains a lot of good information on using OS-9 in general.
Normally $18.95
**Special ---- NOW only $16.00**

**----- FLEX Software -----**

| | | | |
|---|---|---|---|
| TSC "Flex Utilities" | was $75.00, | **NOW only** | **$65.00** |
| TSC "Sort Merge" | was $75.00, | **NOW only** | **$65.00** |
| TSC "6809 Basic" | was $75.00, | **NOW only** | **$65.00** |
| TSC "Extended Basic" | was $100.00, | **NOW only** | **$90.00** |
| TSC "DeBug" | was $75.00, | **NOW only** | **$65.00** |
| TSC "FLEX Diagnostics" | was $75.00, | **NOW only** | **$65.00** |
| TSC "Text Processing System" | was $75.00, | **NOW only** | **$65.00** |
| TSC "68000 Cross Assembler" | was $250.00, | **NOW only** | **$199.95** |

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

VISA MasterCard

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 !!!**

TELEX 558 414 PVT BTH

(615)842-4600

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343

for information
call (615) 842-4601

CoCo OS-9™ FLEX™

SOFTWARE


# PROGRAMMING

## ASSEMBLERS

**ASTRUK09** from **Southeast Media** -- A "Structured Assembler for the 6809" which requires the **TSC Macro Assembler.** F, CCF - **$99.95**

**Macro Assembler** for TSC -- The FLEX **STANDARD** Assembler.
Special -- CCF $35.00; F $50.00

**OSA** Extended 6809 Macro Assembler from **Lloyd I/O.** -- Provides local labels, Motorola S-records, and Intel Hex records; XREF. Generate OS-9 Memory modules under FLEX. FLEX, CCF, OS-9 $99.00

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

**MACE**, by Graham Trott from **Windrush Micro Systems** -- Co-Resident Editor and Assembler; fast interactive A.L. Programming for small to medium-sized Programs. F,CCF - $75.00

**6HACK** -- HACK w/ Cross Assembler for 6800/1/2/3/8 F,CCF - $98.00

**TRUE CROSS ASSEMBLERS** from **Computer Systems Consultants** -- Supports 1802/S, Z-80, 6800/1/2/3/8/11/HC11, 6804, 6805/NCOS/146805, 6809/00/01. 6502 family, 8080/5, 8020/1/2/35/C35/39/40/48/C48/49/C49/50/8748/49, 8031/51/8751, and 68000 Systems. Assembler and Listing formats same as target CPU's format. Produces machine independent Motorola S-Text.
FLEX, CCF, OS-9, UniFLEX each - $50.00
any 3 - **$100.00**
the complete set **w/ C Source** (except the 68000 Source) - **$200.00**

**XASM** Cross Assemblers for **FLEX** from **Computense Ltd.** -- This set of 6800/1/2/3/S/8, 6301, 6502, 8080/5, and Z80 Cross Assemblers uses the familiar **TSC Macro Assembler** Command Line and Source Code format, Assembler options, etc., in providing code for the target CPU's. Complete set, FLEX only - $150.00

**CRASM8** from **Lloyd I/O** -- 8-Bit Macro Cross Assembler with same features as OSM; cross-assemble to 6800/1/2/3/4/5/8/9/11, 6502, 1802, 8048 Sers, 80/85, Z-8, Z-80, TMS-7000 sers. Supports the target chip's standard mnemonics and addressing modes.
FLEX, CCF, OS-9 Full package -- $399.00

**CRASM 16.32** from **Lloyd I/O** -- Cross Assembler for the **68000.**
FLEX, CCF, OS-9 $249.00

## DISASSEMBLERS

**SUPER SLEUTH** from **Computer Systems Consultants** -- Interactive Disassembler; extremely POWERFUL! Disk File Binary/ASCII Examine/Change, Absolute or FULL Disassembly. XREF Generator, Label "Name Changer", and Files of "Standard Label Names" for different Operating Systems

| Color Computer | SS-50 Bus (all w/ A.L. 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. | |

**DYNAMITE +** from **Computer Systems Center** -- Excellent standard "Batch Mode" Disassembler. Includes XREF Generator and "Standard Label" Files. Special OS-9 options w/ OS-9 Version.

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

## PROGRAMMING LANGUAGES

**PL/9** from **Windrush Micro Systems** -- By Graham Trott. A combination Editor/Compiler/Debugger. Direct source-to-object compilation delivering fast, compact, re-entrant, ROM-able, PIC. 8 & 16-bit Integers & 6-digit Real numbers for all real-world problems. Direct control over ALL System resources, including Interrupts. Comprehensive library support; simple Machine Code interface; step-by-step tracer for instant debugging. **500+ page Manual** with tutorial guide. F, CCF - **$198.00**

**WHIMSICAL** from **Whimsical Developments** -- Now supports **Real Numbers.** "Structured Programming" WITHOUT losing the Speed and Control of Assembly Language! Single-pass Compiler features unified, user-defined I/O; produces ROMable Code; Procedures and Modules (including pre-compiled Modules); many "Types" up to 32 bit Integers, **6-digit Real Numbers,** unlimited sized Arrays (vectors only); Interrupt handling; long variable Names; Variable Initialization; Include directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. Run-Time subroutines inserted **as called** during compilation. Normally produces 10% less code than PL/9. F and CCF - **$195.00**

**C Compiler** from **Windrush Micro Systems** by James McCosh. Full C for FLEX except bit-fields, including an Assembler. Requires the TSC Relocating Assembler if user desires to implement his own Libraries. F and CCF - **$295.00**

**C Compiler** from **Introl** -- Full C except Doubles and Bit Fields, streamlined for the 6809. Reliable Compiler; FAST, efficient Code. More UNIX Compatible than most.
F, CCF, and O - **$375.00** U - **$425.00**

**PASCAL Compiler** from **Lucidata** -- ISO Based P-Code Compiler. Designed especially for Microcomputer Systems. Allows linkage to Assembler Code for maximum flexibility.
F and CCF 5" - **$190.00** F 8" - **$205.00**

**PASCAL Compiler** from **OmegaSoft** (now **Certified Software**) -- For the **PROFESSIONAL;** ISO Based, Native Code Compiler. Primarily for Real-Time and Process Control applications. Powerful; Flexible. Requires a "Motorola Compatible" Relo. Assb. and Linking Loader.
F and CCF - $425.00 One Year Maint. - $100.00

**X-BASIC** from **LLOYD I/O** -- A "Native Code" **BASIC Compiler** which is now Fully TSC XBASIC compatible. The compiler compiles to Assembly Language Source Code. A NEW, **streamlined,** Assembler is now included allowing the assembly of LARGE Compiled X-BASIC Programs. Conditional assembly reduces Run-time package.
FLEX, CCF, OS-9 Compiler **with Assembler - $199.00**

**CRUNCH COBOL** from **Computense Ltd.** -- Supports large subset of ANSII Level 1 COBOL with many of the useful Level 2 features. Full FLEX File Structures, including Random Files and the ability to process Keyed Files. Segment and link large programs 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 in effect) -- **$99.95**

**FORTH** from **Stearns Electronics** -- A CoCo **FORTH** Programming Language. Tailored to the CoCo! Supplied on Tape, transferable to disk. Written in FAST ML. Many CoCo functions (Graphics, Sound, etc.). Includes an Editor, **Trace,** etc. Provides CPU Carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. for the "Pro". Excellent "Learning" tool!
Color Computer **ONLY - $58.95**


** SHIPPING **
Add 3% U.S.A.
(min. $2.50)
Add 5% Surface Foreign
10% Air Foreign

VISA

SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
Info (615) 842-4601
SOFTWARE

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


**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 !!!**

# GENERAL

## SOFTWARE DEVELOPMENT

**BasicO9 XRef** from **Southeast Media** -- 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 includes a **Program List Utility** which outputs a fast "pretty printed" listing with line numbers. Requires BasicO9 or RunB.
O & CCD obj. only -- $39.95; w/ Source - $79.95

**Lucidata PASCAL UTILITIES** (**Requires LUCIDATA Pascal ver 3**)

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

**INCLUDE** -- Include other Files in a Source Text, including Binary; unlimited nesting capabilities.

**PROFILER** -- provides an Indented, Numbered, "Structogram" of a Pascal Source Text File; view the overall structure of large programs, program integrity, etc. Supplied in Pascal Source Code; requires compilation.

F, CCF --- **EACH Utility** 5" - $40.00, 8" - $50.00

**DUB** from **Southeast Media** -- A **UniFLEX** "basic" De-Compiler. Re-Create a Source Listing from UniFLEX **Compiled** basic Programs. Works w/ ALL Versions of 6809 UniFLEX basic. U - $219.95

**FULL SCREEN FORMS DISPLAY** from **Computer Systems Consultants** -- **TSC Extended BASIC** program supports any Serial Terminal with Cursor Control or Memory-Mapped Video Displays; substantially extends the capabilities of the Program Designer by providing a table-driven method of describing and using Full Screen Displays.
F and CCF, U - $25.00, w/ Source - $50.00

## DISK UTILITIES

**OS-9 VDisk** from **Southeast Media** -- For **Level I** only. Use the Extended Memory capability of your SWTPC or Gimix CPU card (or similar format DAT) for FAST Program Compiles, CMD execution, high speed inter-process communications (without pipe buffers), etc. - SAVE that System Memory. Virtual Disk size is variable in 4K increments up to 960K. Some Assembly Required.
-- **Level I ONLY** -- OS-9 obj. only - $79.95; w/ Source - $149.95

**O-F** from **Southeast Media** -- Written in BASICO9 (with Source), includes: **BEFORMAT**, a BASICO9 Program that reformats a chosen amount of an OS-9 disk to **FLEX** Format so it can be used normally by FLEX; and **FLEX**, a BASICO9 Program that does the actual **read** or **write** function to the special **O-F Transfer Disk**; user-friendly menu driven. Read the FLEX Directory, Delete FLEX Files, Copy both directions, etc. FLEX users use the special disk just like any other FLEX disk. O - $79.95

**COPYMULT** from **Southeast Media** -- Copy **LARGE** Disks to several **smaller disks.** FLEX utilities allow the backup of **ANY size disk** to any SMALLER size diskettes (Hard Disk to floppies, 8" to 5", etc.) by simply inserting diskettes as requested by **COPYMULT**. No fooling with directory deletions, etc. **COPYMULT.CMD** understands normal "copy" syntax and keeps up with files copied by maintaining directories for both host and receiving disk system. Also includes **BACKUP.CMD** to download any size "random" type file; **RESTORE.CMD** to restructure copied "random" files for copying, or recopying back to the host system; and **FREELINK.CMD** as a "bonus" utility that "relinks" the free chain of floppy or hard disk, eliminating fragmentation.
**Completely documented Assembly Language Source files included.**
ALL 4 Programs (FLEX, 8" or 5") $99.50

**COPYCAT** from **Lucidata** -- 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. Also includes some Utilities to help out. Programs supplied in **Modular Source Code** (Assembly Language) to help solve unusual problems.
F and CCF 5" - $90.00 F 8" - $65.00


TELEX 558 414 PVT BTH
(615)842-4600
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE


**FLEX DISK UTILITIES** from **Computer Systems Consultants** -- Eight (8) different Assembly Language (w/ Source Code) FLEX Utilities for every FLEX Users Toolbox: **Copy** a File with **CRC Errors**; **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 XBASIC Programs including: A **BASIC Resequencer** with **EXTRAS** over "RENUM" like check for missing label definitions, processes Disk to Disk instead of in Memory, etc. Other programs **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 **PROCOMPILER** BASIC Programs.
All Utilities **Include Source** (either BASIC or A.L. Source Code).
F and CCF - $50.00
**BASIC Utilities ONLY for UniFLEX** -- $30.00

## COMMUNICATIONS

**CMODEM Telecommunications Program from Computer Systems Consultants, Inc.** -- 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

**XDATA** from **Southeast Media** -- A **COMMUNICATION** Package for the UniFLEX Operating System. Use with CP/M, Main Frames, other UniFLEX Systems, etc. Verifies Transmission using checksum or CRC; Re-Transmits bad blocks, etc. U - $299.99

## GAME

**RAPIER - 6809 Chess Program from Southeast Media** -- 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

VISA MasterCard

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


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 !!!**

TELEX 558 414 PVT BTH
(615)842-4600
SOUTH EAST MEDIA
5900 Cassandra Smith Rd.
Hixson, TN 37343
for information
call (615) 842-4601
CoCo OS-9™ FLEX™
SOFTWARE

## BUSINESS

### WORD PROCESSING

**SCREDITOR III from Windrush Micro Systems** -- Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; over 300 pages of Documentation with Tutorial. Features Multi-Column display and editing, "decimal align" columns (AND add them up automatically), multiple keystroke macros, even/odd page headers and footers, imbedded printer control codes, all justifications, "help" support, store common command series on disk, etc. Use supplied "set-ups", or remap the keyboard to your needs. Except for proportional printing, this package will DO IT ALL!

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

**STYLO-GRAPH from Great Plains Computer Co.** -- A full-screen oriented WORD PROCESSOR -- (uses the 51 x 24 Display Screens on CoCo FLEX/STAR-DOS, or PBJ Wordpak). Full screen display and editing; supports the Daisy Wheel proportional printers.
NEW PRICES --> CCF and CCO - $99.95, F or O - $179.95, U - $299.95

**STYLO-SPELL from Great Plains Computer Co.** -- Fast Computer Dictionary. Complements Stylograph.
NEW PRICES --> CCF and CCO - $69.95, F or O - $99.95, U - $149.95

**STYLO-MERGE from Great Plains Computer Co.** -- Merge Mailing List to "Form" Letters, Print multiple Files, etc., through Stylo.
NEW PRICES --> CCF and CCO - $59.95, F or O - $79.95, U - $129.95

**STYLO-PAK** -- Graph + Spell + Merge Package Deal!!!
F or O - $329.95, U - $549.95

**JUST from Southeast Media** -- Text Formatter developed by Ron Anderson; for Dot Matrix Printers, provides many unique features. Output "Formatted" Text to the Display. Use the FPRINT.CMD supplied for producing multiple copies of the "Formatted" Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (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); up to ten (10) imbedded "Printer Control Commands". Compensates 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 - $89.95

**SPELL8 "Computer Dictionary" from Southeast Media** -- OVER 120,000 words! Look up a word from within your Editor or Word Processor (with the SPN.CMD Utility which operates in the FLEX UCS). Or check and update the Text after entry; ADD WORDS to the Dictionary, "Flag" questionable words in the Text, "View a word in context" before changing or ignoring, etc. SPELL8 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. SPELL8 also allows the use of Small Disk Storage systems.

F and CCF - $129.95

### DATA BASE - ACCOUNTING

**XDMS from Westchester Applied Business Systems** -- Powerful DBMS; M.I. program will work on a single sided 5" disk, yet is F-A-S-T. Supports Relational, Sequential, Hierarchical, and Random Access File Structures; has Virtual Memory capabilities for Giant Data Bases. XDMS Level I provides an "entry level" System for defining a Data Base, entering and changing the Data, and producing Reports. XDMS Level II adds the POWERFUL "GENERATE" facility with an English Language Command Structure for manipulating the Data to create new File Structures, Sort, Select, Calculate, etc. XDMS Level III adds special "Utilities" which provide additional ease in setting up a Data Base, such as copying old data into new Data Structures, changing System Parameters, etc.

XDMS System Manual - $24.95
XDMS Lvl I - F & CCF - $129.95
XDMS Lvl II - F & CCF - $299.95
XDMS Lvl III - F & CCF - $269.95

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

### MISCELLANEOUS

**TABULA RASA SPREADSHEET from Computer Systems Consultants** -- TABULA RASA is similar to DESKTOP/PLAN; provides use of tabular computation schemes used for analysis of business, sales, and economic conditions. Menu-driven; extensive report-generation capabilities. Requires TSC's Extended BASIC.

F and CCF, U - $50.00, w/ Source - $100.00

**DYNACALC from Computer Systems Center** -- Electronic Spread Sheet for the 6809.

F and SPECIAL CCF - $200.00, U - $395.00

**FULL SCREEN INVENTORY/MRP from Computer Systems Consultants** -- Use the Full Screen Inventory System/Materials Requirement Planning for maintaining inventories. Keeps item field file in alphabetical order for easier inquiry. Locate and/or print records matching partial or complete item, description, vendor, or attributes; find backorder or below stock levels. Print-outs in item or vendor order. MRP capability for the maintenance and analysis of Hierarchical assemblies of items in the inventory file. Requires TSC's Extended BASIC.

F and CCF, U - $50.00, w/ Source - $100.00

**FULL SCREEN MAILING LIST from Computer Systems Consultants** -- The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Locate all records matching on partial or complete name, city, state, zip, or attributes for Listings or Labels, etc. Requires TSC's Extended BASIC.

F and CCF, U - $50.00, w/ Source - $100.00

**DIET-TRAC Forecaster from Southeast Media** -- An XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P &) 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. Provides number of days and daily calendar after weight goal and calorie plan is determined.

F - $59.95, U - $89.95

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

VISA MasterCard

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

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

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

```
ELSE
    IF WS-NEW-ZIP = "-"
        MOVE SPACES TO WS-NAR-ZIP
    ELSE
        MOVE WS-NEW-ZIP TO WS-NAR-ZIP.
IF WS-NEW-PHONE = "0"
    NEXT SENTENCE
ELSE
IF WS-NEW-PHONE NOT > " "
    NEXT SENTENCE
ELSE
    IF WS-NEW-PHONE = "-"
        MOVE SPACES TO WS-NAR-PHONE
    ELSE
        MOVE WS-NEW-PHONE TO WS-NAR-PHONE.
IF WS-NEW-XMAS = "0"
    NEXT SENTENCE
ELSE
IF WS-NEW-XMAS NOT > " "
    NEXT SENTENCE
ELSE
    IF WS-NEW-XMAS = "-"
        MOVE SPACES TO WS-NAR-XMAS
    ELSE
        MOVE WS-NEW-XMAS TO WS-NAR-XMAS.
IF WS-NEW-PARTY = "0"
    NEXT SENTENCE
ELSE
IF WS-NEW-PARTY NOT > " "
    NEXT SENTENCE
ELSE
    IF WS-NEW-PARTY = "-"
        MOVE SPACES TO WS-NAR-PARTY
    ELSE
        MOVE WS-NEW-PARTY TO WS-NAR-PARTY.
IF WS-NEW-BUSINESS = "0"
    NEXT SENTENCE
ELSE
IF WS-NEW-BUSINESS NOT > " "
    NEXT SENTENCE
ELSE

        IF WS-NEW-BUSINESS = "-"
            MOVE SPACES TO WS-NAR-BUSINESS
        ELSE
            MOVE WS-NEW-BUSINESS TO WS-NAR-BUSINESS.
    IF WS-NEW-BILL = "0"
        NEXT SENTENCE
    ELSE
    IF WS-NEW-BILL NOT > " "
        NEXT SENTENCE
    ELSE
        IF WS-NEW-BILL = "-"
            MOVE SPACES TO WS-NAR-BILL
        ELSE
            MOVE WS-NEW-BILL TO WS-NAR-BILL.
B199X. EXIT.
R100-READ SECTION.
    READ NAME-FILE INTO WS-NAME-ADDRESS-RECORD
        KEY IS WS-REC-NO
            INVALID KEY
            MOVE 1 TO EOF-FLAG
            MOVE "1" TO WS-LAST-NAME
            MOVE "1" TO WS-NAR-LAST-NAME.
R100X. EXIT.
Z000-TERMINAL-CONTROL SECTION.
Z001-DELAY.
    ADD 1 TO DELAY-VAR.
Z001X. EXIT.
Z020-EOL-ERASE.
    DISPLAY I-O $05 UPON CONSOLE.
    MOVE ZEROES TO DELAY-VAR.
    PERFORM Z001-DELAY THRU Z001X
        UNTIL DELAY-VAR > 600.
Z020X. EXIT.
Z110-ACA.
    DISPLAY I-O $1B $3D
        ACA-ADDRESS (X) ACA-ADDRESS (Y)
        UPON CONSOLE.
Z110X. EXIT.
```

```
IDENTIFICATION DIVISION.
PROGRAM-IO. NAMP004B.
ENVIRONMENT DIVISON.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT NAME-FILE ASSIGN TO "WNAMD0001DAT"
        ACCESS MODE IS RANDOM
        FILE STATUS IS NAME-STAT.
DATA DIVISON.
FILE SECTION.
FD  NAME-FILE.
01  NAME-RECORD PIC X(126).
WORKING-STORAGE SECTION.
     COPY "WNAMCDUPDCBL".
    COPY "WWFCCDTOOCBL".
    COPY "WNAMCDMSTCBL".
PROCEDURE DIVISION.
C100-CLEAR-FIELDS SECTION.
    MOVE "1" TO OV-SWITCH.
    MOVE 5 TO X.
    MOVE 13 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM C200-DISPLAY-DASH THRU C200X 20 TIMES.
    MOVE 45 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM C200-DISPLAY-DASH THRU C200X 15 TIMES.
    MOVE 7 TO X.
    MOVE 13 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM C200-DISPLAY-DASH THRU C200X 20 TIMES.
    MOVE 45 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM C200-DISPLAY-DASH THRU C200X 20 TIMES.
    MOVE 9 TO X.
    MOVE 13 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM C200-DISPLAY-DASH THRU C200X 10 TIMES.
    MOVE 31 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM C200-DISPLAY-DASR THRU C200X 2 TIMES.
    MOVE 39 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM C200-DISPLAY-DASH THRU C200X 9 TIMES.
    MOVE 54 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM C200-DISPLAY-DASH THRU C200X 10 TIMES.
    MOVE 12 TO Y.
    MOVE 11 TO X.
    PERFORM C300-CLEAR-FLAGS THRU C300X 4 TIMES.
    MOVE 16 TO X.
    MOVE 1 TO Y.

    PERFORM Z110-ACA THRU Z110X.
    PERFORM Z020-EOL-ERASE THRU Z020X.
    MOVE 5 TO X.
    MOVE 45 TO Y.
    PERFORM Z110-ACA THRU Z110X.
    CALL "NAMP004A" OVERLAY.
C100X. EXIT.
C200-DISPLAY-DASH.
    DISPLAY I-O "_" UPON CONSOLE.
C200X. EXIT.
C300-CLEAR-FLAGS.
    PERFORM Z110-ACA THRU Z110X.
    PERFORM C200-DISPLAY-DASH THRU C200X.
    ADD 1 TO X.
C300X. EXIT.
Z000-TERMINAL-CONTROL SECTION.
Z001-DELAY.
    ADD 1 TO DELAY-VAR.
Z001X. EXIT.
Z020-EOL-ERASE.
    DISPLAY I-O $05 UPON CONSOLE.
    MOVE ZEROES TO DELAY-VAR.
    PERFORM Z001-DELAY THRU Z001X
        UNTIL DELAY-VAR > 600.
Z020X. EXIT.
Z110-ACA.
    DISPLAY I-O $1B $3D
        ACA-ADDRESS (X) ACA-ADDRESS (Y)
        UPON CONSOLE.
Z110X. EXIT.
```

```
Z000-TERMINAL-CONTROL SECTION.
Z001-DELAY.
    ADD 1 TO DELAY-VAR.
Z001X. EXIT.
Z010-HOME-CURSOR.
    DISPLAY I-O $04 UPON CONSOLE.
Z010X. EXIT.
Z020-EOL-ERASE.
    DISPLAY I-O $05 UPON CONSOLE.
    MOVE ZEROES TO DELAY-VAR.
    PERFORM Z001-DELAY THRU Z001X
        UNTIL DELAY-VAR > 800.
Z020X. EXIT.
Z030-EOS-ERASE.
    DISPLAY I-O $06 UPON CONSOLE.
    MOVE ZEROES TO DELAY-VAR.
    PERFORM Z001-DELAY THRU Z001X
        UNTIL DELAY-VAR > 100.
Z030X. EXIT.
Z040-BACK-SPACE-CURSOR.
    DISPLAY I-O $08 UPON CONSOLE.
Z040X. EXIT.
Z050-HOR-TAB-CURSOR.
    DISPLAY I-O $09 UPON CONSOLE.
Z050X. EXIT.
Z060-LINE-FEED.
    DISPLAY I-O $0A UPON CONSOLE.
Z060X. EXIT.
Z070-RETURN-CURSOR.
    DISPLAY I-O $0D UPON CONSOLE.
Z070X. EXIT.
Z080-VERT-TAB-CURSOR.
    DISPLAY I-O $0B UPON CONSOLE.
Z080X. EXIT.
Z090-CLEAR-SCREEN.
    DISPLAY I-O $0C UPON CONSOLE.
    MOVE ZEROES TO DELAY-VAR.
    PERFORM Z001-DELAY THRU Z001X
        UNTIL DELAY-VAR > 200.
Z090X. EXIT.
Z100-DEL-CHAR.
    DISPLAY I-O $7F UPON CONSOLE.
Z100X. EXIT.
Z110-ACA.
    DISPLAY I-O $1B $3D
        ACA-ADDRESS (X) ACA-ADDRESS (Y)
        UPON CONSOLE.
Z110X. EXIT.

*****************************************************
*                                                   *
*   The code in this copy member clears the         *
*   screen and displays the standard                *
*   WESTPORK CONSULTING title on the first          *
*   two lines of the screen. Current date and       *
*   active program name are also displayed on       *
*   the first line.                                 *
*                                                   *
*****************************************************
     PERFORM Z090-CLEAR-SCREEN THRU Z090X.
     ACCEPT WS-DATE FROM DATE.
     MOVE WS-YEAR TO SLI-YEAR.
     MOVE WS-MON  TO SLI-MON.
     MOVE WS-DAY  TO SLI-DAY.
     DISPLAY I-O SLI-DATE UPON CONSOLE.
     MOVE 1 TO X.
     MOVE 19 TO Y.
     PERFORM Z110-ACA THRU Z110X.
     DISPLAY I-O WS-COMPANY UPON CONSOLE.
     MOVE 52 TO Y.
     PERFORM Z110-ACA THRU Z110X.
     DISPLAY I-O WS-PROG UPON CONSOLE.
     MOVE 2 TO X.
     MOVE 19 TO Y.
     PERFORM Z110-ACA THRU Z110X.
     DISPLAY I-O WS-TITLE UPON CONSOLE.
 01  PROG-NAME REDEFINES FILE-D.
     05  WS-PROG        PIC X(12).
     05  FILLER         PIC X(04).
 01  DELAY-VAR          PIC 1(2) VALUE ZEROES.
 01  WS-COMPANY         PIC X(25) VALUE
         "WESTPORK CONSULTING, INC.".
 01  WS-DATE.
     05  WS-YEAR        PIC X(2).
     05  WS-MON         PIC X(2).
     05  WS-DAY         PIC X(2).
 01  SLI-DATE.
     05  SLI-MON        PIC X(2).
     05  FILLER         PIC X(1)    VALUE "/".
     05  SLI-DAY        PIC X(2).
     05  FILLER         PIC X(1)    VALUE "/".
     05  SLI-YEAR       PIC X(2).
 01  ACA-TABLE.
     05  ACA-VALUES.
         10  FILLER PIC X(16)
             VALUE "@ABCDEFGHIJKLMNO".
         10  FILLER PIC X(15)
             VALUE "PQRSTUVWXYZ[\]^".
         10  FILLER PIC X(4) VALUE $5F202122.
         10  FILLER PIC X(13)
             VALUE "#$%&'()*+,-./".
         10  FILLER PIC X(16)
             VALUE "0123456789:;<=>?".
     05  ACA-ADDRESS REDEFINES ACA-VALUES
         OCCURS 64 TIMES PIC X(1).
     05  X      PIC 9(2) VALUE ZEROES.
     05  Y      PIC 9(2) VALUE ZEROES.

 01  WS-NAME-ADDRESS-RECORD.
     05  WS-NAR-FIRST-NAME      PIC X(20).
     05  WS-NAR-LAST-NAME       PIC X(15).
     05  WS-NAR-LINE1           PIC X(20).
     05  WS-NAR-LINE2           PIC X(20).
     05  WS-NAR-CITY            PIC X(10).
     05  WS-NAR-STATE           PIC X(02).
     05  WS-NAR-ZIP             PIC X(09).
     05  WS-NAR-PHONE           PIC X(10).
     05  WS-NAR-FLAGS.
         10 WS-NAR-XMAS         PIC X(01).
         10 WS-NAR-PARTY        PIC X(01).
         10 WS-NAR-BUSINESS     PIC X(01).
         10 WS-NAR-BILL         PIC X(01).
         10 FILLER              PIC X(05).
     05  WS-NAR-EOR             PIC X(01).


 01  OV-SWITCH PIC X(1) VALUE "0".
 01  WS-TITLE     PIC X(27)
         VALUE   "NAME/ADDRESS MAINTENANCE ".
 01  WS-RESPONSE PIC X(01).
 01  WS-LAST-NAME PIC X(15).
 01  SAVE-LAST PIC X(15).
 01  SAVE-TEL  PIC X(10).
 01  WS-REC-NO      PIC 1(2) VALUE 0.
 01  EOP-FLAG       PIC 1(2) VALUE 0.
 01  WS-NEW-RECORD.
     05  WS-NEW-FIRST-NAME      PIC X(20).
     05  WS-NEW-LAST-NAME       PIC X(15).
     05  WS-NEW-LINE1           PIC X(20).
     05  WS-NEW-LINE2           PIC X(20).
     05  WS-NEW-CITY            PIC X(10).
     05  WS-NEW-STATE           PIC X(02).
     05  WS-NEW-ZIP             PIC X(09).
     05  WS-NEW-PHONE           PIC X(10).
     05  WS-NEW-PARTY           PIC X(01).
     05  WS-NEW-BUSINESS        PIC X(01).
     05  WS-NEW-BILL            PIC X(01).
     05  WS-NEW-XMAS            PIC X(01).

 IDENTIFICATION DIVISION.
 PROGRAM-ID. LABELS.
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT NAME-FILE ASSIGN TO "1NAMD0001DAT"
         ACCESS IS RANDOM
         FILE STATUS IS NAME-STAT.
     SELECT LABEL-OUT ASSIGN TO PRINTER.
 DATA DIVISION.
 FILE SECTION.
 FD  NAME-FILE.
 01  NAME-REC            PIC X(126).
```

```
FD  LABEL-OUT.
01  LABEL-REC                PIC X(80).
WORKING-STORAGE SECTION.
01  WS-REC-NO   PIC 1(2) VALUE ZEROES.
01  TABLE-INDEX   PIC 9(2) VALUE ZEROES.
01  SUB1   PIC 9(2) VALUE ZEROES.
01  SUB2   PIC 9(2) VALUE ZEROES.
01  NAME-WORK.
    05  NAME-BYTE OCCURS 35 TIMES PIC X(1).
01  LAST-NAME-ARRAY.
    05  LN-BYTE OCCURS 15 TIMES PIC X(01).
01  EOP-FLAG      PIC X(1) VALUE "0".
    COPY "WNAMCOMSTCBL".
01  PRINT-ARRAY.
    05  PA-TABLE OCCURS 2 TIMES.
        10  PA-FIRST-NAME     PIC X(20).
        10  PA-LAST-NAME      PIC X(15).
        10  PA-LINE1          PIC X(20).
        10  PA-LINE2          PIC X(20).
        10  PA-CITY           PIC X(10).
        10  PA-STATE          PIC X(02).
        10  PA-ZIP            PIC X(09).
        10  FILLER            PIC X(30).
01  WS-LABEL-LINE.
    05  FILLER                  PIC X(1) VALUE SPACES.
    05  WS-LBL1.
        10  WS-L1-CITY    PIC X(10).
        10  FILLER        PIC X(01).
        10  WS-L1-STATE   PIC X(02).
        10  FILLER        PIC X(01).
        10  WS-L1-ZIP     PIC X(09).
        10  FILLER        PIC X(11).
    05  FILLER                  PIC X(09) VALUE SPACES.
    05  WS-LBL2.
        10  WS-L2-CITY    PIC X(10).
        10  FILLER        PIC X(01).
        10  WS-L2-STATE   PIC X(02).
        10  FILLER        PIC X(01).
        10  WS-L2-ZIP     PIC X(09).
        10  FILLER        PIC X(11).
PROCEDURE DIVISION.
MAIN-LINE.
    OPEN INPUT NAME-FILE.
    OPEN OUTPUT LABEL-OUT.
    PERFORM A100-PROCESS THRU A100X
      UNTIL EOP-FLAG NOT = "0".
    CLOSE NAME-FILE.
    CLOSE LABEL-OUT.
    STOP RUN.
A100-PROCESS.
    PERFORM A200-READ THRU A200X.
    IF EOP-FLAG NOT = "0"
        GO TO A100X.
    MOVE SPACES TO PRINT-ARRAY.
    MOVE WS-NAME-ADDRESS-RECORD TO PA-TABLE (1).
    PERFORM A200-READ THRU A200X.
    IF EOP-FLAG = "0"
        MOVE WS-NAME-ADDRESS-RECORD TO PA-TABLE (2).
    MOVE 1 TO TABLE-INDEX.
    PERFORM A300-SHRINK-NAME THRU A300X.
    MOVE NAME-WORK TO WS-LBL1.
    MOVE 2 TO TABLE-INDEX.
    PERFORM A300-SHRINK-NAME THRU A300X.
    MOVE NAME-WORK TO WS-LBL2.
    PERFORM A400-WRITE THRU A400X.
    MOVE PA-LINE1 (1) TO WS-LBL1.
    MOVE PA-LINE1 (2) TO WS-LBL2.
    PERFORM A400-WRITE THRU A400X.
    MOVE PA-LINE2 (1) TO WS-LBL1.
    MOVE PA-LINE2 (2) TO WS-LBL2.
    PERFORM A400-WRITE THRU A400X.
    MOVE SPACES TO WS-LABEL-LINE.
    MOVE PA-CITY (1) TO WS-L1-CITY.
    MOVE PA-CITY (2) TO WS-L2-CITY.
    MOVE PA-STATE (1) TO WS-L1-STATE.
    MOVE PA-STATE (2) TO WS-L2-STATE.
    MOVE PA-ZIP (1) TO WS-L1-ZIP.
    MOVE PA-ZIP (2) TO WS-L2-ZIP.
    PERFORM A400-WRITE THRU A400X.
    MOVE SPACES TO WS-LABEL-LINE.
    PERFORM A400-WRITE THRU A400X 2 TIMES.
A100X. EXIT.
A200-READ.
    READ NAME-FILE INTO WS-NAME-ADDRESS-RECORD
        KEY IS WS-REC-NO
        INVALID KEY MOVE "1" TO EOP-FLAG
                    GO TO A200X.
    ADD 1 TO WS-REC-NO.
    IF WS-NAR-XMAS NOT = "Y"
        GO TO A200-READ.
    IF WS-NAR-LINE2 NOT > " "
        GO TO A200-READ.
A200X. EXIT.
A300-SHRINK-NAME.
    MOVE PA-FIRST-NAME (TABLE-INDEX) TO NAME-WORK.
    MOVE 21 TO SUB1.
    PERFORM A310-SCAN THRU A310X
        UNTIL NAME-BYTE (SUB1) NOT = " ".
    MOVE PA-LAST-NAME (TABLE-INDEX) TO LAST-NAME-ARRAY.
    ADD 2 TO SUB1.
    MOVE 1 TO SUB2.
    PERFORM A320-MOVE THRU A320X 15 TIMES.
A300X. EXIT.
A310-SCAN.
    SUBTRACT 1 FROM SUB1.
    IF SUB1 = 1
        MOVE "." TO NAME-BYTE (SUB1).
A310X. EXIT.
A320-MOVE.
    MOVE LN-BYTE (SUB2) TO NAME-BYTE (SUB1).
    ADD 1 TO SUB1.
    ADD 1 TO SUB2.
A320X. EXIT.
A400-WRITE.
    WRITE LABEL-REC FROM WS-LABEL-LINE.
A400X. EXIT.
```

# Bit Bucket

FROM: 007684 85/08 MJ
JOHN SACRISON
P. O. BOX 18
TUMACACORI AZ 85640

Dear Don-

My subscription to '68' Micro Journal has just expired - Please extend it for another year. My check for $24.50 is enclosed. Also, we've just moved into the Big City to take advantage of the shorter commutes and higher client density, so please change my mailing address as well. My mailing label is reproduced above. My new address is:

John Sacrison
P. O. Box 5545
Glendale, AZ 85312

Your magazine, always very interesting and useful, seems to get better and better as I learn more and as my experience grows. As many of us do, I use 6809 systems at home by choice, and MS-DOS machines in most clients' offices out of necessity. After a couple of years' experience with PC clones and nearly five with 68XX machines, I can safely say that as long as you keep supplying the kind of hard information and editorial content you have been up to now, you'll have a lifetime

subscriber. Thank heaven there's someone out there who has the common sense not to ruin an excellent product simply so the Marketing types have something "New" to sell!

A consideration you might want to pass along to your readers who are considering the purchase of one of those "other" machines: The IBM version itself is SLOW! If they absolutely must have the peace of mind that only the massive support provided by IBM can provide, that's a perfectly valid reason to buy one - go ahead. But an unassisted (no co-processor) Big Blue PC or XT seems to run most applications not terribly much faster than a CoCo, and a 2MHz SS-50 system offers a significant speed advantage. I would strongly advise that they purchase instead on of the well-made "clones" that advertise a higher speed capability. At the very least, they should get out and TRY the machine they're considering, preferably with software similar to what they expect to use on it. They might be disappointed!

I enjoyed the August issue very much, especially "The Bit Bucket." It seems that a lot of people have looked around and with some surprise discovered that the demise of the 68XX machines has been greatly exaggerated! I can't help but feel that a stronger marketing push by Motorola could have given things a much-deserved boost in the marketplace. As it is, Tandy has probably spent more on marketing the Color Computers than Motorola has on the entire 68XX Series (feels like it anyway). Now if only the rumored "new" Tandy machine has an SS-50 slot for expansion. . . ! Keep up the good work.

John Sacrison, C.P.L.

# Disk Sector Interleave

Does your FLEX system read files as fast as it should? I've found that a lack of understanding of disk interleave has resulted in many FLEX systems reading disk files at less than half of the potential FLEX read speed. The first half of this article should be of interest to all FLEX users. I will present a simple test that can be completed in 15 minutes that will reveal whether your FLEX system is reading properly. If not, I will suggest how to fix the problem. The second half is for the technical people who are interested in what interleave REALLY does and the pitfalls of misunderstanding what it accomplishes.

### INTERLEAVE TEST

To measure your FLEX read speed we must use a 'standard benchmark' so that we are all comparing the same measurement. The following test has been tried on numerous FLEX systems and has provided consistant results. More important, it has enabled me to speed up some of those systems by a factor of 2-3. Start by making a 100 sector text file on a scratch disk. It won't really matter what is in the file, but for consistency you can use the following TSC BASIC program to create the file for you:

```
10 REM INTERLEAVE TEST
20 OPEN NEW "T.TXT" AS 1
30 FOR I=1 TO 928
40 PRINT #1,"THIS IS A TEST OF THE DISK"
50 NEXT I
60 CLOSE 1
70 END
```

For other BASICs you may have to adjust line 30 so the file is 100 sectors long. Now that the file is prepared, take another disk and format it with NEWDISK, INIT, or what have you. You must make a new, clean disk so the file will start at the first track. Copy your test file T.TXT onto the freshly formatted disk and you are ready to begin the measurements.

We are going to use the LIST command to read the file into a bit bucket. LIST must not take more than a second to be loaded in or it will distort your results. If LIST is near the inside of your system disk, you may want to copy LIST onto another disk for use as a temporary system disk. The command we will use is:

```
LIST T 9999
```

Time how long it takes from pressing RETURN until the FLEX prompt returns. The time measured will be about 5-30 seconds. The 'ideal' time to read a 100 sector file is around 10 seconds for a 1 MHZ CPU and 5 seconds for a 2 MHZ CPU. If you are within 30-40 percent of the ideal speed, you have little to worry about. If you are 200-300 percent of the ideal speed, you may have an interleave problem.

### EXAMPLES OF PROBLEMS

There are many reasons that FLEX systems often 'fail' the above test. A few examples:

1. Five years ago, when I first formatted 8 inch disks with my F&D INIT command, I found they were slow compared to the 5 inch disks I made with TSC NEWDISK. I spent a long time learning about disk interleave and discovered F&D hadn't used ANY interleave on their disks,(i.e. the sectors were stored in numerical order).

2. About 4 years ago my friend Bruno Puglia remarked that my F&D I/O disk controller always seemed faster than his SWTP DMAP-1 controller. One day, when I read a disk formatted by him, we noticed it read as slow on my computer as his. He suddenly realized the problem was the NEWDISK command that came with his SWTP FLEX. SWTP had started selling 2 MHZ computers. Without telling anyone, they modified their NEWDISK interleave for the higher speed. Everyone who buys SWTP FLEX and uses it on a 1 MHZ system will find their computer reads sslloowwllyy!

3. Several well intentioned people (such as William Hart) came up with a "Software Ready Check" to check if a disk is available before accessing it. What they did was use software to check for an index pulse before selecting a drive. This worked for its intended purpose, but created a read speed problem. Unknown to the programmers, FLEX calls the select routine as part of the read command. This defeats all sector interleave since the disk must make a whole revolution to produce an index pulse before each sector is read.

4. Though not really a 'failure' of the test, many FLEX

users have upgraded their CPU to 2 MHZ but didn't upgrade their interleave at the same time. This results in test times of 10-12 seconds which can be almost halved with the correct interleave.

## HOW TO FIX IT

The first step in trying to solve an interleave problem is seeing if someone with a system like yours has already solved the problem! A few calls around to FLEX users may turn up a NEWDISK, INIT, or similar program that has the correct interleave for your system. A call to the supplier of your FLEX may be fruitful, perhaps they read this article too!

A second possibility is you may be able to use the INIT program by Joe Hardo and myself. The program has been expanded to include several SS-30/50 controllers as well as double density. The interleave tables included should be within 50 percent of the 'ideal' speed for your system; adding your own table to the source should bring you even closer to maximum speed for your system. The program is included in the GIVEAWAY package which is free if you send me 4 five inch disks (or 2 eights) with a stamped return disk envelope. If you don't have any method of printing the documentation files, let me know and I'll include the 30-40 pages of photocopies in your envelope.

Finally, if you have to go it alone, I can only give you a general outline of what needs to be done. There are so many versions of NEWDISK there is no way I can tell you exactly what bytes to change.

a. Find the interleave tables. They will show up as hex codes from $01 to the number of sectors on a track and will stand out in a dis-assembly as odd instructions. Usually there will be more than one table to account for different densities, sides, etc.

b. Find the table you are using. It will start with $01 and contain all the sectors you have on a track in what appears to be RANDOM ORDER. For example if you use SSSD five inch drives you have ten sectors per track, the table you are looking for will be 10 bytes long. It may be followed with $00, or the start of another table.

c. Replace the old table using a table derived from reading the rest of this article.

d. Repeat the entire speed test. If the speed is the same, you may have altered the wrong table. If the program crashes, be sure you didn't alter the length of the table (perhaps you overwrote the null at the end).

## WHY INTERLEAVE IS NEEDED

Interleave, as it applies to FLEX, is the order that the sectors are recorded on the disk. The TSC manual mentions this, but doesn't discuss it in detail. I've found few people really understand the operation of how FLEX reads a sector, as a result many systems are not interleaved correctly. I think, by using some actual measurements, I'll be able to take some of the mystery out of disk formatting.

The reason we need interleave is that FLEX can't process data as fast as the disk drive is capable of reading it. I'll start by showing what happens without interleave. Let's use a ten sector disk for our examples. The sectors would be ordered:

```
V--------V
1 2 3 4 5 6 7 8 9 A
```

When FLEX finished processing sector 1, the disk would have rotated almost 2/5 a revolution. When we are ready to read sector 2, sector 5 would be passing over the head. As a result we would have to wait as the disk makes another 3/5 revolution to bring sector 2 under the head. We have just discovered our first rule of interleave: If you have the wrong sector order the disk will be read at approximately 1 sector per revolution. To put that in perspective a five inch drive rotates in 200 msec so a 100 sector file will take 20 seconds to read (and flunk the above test!).

Why does FLEX take 2/5 revolution to process a sector in the above example? FLEX has an elegant File Manager which does a lot of work when processing a sector. When writing a utility, you may use a read routine similar to this:

```
LOOP    LDX #FCB
        JSR FMS       CALL FILE MANAGER
        BNE ERROR
        LDX POINTR
        STAA 0,X      STORE CHARACTER
        INX
        STX POINTR    UPDATE POINTER
        CPX MEMEND    OVERFLOW?
        BNE LOOP
```

This looks like a fast loop with only nine instructions. Unfortunately, the subroutine FMS is about 45 instructions long. The FMS must save a pointer, check the function byte, check the activity status, check for TAB expansion, read a character from the FCB, bump a counter, check for three control characters, set some status, restore a pointer, and return. All this must be done 252 times per sector! You can see why the software overhead for processing a sector is about 60 milliseconds.

You may have noticed I haven't discussed hardware so far! The 60 milliseconds of CPU time is independent of the of size of the disk, density, sides, etc. The slowest disks are 5 inch SSSD which as mentioned above read 10 sectors at 5 rps (20 msec per sector). The fastest popular disks are 5 inch winchesters which read 32 sectors at 60 rps (.5 msec per sector). Our next rule of interleave: The disks are always faster than the FLEX File Manager. A winchester disk SHOULD only be slightly faster (60+.5) than a floppy (60+20) when reading a file. Those who say their winchesters are 3-4 times faster than their floppies (David Graves 68 MJ Oct 82) should spend some time fixing the interleave on their floppies! Note that the winchester rotates in 16.6 MSEC, so the drive makes a couple of revolutions while FLEX processes a sector.

## CALCULATING INTERLEAVE

The method FLEX uses to implement interleave is very flexible. Unlike some systems that compute interleave when reading (CP/M), FLEX computes interleave when formatting. FLEX computers can read disks regardless of the interleave, slowly perhaps, but successfully. This is why many FLEX users do not realize they are using the wrong interleave.

The sectors on a FLEX disk are written in non-sequential order. The TSC NEWDISK program I used in 1980 formatted SSSD disks with the following order:

```
V--------V
1 3 5 7 9 2 4 6 8 A
```

While processing sector 1 in 60 msec of CPU time the disk will advance to sector 9. We will be ready to read sector 2 when it passes under the head. If you follow around the table, you will see we can read two sectors per revolution (one odd number and one even number). This disk will read twice as fast as a disk formatted in sequential order. The above table is said to have an interleave of five, since sector 2 is five sectors after sector 1 (sequential order is called interleave of one). Another rule: Interleave number is the minimum count from any sector to the next number sector. Sharp eyed readers may notice the above table has a count of six from sector 2 to sector 3. Sometimes it is impossible to achieve the desired interleave for every sector.

Looking back, we noticed that when the system was ready for sector 2 the head was still one sector away. Let's make a new table with in interleave of four which may read faster. I used to make up tables with pencil and paper using the following method

```
1 2 3 4 5 6 7 8 9 A       make ten columns
1                         first sector is 1
1       2                 count over 4
1       2       3         count over 4
1   4   2       3         count around 4
1   4   2   5   3         count over 4
1 6 4   2   5   3         oops, around 5
1 6 4 9 2 7 5 A 3 8       finish counting 4
```

As you can see, when we wanted to place sector 6 the desired cell was filled. The only choice is to insert an additional count. Some interleaves come out 'perfect' such as three:

1 8 5 2 9 6 3 A 7 4

Double sided double density tables are not difficult to create. They can be thought of as two parts; the first side looks just like what we've seen, and the second side is the same table over again with the side size added to it. The following table is for DSDD with 18 sectors per side and an interleave factor of four

```
01 0A 06 0F 02 08 07 10 03    first side
0C 08 11 04 0D 09 12 05 0E    first side
13 1C 18 21 14 1D 19 22 15    second side
1E 1A 23 16 1F 18 24 17 20    second side
```

I've found the best way to find the ideal interleave is by trial and error. There are so many variables to consider that computation only puts you in the ball park! Some of the complications are:

1. Placement of the last sector. An interleave table that looks 'perfect' may not allow enough time between the last sector and sector 1 of the next track. This will waste a revolution for each track step. This could be solved with track to track skew, but I haven't seen skew in any FLEX formatting programs.

2. Headload delay. Some drivers (such as the original TSC) set the head load bit on the read command. The controller then plays dead for 10 msec before reading, perhaps long enough to miss a sector.

3. Application software delays. Some programs will have more processing to do between sectors, causing the disk to make a whole revolution per sector. I've found that LIST is a good test, if LIST works then all buffered programs will work. It is conceivable to select an interleave to match the application. HP computer manuals suggest different interleave for boot disks, program disks, data disks, etc.

4. Sector format variations. The gap characters between each sector vary depending on density and the whims of the person who wrote the formatter. You can squeeze extra sectors on a track by scrimping on the gap sizes. There are also variations in how long a controller takes to lock onto the sector header.

5. CPU clock speed. This is the main factor that controls the fastest interleave that will work. A 2 MHZ CPU will accept an interleave number of slightly over half that of a 1 MHZ CPU.

## HERE'S THE BEEF

Enough theory, here are some actual speed tests. The tests were run on various interleaves using a 6809 at 1 and 2 MHZ. The drives used in the tests were

four sided 5 inch 180 tracks (SA-604)
single sided 8 inch 77 tracks (SA-800)
double sided 5 inch 80 tracks (TM 100-4)
single sided 5 inch 35 tracks (SA-400)

| DRIVE | CPU MHZ | INTER-LEAVE | DENSITY | SIDES | SEC |
|---|---|---|---|---|---|
| RAMDISK | 2 | N/A | N/A | N/A | 6 * |
| WINCHESTER | 2 | 65 | N/A | 4 | 5 * |
| 8 INCH SS | 2 | 5 | 1 | 1 | 6 * |
| 8 INCH SS | 2 | 6 | 1 | 1 | 7 |
| 8 INCH SS | 2 | 9 | 1 | 1 | 11 |
| 8 INCH SS | 1 | 9 | 1 | 1 | 11 |
| 5 INCH DS | 2 | 4 | 2 | 2 | 6 |
| 5 INCH DS | 2 | 3 | 1 | 2 | 6 * |
| 5 INCH DS | 2 | 4 | 1 | 2 | 8 @ |
| 5 INCH DS | 2 | 4 | 1 | 1 | 9 @ |
| 5 INCH DS | 2 | 3 | 2 | 2 | 12 |
| 5 INCH DS | 2 | 5 | 1 | 2 | 12 |
| 5 INCH DS | 1 | 5 | 1 | 2 | 12 |
| 5 INCH SS | 2 | 3 | 1 | 1 | 8 # |
| 5 INCH SS | 2 | 4 | 1 | 1 | 10 |
| 5 INCH SS | 2 | 5 | 1 | 1 | 12 # |
| 5 INCH SS | 1 | 4 | 1 | 1 | 22 # |
| 5 INCH SS | 1 | 5 | 1 | 1 | 12 # |

* Note that the speed of the following drives were within 20% of each other::

1. RAMDISK written in flexable 6800 code.
2. Winchester using formatter and driver by Dave Graves and Robert Zeff.
3. Old 8 inch single density single sided.
4. New 5 inch single density double sided.

@ Note that double sided will sometimes be faster due to fewer track to track steps.

# Note that using the proper interleave resulted in 8 seconds for 2 MHZ and 12 seconds for 1 MHZ. Using the wrong interleave resulted in 12 and 22 seconds.

## RELATED SUBJECTS

So far we have discussed the effect of interleave on FLEX read speed; what about its effect on write speed? Actually there is very little change in write speed no matter how you format a disk. FLEX must make three passes over each sector that it's writing! First FLEX reads the sector to find the link to the next available sector. Second FLEX writes the new sector with the old link in the first two bytes. Third FLEX does an optional VERIFY of the sector. This requires two complete revolutions plus a partial revolution to reach the next sector. This can be reduced 40% by turning off VERIFY. You can see that the rotational speed controls the write speed which is why winchesters write much faster than floppies.

Seek speed is another factor that is hardware dependent. The seek speed is selected by software, but must be no faster than the drive will allow. Five inch drives vary over a wide range, from three to forty msec per step. The seek time is usually small compared to the read or write time, but gets a lot of attention since it is audible! The drive sounds faster, therefore everyone assumes it is faster. Be aware that there isn't any feedback from the drive as to when it has completed a step. Two of my friends insisted that the 80 track drives they purchased were 'MUCH FASTER' than the 40 track drives they were used to! Only after I had them run my speed test did they realize it was all in the ears! Connecting a drive that is capable of a shorter seek time does nothing unless the software is changed to step faster. FLEX random files require a lot of seeking, thus will be faster on a drive with a short seek time.

## CONCLUSION

I hope I've provided some insight into the subject of interleave and how it affects FLEX speed. I've said quite a bit, there is still more information I've left out! I welcome any questions on this or any other topic involving FLEX.

Leo Taylor
109 Twin Brook Road
Hamden, Conn. 06514

# Rambling Around NCC

**Ramblings & Such**
**ABOUT TIME!**

Well, here it is about 2 days since the close of NCC Chicago 1985. Larry is back from the show and I am delighted with the reception OUR folks stacking up. After all, for the past few years the Fruit and 88's have been nipping us pretty hard. But at the "crown jewel" of computer shows, our folks came out on top! GIMIX, Microware and Smoke Signal Broadcasting stood toe to toe with the biggies, and came out winners!

Yes, you read it right, Winners. Spelled with a capital W. And it all revolved about the Motorola 68000 family of microcomputer devices.

When I and a lot of you first started out in this thing, you remember, in those pre-mostateverything days, as awed as we were, at least I, never envisioned the power of computing that was to be in just a few short years. But it is here in the 68000 series, now. And we have not even really begin to scratch the surface.

The time span has actually been so short that I can clearly remember the giant steps as they fell, each louder than the previous. When I first began to program and build with that lovely 6800 there was not one, no not one editor (yes, even before the excellent TSC Editor) that was available for it. Then SWTPC made available on tape (paper first, later magnetic) an editor (of Motorola origin I believe) and an assembler to put the whole mess together. Before that we punched it in hex by hand, byte by byte. Which was better than the others, as most of them had to punch it in by hand, in octal (ugh!!), bit by bit. Try that some time you non-lovers of HLLs, a group I have since gladly joined.

The editor was really great. To use it you didn't have to worry about looking for your errors on the CRT screen. Most didn't have a CRT screen. Some had (the lucky ones) the clanking Model 33 Teletype, and a paper screen. Others used all sorts of devices, from the new and quite modern modified TV typewriter (SWTPC, slower than a snail in cement, but cheap to build and sure saved on paper) to various military surplus ASCII teletype machines, and other strange and maddening devices. But if you did happen to have some sort of printout device, it didn't matter, the editor was neither line or screen orientated. Fact is wae wasn't anything orientated. That is except your own grey matter. For everything was kept in your head, and a hidden RAM buffer, until you called for a printout. And using paper at 110 baud or less, printouts were not called for often.

Here is roughly how it worked. You typed (I am going to get a lot of letters on this because I do not have the documentation any longer and am going on a foggy memory) a character, I believe 'T' to get to the top of the page (buffer). Then you started to input your text (mostly machine hex source code - you wrote most of your own stuff) inputting what you hoped was largely error free. Mine never was, but I heard tales, which I took with both awe and a grain of salt, of those who used the thing and never made mistakes. When they called for a printout, they had to make no corrections. I still classify those stories with the tooth fairy.

Now for most of us mortal programmers, it was not quite the same, we made mistakes. And we really paid dearly for those lapses of mental and finger dexterity. You see, when you reentered the editor, on tape, then your text file, on tape, the coffee was not only done that you put on when the tape first started running (first time - remember, before it got down to the last few bytes and then bombed) but it was all drunkup by the time you had your program or data finally loaded without a jump-out to the monitor, which dinged the bell or something and then announced in a matter of fact way that something had gone belly-up (hardly ever the same thing twice in a row) and you started all over from scratch. That is after you rewound the tape. Now if you were real lucky it was only a couple of more tries with the tape to get the source or text loaded. For a program of some 40 or 50 lines (a biggie in those days as most had only a 4K machine) this whole process could take an hour or so, if you were lucky. I have seen whole nights pass by and the lucky combination was still eluding.

Finally you were back in business (praying, "Please Lord, please don't let the lights flicker!!). Once running again you then input the 'T', or something, and the editor was then pointing at the first character, on the first line of whatever you had loaded. Having looked at you printout you see that in the third word over, which was say 'the', you had input it 'tht' or some other nonsense. Now the 't' that should have been an 'e' was, by counting over from the first character, the 12th character (remember spaces are characters also). So you input "^" or ">" or "L" (all upper case, only the very rich had lower) and "12". This placed the pointer to the offending character, you hoped. Then you input an escape sequence of some sort and then input the proper character. Now there was another error a few lines down and so many characters in. So you input "D" and the number of lines down (delimited by the $0D - C/R). The "D" meant 'down' I believe and of course the now familiar over so many characters over and the escape sequence again. Boy, you talk about fun! Maybe the "^" and "-" sent you up and down line numbers, I can't remember, and really don't want to. There were many other 'hidden' editing features, none of which I desire to recall any longer. The whole point being, it was a marvel at the time, but looking back now it was for the pits.

Then came the TSC Editor! To me that was a milestone of progress exceeding the atomic bomb. Also available from TSC at about the same time was their assembler. Each was on tape, or you could order just the printed source listing and enter it in in hex, by hand. I did. And it was essentially the same as what you buy today. Only today you pay a whole heap more and do not get that beautiful commented source listing. I wonder how many 68XX programmers cut their programming teeth by studying those wonderful source listings. I did. I can tell who some were when I receive an assembler source commented, as soon as I see: "INX BUMP THE POINTER" I know where you came from. Thanks TSC. That was one giant step forward.

Then later SWTPC (who supported TSC software completely at the time) offered a version that loaded both the editor and assembler from one tape run. This meant that you could bounce from the editor, once you were satisfied with your source code, and go immediately to the assembler, without waiting, and waiting and ..... Then if (and normally when the assembler bombed your error(s)) you could jump right back into the editor and make the correction(s) (hopefully). It was called CoRes and was a jewel. Sold for the princely sum of $8.00, if I remember correctly. I believe it was the SWTPC idea to attempt to make things affordable for the beginner, that had a lot to do with our survival. Gosh, what happened?

Milestones came faster and faster. The twin cassette tape deck from SWTPC, the fast (1200 baud) cassette from PERCOM and an even better tape system from JPC Products. It was raining manna, and I was going nuts trying to keep up with progress. Then a super milestone rolled in - floppy disk. Mine was the first one shipped by SWTPC. It had the original drive in it that had been the prototype. I uncrated it actually shaking, for I knew that at last I had the REAL THING! What had taken twenty minutes before (loading BASIC) would now be reduced to seconds (actually about 20 seconds). Nothing, I mean nothing could ever

exceed that. 340 sectors, 35 tracks on one side was like a boxcar load of tapes.

I fired it up, and while sitting in awe running "DIR" over and over again, the lights flickered. BLAM, it would not boot. I called SWTPC and Dan Meyer put a 'rush' second disk on the way, but Mickey (I had called, he had gotten out of bed and was on the way over when it died) and I decided to try to take the code apart and see what we had. Within a couple of hours he had figured out the 'boot' function and we wrote one to tape. It was murdered while the head was sitting back on the 'home' position, right over the boot track. I learned about leaving disk doors closed, with that one. For the next couple days we booted it with a short 75 or so byte program on tape and then enjoyed the luxury of disk computing. To me the disk system was the second great milestone, right after the 6800. So we had major and minor milestones, each one a marvel and seemingly unsurmountable, forever.

I guess by this time you are wondering what all this stuff has to do with NCC '85 and our group. Well, we have come to another milestone. The Motorola 68020. But first let me tell you a little about what happened at NCC, as I am told by some others who were actually there. I stayed home this year minding the store.

First, to keep out of hot water I am going to go alpha, so GIMIX comes first. GIMIX was showing their new and POWERFUL 68020 megabyte, mega-plus speed system, running both OS-9 68K (I told you so, even when they were denying it) and UniFLEX VM. Now it has only been completed and running a few weeks, but even at this stage it is about 30 to 200 times faster than a 6809 doing the same function. While the average 68008 is somewhat slower than many 6809 functions, the 020 version is a speeding bullet. It is so fast that sometimes you think it didn't do anything. With both OS-9 and UniFLEX you gets your choice. And like I said a few months back; a level one 68K ain't exactly the same as a level one 6809, be aware. The offshoot of all this power being at NCC was that the GIMIX booth was one of, if not the most active booth at the show, so I am told. It was told me that some 'big wheels' from some of the real big 'hot shot' computer makers, stood around, jaws drooped, wondering where this GIMIX outfit had popped up from. I do not believe, from what I hear, that there is another 68(anything) that can stand toe-to-toe with the GIMIX 020 system. It has it all! And now the same thing in an economy single board computer. Wow! For you smart ones who have held off purchasing a 68XXX single board computer, your patience is soon (reel soon) to be rewarded. **The GIMIX board is so far advanced over the other 68XXX single board computers (todate), there is no comparison.** Let the turtles beware!

Not to be outshone, Microware was displaying their OS-9 systems and other hardware and Microware software. There was also an active and busy booth.

An impressive demonstration at the Microware booth was the Microboards Model 32, running OS-9 68K, from one of Japans larger electronics manufacturers. This system is a shared resource system, running con-current Unix Sys 5 and OS-9 68K. It's primarily a development system. The hard disk and other system resources are shared by both. A Mizar system was also in the Microware booth but we never saw it fired up. Although I was told it was demonstrated occasionally. Also there were so many shared CRTs that is was difficult to determine what was hooked to what.

Also running very impressive color graphics was the new system from Microware and Hitachi (see press release, Bit Bucket). The Software was a joint effort of Graphics Software Systems (GSS) and Microware. Some industry experts say that this product design is fast become the industry standard for international high-level graphics. This alone attest to the maturity of Microware products.

The acceptance that many quality hardware manufacturers have given to OS-9, is a tribute to the respect and attention Microware has devoted to its products, customers and users, small and large, over the years. I have watched, from an advantageous view point, all those years. I believe one thing that has been a difference between success and failures in our community group has been not forgetting our beginnings. Those little 'hobby' types that some sneered at yesterday, are the guys who are now buying the big bucks software and hardware that has switched the fortunes of some of our folks. Some of the others never could get it into their business heads that 'hobby types' develope into 'big bucks types'. I know hundreds! Microware has apparently not forgotten it's beginnings (RT-68, etc) and today is a major force in the quality computer world. It sure is nice to know 'world class' folks.

Also I want to compliment Microware on the quality of their brochures, spec sheets and other handouts at the show. They say quality plus all the way. Even their all Japanese stuff was 'print art'. I am impressed, so much so that I guess I will have to start saying Mr. Kaplan. Seriously, they were quite eye catching, even more so when you read the long list of their worldwide clients. A **good reputation** is to be envied.

Smoke Signal Broadcasting (SSB) was well represented at the '85 NCC. They were proudly showing their VAR/68K systems to those who were looking for a 'close' UNIX like system. For the price the SSB system is hard to beat. And with either OS-9 or REGULUS on the SSB hardware, you have a choice of both worlds. Needless to say, but SSB is as has been their practice in the past - "giving us a choice". Me, I like that. For those wanting a real close look-a-like for UNIX, they seem to have it, in two delightful flavors. SSB is another one of those that has kept an ear cocked towards their users.

So there you have it, a short report on our folks at the premiere computer show - '85 NCC. I am proud of the fine showing by them. They have proven the old adage about mousetraps and such. They hung in there, did their thing, didn't forget where they came from, and it is all starting to pay off. Loyalty is a two way street. Too many knowledgeable folks have already told me so. In 85 **we took top billing!**

So, that brings me to milestone number something or another. But it is a biggie. The 68000, and especially the 68020. If ever Motorola did anything really right, this is it.

Over the years I have not been too kind with Motorola. But I called it as I saw it, and at times I became so vexed I felt it unfair to you, my readers, to hold back. As a result I guess you could say that we never did really make it, Motorola and I. That has surprised some folks. I have had officers and staff of other magazines express to me amazement that I continued to devote my entire effort to Motorola products. This while some of our competitors (start-up 68XX magazines) were having a major portion of their articles written by Motorola employees. It made no difference to me. I felt it better that I had no obligation to Motorola. Fact is, we never even asked Motorola to advertise in our pages, despite our total support for devices devoted to their products. We never received any inducement from Motorola, in any form, to do what we have done (we receive device spec sheets, and PR releases only). We did not and do not owe them anything. Nor they us. We do it because you, our readers and supporters, saw fit to use their products. So, we were and are keeping our pledge, from day one, supporting you. However, that is not to say that we would not like some of their advertising bucks, but it would not change our editorial policy one bit (no pun intended). I still am going to squeal loud if I think they, or anyone else, lets us down, or stick it to us. I didn't like the flakey 6809 bit a few years back and I do not think they supported the 6800 and 6809 as well as they could or should have. I don't believe that the Z80 and 6502 should have made it to where they did! But then maybe I liked them (68XX) better than Motorola did.

But now I gotta hand it to them. They have really done it right with the more recent 68XXX offerings. Especially the 68010 and 68020 CPU devices. I have no doubt but what these are the finest microcomputers (micro?) available, anywhere, from anyone! **You are to be congratulated Motorola, you guys put it all together on this series, and we thank you!**

Anyway, to you Motorola for making it, and to you GIMIX, SSB, Microware and all you others using it, we say **CONGRATULATIONS!** You are all winners.

DMW

- - -

SAINT JOSEPH'S COLLEGE

COMPUTER CENTER
P.O. BOX 909
RENSSELAER, IN 47978

Mr. Larry Williams
68 Micro Journal
5900 Cassandra Smith Road
Hixson, TN 37343

Dear Mr. Williams:

Thought you might be interested in how we're using our GMX III microcomputer here at St. Joseph's College.

We are now in our third year using this machine for a variety of purposes within the Computer Science and Business curricula at the College. Our system currently supports up to 14 users, with 9-12 terminals being connected to the system at any one time. We have 384K of RAM in the system, and utilize the 20MB Winchester disk.

Our OS-9 Gimix is used as the only machine for teaching Assembly Language. It is also the primary machine studied in our Operating Systems class.

Additionally, because of the ease of interfacing the machine to other RS-232 machines, we use the Gimix for coordinating all data transfers between the various machines on campus (including two PRIME minicomputers--a model 2250 and a model 550 II) and outside computers and networks that we subscribe to. I might add that we have so far been unable to effect any networking control with our PRIMEs, since we chose not to install their networking software at a cost of over $10,000. The Gimix is invaluable to us in that respect.

We have lately added another word processing software package to the machine, and our campus Computer Use Committee authorized a policy of "heavy recommendation" that our students and faculty members use the Gimix exclusively for their word processing needs. We also have it networked to a Compugraphic electronic typesetter at a local printer, and we use it for printing a number of college publications--going directly from OS-9 text files to camera-ready film without any human intervention.

Just this summer, we have begun to use the Gimix heavily for programmers choosing to use the C programming language, chiefly because of the similarities between OS-9 C and UNIX C compilers.

Finally, our Business department will be using the Gimix, running the Dynacalc spreadsheet program, in its introductory "Computer Use in Business" classes, that will be required for a large number of our Commerce majors. The teachers using this package have had nothing but favorable comments on Dynacalc, and one of them has said she is considering using Dynacalc to replace a lab class she now teaches using SPSS on the PRIME.

In short, the Gimix has proven to be very heavily used here, so much so that the Computer Use Committee has recommended to the Administration that we purchase another identical machine, to be used solely for word processing by both students and faculty. It should also be mentioned that we have yet to experience even a minute of down time of any sort on this machine. Also, during repeated dips in power supply voltage (due to our reliance on rural electric supply) we have had several problems with our PRIME machines "going down" due to the dips. The Gimix sails along unscathed, and by its reliability has proven to be somewhat of an embarrassment for the PRIME sales personnel.

If you need more information about any of these applications, feel free to give me a call.

Sincerely

Bruce Mathey

Bruce Mathey
Assistant Director of Computer Services

Gentlemen, ***

I presently have a Radio Shack 64K Color Computer with two Teac double sided disk drives. I am interested in upgrading to a motorola 68000 based machine. I have looked at the Apple Macintosh but am not satified with it.

Plase forward any information that you can provide on hobby computers using Motorola 68000, 68008, 68010, 68020 etc. microprocessors. In studying your publication I found address for SWTPC, AAA chicago Computer Center and Acorn Computer Center. I also saw an advertisement for GIMIX, but their advertised prices are beyond my budget. I am hoping to find a machine that will allow me to utilize my existing CRT and disk drives. I would not be opposed to a kit or partially assembled machine, should there be such kits available.

My address is as follows;

Sincerely.

Larry D. Dreadon WA4GOT
1212 Clearview Drive
Mt. Juliet, TN 37122

PRESS RELEASE

FOR IMMEDIATE RELEASE

FOR MORE INFORMATION CONTACT:

| | | |
|---|---|---|
| Harry McGrath | (Hitachi) | 408/992-1500 |
| Ken Kaplan | (Microware) | 515/224-1929 |
| Dan Pinsberg | (OSS) | 503/682-1606 |

**HITACHI ANNOUNCES COMPLETE THIRD PARTY SOFTWARE SUPPORT FOR HIGH-PERFORMANCE ADVANCED CRT CONTROLLER DEVICE**

CHICAGO, July 15, 1985 -- The Semiconductor and IC Division of Hitachi America, Ltd. today announced that Microware Systems Corp. and Graphic Software Systems (OSS), two leading software houses, have jointly developed complete graphics software support for Hitachi's Advanced Graphics and CRT Controller (ACRTC) device, the HD63484.

Microware has customized a special version of its OS-9 real-time, multitasking, UNIX-like 68000-based operating system for the ACRTC and offers a C compiler for application-program development. OSS and Microware are cooperating to develop an ACRTC-optimized version of OSS' highly regarded OSS-DMY9000 product, which provides a powerful, device-independent method for creating portable graphics-oriented application software. OSS-DMY9000 conforms to the proposed ANSI/ISO Virtual Device Interface (VDI) standard. Microware is porting the OSS-developed package to the OS-9/68000 operating system. Later this year, OSS and Microware will also offer a Graphics Kernel System (GKS) option, which is quickly gaining prominence as the international high-level graphics standard.

"For the first time a complex peripheral such as the ACRTC has been completely supported with industry standard software tools," said Tony Moreyra, Hitachi's strategic marketing manager for microprocessors and CMOS gate arrays. "The operating system, programming languages and graphics driver support for the ACRTC provide a comprehensive software package and development environment for advanced graphics-oriented applications."

The OS-9/68000 operating system is rapidly gaining popularity in industrial automation, personal computer, telecommunications and related markets. The new version's software drivers take full advantage of the ACRTC's high-speed intelligent graphics drawing functions. Graphics images can be combined with text data in multiple fonts. Software for the ACRTC can be developed using Microware's C compiler running on either a 68000-based development system or a large DEC VAX timesharing system. BASIC, Pascal and assembler programming languages with graphics capabilities are also available.

Kenneth Kaplan, president of Microware, said "We have achieved an incredible level of integration of all the hardware and software pieces an OEM needs to build a state-of-the-art graphics system. Hitachi and Microware can now offer a total solution to manufacturers of personal computers, CAD/CAM systems, graphics workstations, and similar products."

According to Moreyra, "Because so many graphics and CRT controller functions are implemented in hardware, the new generation HD63484 elegantly handles colors and graphics data processes 10 to 50 times faster than competitive devices."

The HD63484, introduced last June, employs Hitachi's 2-micron all-CMOS technology to integrate CRT and graphics functions on a single chip. The bit-mapped controller has three on-board processors to relieve the system's CPU of many time-consuming supervisory tasks. The chip operates with a 10-Mhz clock, draws at a 400 nanosecond/dot rate and can create images of up to 4096 by 4096 pixels. It can produce dots, lines, rectangles, polylines, polygons, circles and ellipses, and can paint, fill and copy objects in 65,536 different colors.

"The resolution and performance gains from chips like Hitachi's ACRTC will spur a dramatic increase in graphics application development" said OSS President Tom Clarkson. "Our ANSI/ISO-based VDI software cuts development time and greatly expands the market through compatibility with more than 65 peripheral devices while it maintains DOS to UNIX portability. These are some of the reasons why it was selected by IBM and AT&T as the de-facto standard graphics development environment for PCs."

The software agreement calls for Microware to bundle OSM licensing of all modules of the ACRTC software system including the OSS-DMY9000 program. Both Hitachi and Microware have special procedures in place to give ACRTC users coordinated technical and marketing support.

Microware is a Des Moines, Iowa-based software house specializing in 68000-based programming languages and operating systems. OSS, a Portland, Oregon-based software firm, is a leading supplier of industry standard graphics software packages.

The Hitachi Semiconductor and IC Division handles the marketing, sales and service activities for Hitachi America, Ltd., a U.S. corporation, which is a wholly owned subsidiary of Hitachi, Ltd. of Japan. Shares of Hitachi, Ltd. are traded as American Depository Receipts on the New York Stock Exchange. The division sells and services microprocessor, MOS memory, opto-electronic, bipolar and other integrated circuits to OEMs and distributors.

The company is located at 2210 O'Toole Ave., San Jose, Calif., 95131. Telephone 408/992-1500.

***

# An Open Letter from the OS-9 Users Group

Dear Don,

Please accept our sincere apologizes for letting your application for membership in the OS-9 Users Group slip through the cracks. As soon as I read your note in your August issue, I sent membership committee chairman Joe Dubuc a note and asked him to enter 68 Micro Journal into the OS-9 Users Group data base immediately. I also asked him to send you Users Group Disk Number Zero and a copy of each back issue of MOTD he has on hand. MOTD is the group's bi-monthly newsletter. I am sorry for any inconvenience we have ca ed.

I hope you can find it in your heart to publish the following kudos to a small cadre of hard workers who have helped this group grow from just more than 50 members in August of 1983 to nearly 800 today. The kudos will take the form of a listing of the groups accomplishments during this same period.

First, a tip of our hat goes to Tim Grovac and his wife Lori in Kent, Washington. They both work hard to publish MOTD every other month. Tim and Lori stepped into publish the newsletter after Dick Dunbon passed away last December. The last issue contained more than 20 pages of solid information. Several have been longer. Our hats are also off to the regular contributors that make Tim's job easier. Greg Morse, who writes an information packed column called BASIC09 Corner; Jim Schmidt, who's CoCo Advocate column is both informational and fun, Bert Schneider, who writes Users Group Software Commentary, a regular series of reviews covering software contributed to our exchange library, Dave Kaleita, who updates readers about the latest library contributions and offerings and George Dorner who has contibuted several interesting articles in addition to his regular treasurer's reports. These men form the heart of MOTD, but more and more members are contributing ideas and short notes every issue. Thanks to all.

Our next round of Kudos goes to Dave Kaleita, Chairman of our Software Exchange Committee. Dave almost single-handedly organized the group's software holdings into a coherent set of tools that any member can order and use. We must also thank the several dozen members who contributed the 200 plus programs

that Dave has organized on 35 disks. The list reads like the Who's Who of the 6809 world, including Hal Snyder, Carl Kreider, Eric Williams, Peter Dibble, Bryan Capouch, Greg Morse, Peter Lyall -- to name just a few. But more importantly, these disks can't be used by anyone until they are duplicated and distributed to members. Kudos for this hard work go to Frank and Carol at FHL who have shipped nearly 1,000 disks for us since the first of the year.

So what else has the OS-9 Users Group done for its members? Thanks to the imagination and efforts of our treasurer, George Dorner, we hosted several hundred OS-9 users at a special hospitality room during Rainbowfest Chicago. Since it was such a great success, we hope to do the same thing in Princeton this October. By the way, we plan to hold a meeting of the Users Group at Princeton following the last scheduled seminar Saturday.

Don, we realize that the group has had its problems. However, most of them have been logistic in nature and we are working hard to fix them. For example, the most serious problem has been with the first link -- from our mailbox in Des Moines to George Dorner who sorts the mail and distributes it to the proper committee for action. We are recruiting now, hoping to nominate and elect a new secretary -- from Des Moines -- who can take charge of the mail problem. But we do not plan to make him do it on his own. We are preparing to hire someone there to do the actual work. This should smooth up the operation tremendously.

We realize also that we need to find a way to handle simple technical questions from people. We are leaning toward appointing a committee to do this. But, please advise your readers to give us their phone number when they write with a question. Since we don't have a professional staff, it is almost impossible to mail individual answers. However, when we have a number, we often call the person with an answer -- when we know it.

Thanks to Joe Dubuc, Robert Ringrose and James Petty of our membership committee in Oklahoma City, administrative functions have ran pretty smoothly for the past year. Joe has installed an impressive data base that has helped us provide better service to everyone.

Perhaps a bit of history is appropriate here. Brian Capouch spearheaded efforts to get a group going at the Microware Seminar in 1982. However, there was a lot of bickering among the other officers that first year and the movement never really picked up steam. Peter Dibble and I observed the attitude -- and the total lack of volunteers to serve as officers -- at the Microware Seminar in 1983. We chatted a few moments and decided we should stand up and go for it. Needless to say, we were elected immediately. Taking our lead, Tom Murphy of Suntel Systems in St. Louis and George Dorner of Harper College stepped up to serve as Secretary and Treasurer respectively. When Tom had to resign to take care of a growing business, Dave Gibson, a programmer with the Department of Transportation in Washington, D.C. stepped forward to the challenge.

Thanks to Tom's business management skills, George's tenancity, and the access to the media enjoyed by Peter and I, we got the ball rolling. The accomplishment of the goals we set is the result of the hard work of the people named above and many others. But, nothing would have happened if someone hadn't stood up and accepted the challenge. We're glad we did it.

When we volunteered, Peter and I agreed to give it a try for a year. That year came and went and until recently no one has stepped up to take our place -- or even nominate replacements. Yet, I bring you good news. We now have a new slate running for office and we hope to pass the torch at Microware's Seminar in November. Bryan Lentz, who has written a lot of OS-9 software for Computerware and FHL is running for President. He hopes to have Bill Taylor, who works for GTE Sprint and has written for Interface Age, on his slate. And, George Dorner has promised to run for treasurer again if we promise to relieve him of some of the collateral duties that he has handled so well during the past two years.

We welcome additional volunteers. In fact, we would be proud to have someone from 68 Micro Journal serve as an officer of the OS-9 Users Group. We will be publishing information about the election in the August issue of MOTD and sending ballots along with stamped return envelopes to all members in September. The deadline for return will be October 15. New officers will be installed at the Microware Seminar in Des Moines Nov. 2.

Sincerely,

Dale

Dale L. Puckett
President, OS-9 Users Group

Editor's Note: Thanks Dale for your reply to my August '85 notes. I have certainly gotten the groups attention this time. I hope something constructive comes of it. The number of replies received so far exceed about all previous correspondence from readers concerning the group.

First off let me say that I am not personally upset, neither is any of the staff of Computer Publishing. It is simply that OS-9 is a fine system and most of the users, we know, are dedicated hackers or professionals. The users who have hung-on during the Big Blue onslaught deserve all the help and support they can get. Mutual help and assistance should be the 'battle cry' of any users group. I am delighted to learn that the OS-9 group is doing so well! As with most organizations, the main ingredients normally are a sense of direction, mutual understanding and dedicated leadership. From your letter I feel that things are improved.

As you might guess, we also are OS-9 users, having some three or four OS-9 systems, ranging from a GIMIX III with a megabyte of RAM to several level one and two systems, including the UniBoard, Sardis, PT-69, CoCo and even a SWTPC. Also over the years we have accumulated (and published) more OS-9 software and other OS-9 material, than probably any other publication. I continually receive correspondence and telephone calls from both domestic and overseas users and potential users, inquiring about OS-9. We also are a 'clearing house' for OS-9 information. Having a good relationship with the guys and gals in the trenches can be often quite productive. I could (but won't) name you several (or more) large users who were led to OS-9 through their personnel who were acquainted with OS-9 through 68 Micro Journal articles. Having a productive users group to refer to will certainly help in introducing OS-9 to newcomers as well as enlighten some of the older non-group users. Mutual cooperation between us can only help!

Even Peter Dibble's book that we are publishing, is also being published in Japan, in Japanese of course. As well as many selected articles from 68 Micro Journal. To be able to honestly report to them that OS-9 enjoys an active and productive users group is a plus.

As to a rep from 68 Micro Journal being on the board or an officer, would, in my opinion be a mistake. As long as competitive groups have mutual interest in a particular project, it would be a tactical mistake to invite or even allow either faction a voting or directional place in the administration of the operation. I think that you (the group) may have already experienced some of that. Where one competitor or another has access or privy to information or can control, to some degree or another, any facet of the group, then you can readily predict contention

and animosity between factions within the group. And we both know that bickering factions is the one thing the group does not need. Strong but impartial leadership with strict non-commercial, non-factional board, officers or committee members seems to be a need not to be denied.

I believe that we have already demonstrated our desire and willingness to do all we can for the group. After all, they are for the most part, our customers and readers also. We owe them that!

So, thanks again Dale for your thoughts and kind words expressed. I want to also acknowledge all those you mention (and some I imagine not mentioned) who have given freely of their time and energies to the group. To them we all owe a large portion of **gratitude and thanks**

DMW

***

Preliminary Information
Delivery: September 85

GMX™ Micro-20™

32-bit MC68020 based Single-board Computer

The Micro-20 single-board computer provides the basis for a powerful, compact, 32-bit computing system featuring the state-of-the-art Motorola MC68020 microprocessor. It also includes: 2 megabytes of high-speed dynamic RAM, serial and parallel I/O ports, a floppy disk controller, a SASI interface for intelligent hard disk controllers, and a time-of-day clock with battery backup. The board has provisions for an optional MC68881 floating-point coprocessor for number-crunching applications. An optional network interface uses one serial port as a 125K bits/second network channel that supports as many as 32 nodes.

The Micro-20 is ideally suited to a wide variety of applications. It provides an cost-effective alternative to the other MC68020 evaluation and development systems currently available. As an educational tool, it provides an excellent introduction to the power and versatility of the MC68020 microprocessor. In practical applications, it can be used as a software development station, a general purpose scientific or small business computer, or a real-time controller in process control systems, to name just a few of the possibilities.

The only external devices required to form a working system are a power supply and a standard serial (ASCII) terminal. This basic configuration is easily expanded to include one or two 5 1/4" floppy disk drives, a SASI compatible controller with one or two hard disks, up to 3 additional terminals or other serial devices (4 total), and a parallel printer. System packages, with the Micro-20, cabinet, power supply, and disk drives will also be available.

The Micro-20 is compact, mounting directly to a standard 5 1/4" disk drive, and can easily be incorporated into new or existing product designs for OEM applications. A small auxiliary board provides RS-232 drivers, receivers, and standard (D-type) connectors for the four serial ports. Connectors on the main board can be cabled directly to the floppy disk drive(s), SASI hard disk controller, parallel printer (Centronics-type interface), and power supply.

A GMX version of Motorola's 68000ug is included with the Micro-20. 68000ug is a ROM-based debugging package with facilities for downloading and executing user programs from a host system. It includes commands for display and modification of memory, breakpoint capabilities, a powerful assembler/disassembler, and system diagnostics. Various 68000ug system routines, such as I/O handlers, are available to user programs.

The Microware OS-9/68000 disk operating system will be available as an option. This multi-user, multi-tasking OS is suitable for both disk and ROM based systems.

FEATURES

- 12.5 MHz MC68020 32-bit Microprocessor
  - 32-bit wide non-multiplexed data & address buses
  - On-chip instruction cache
  - Object-code compatible with earlier M68000 family processors (68000/68008/68010)
  - Enhanced instruction set
  - Coprocessor interface
- Optional 68881 Floating-point Coprocessor (12.5 MHz)
  - Direct extension of 68020 instruction set
  - Full support of IEEE P754, draft 10.0
  - Transcendentals and other math functions
- 2 Megabytes of RAM (512K x 32-bit organization) †
- Up to 128K Bytes of EPROM (64K x 16-bit)
  - Uses two 2764, 27128, 27256, or 27512 EPROMs
- Four Asynchronous Serial I/O Ports (2 x MC68681 DUART)
  - Software programmable baud rates to 19.2K
  - Standard RS-232 interface
  - Optional network interface on one port
- Buffered 8-bit Parallel I/O Port (1/2 MC68230)
  - Centronics-type parallel printer pinout
  - May also be used as an input port
- Time-of-Day Clock/Calendar w/Battery Backup
- Controller for up to two 5 1/4" Floppy Disk Drives
  - Single or double sided
  - Single or double density
  - 48 or 96 Tracks Per Inch (40/80 Track)
- Mounts Directly to a Standard 5 1/4" Disk Drive
- SASI Interface for Intelligent Hard Disk Controllers
- Programmable Periodic Interrupt Generator
  - For time-slicing and real-time applications
  - Interrupt rates from microseconds to seconds
  - Highly accurate timebase (5 PPM)
- 5-bit sense switch, readable by the processor
- Hardware single-step capability

† A 1 Megabyte version of the board will also be available, with the same features as the 2 Megabyte Board. (The 1 Mbyte board can not be expanded to 2 Mbytes.)

GMX Inc - 1337 W 37th Place - Chicago, IL 60609
(312) 927-5510 - TWX 910-221-4055

***

COMPUTER EXCELLENCE INC.
4834 N.E. 12th Ave
Fort Lauderdale, FL 33334
305 752-8321

Larry Williams
P. O. Box 849
Hixon, TN 37343

Dear Larry;

The price of 256K DRAM chips has dropped again. We would like to announce an immediate price reduction. The 1 Meg. card will be $895., the 512K card will be $745, the 256K $695, and the 128K $575. We will also sell an assembled PC card without RAM for $495 and a bare PC card for $150. Included is the Manual and the TURBO Disk Software with the new TURBOCOPY routine.

Thank you;

T. D. Farnsworth
Vice President

***

Coeputer Publishing Center
68' Micro Journal
5900 Cassandra Smith Road
Hixson, Tn. 37343

Christopher F.Deme
1435 Lexington Avenue
New York, N.Y. 10128

Dear Don:

I have never written to your publication before but I am doing so now for several reasons.

First off I would like to say a long overdue big THANK YOU for publishing the 68'MJ. I have subscribed to your publication for several years now and have, as always, enjoyed the articles and useful programs that you include. Many of the programs have proven to be instructive and very useful to my work as well as to my computer hobby. I own a SWTPC 6800 computer which I had purchased and built back in 1975. My system is one of the old timers by todays standards but it is alive and well and quite useful to me running my own software and programs which run under TSC's Flex 2.0 DOS control. Please don't stop coverage of the 6800 series of processor chips if it is at all possible. I would be forever lost without you!! Ron Anderson's "Flex User Notes" column is most dear to me as I am sure it is to many of your readers.

Getting to my next reason for writing I would wish to say after having read the August issue and the letter from Mr. O.E. Groves. I could not agree more. I too have had the good fortune of dealing with Jerry Koppel from AAA Chicago Computers in the Past and am very happy to see that he has surfaced again. I have delt with several suppliers that advertise through your publication and have always been quite satisfied with their product. I feel very strongly about supporting those companies and people that have stood by us all these years. To me the difference in price on various items that can be purchased through your advertisers as apposed to the rest of the world is well worth it! The helpfulness and the courtesy beyond the call of duty that these people have shown me has made me a loyal customer. It may sound abit corny but I would like them to know that I DO CARE about them too!

Lastly, since I have been getting useful information through you all these years I felt it was time to offer something back in return. If you feel it would be of interest to your readers, I have written software for and built the necessary hardware needed to control a personal home telephone communications system.

The system I built is currently in use as the main local internal telephone system for the building in which I live in New York. The system can be used on anything from two stations to over one hundred if desired. The best part of all is that it is all controlled by my SWTPC 6800 computer! Through software I am able to use the computer to monitor other things in the building at the same time as well like security, temperature and paging facilities.. I have my 170 terminals remotely located from the computer and use them for system status reports as well as for personal work. My system has been running this way 24 hours aday 365 days a year for the last four years. Barring Con Edison

electric it has been running like a champ. I feel that no computer should be sitting on the shelf when there is all that work out there to be done! Let me know if you would want an article of this nature.

Forgive me if I have rambled on abit but once I get started I find it hard to stop!

Sincerely yours,

Christopher F. Deme

Editor's Note: Chris, thanks for the kind words. I hear the sort of thing you are saying quite often.

As soon as an advertiser drops out of 68 Micro Journal, for any reason, I am soon flooded with calls and letters, wanting to know what happened. When one has gone under, you would think that a close personal human friend had died. Our readers have a much closer relationship with our advertisers than any other computer magazine I know of. It is sad but true, some are failing. But others seem to thrive in the same market. Why? I think I know, but some folks just seem determined to go their own way.

I guess that one of the most important considerations is *SERVICE and communication*. When an advertiser start to look shakey (hard to get on phone, quality gone to pot, advertising and customer communication decreased or gone, etc.) then most of us begin to wonder. After all, we only want to do business with someone who is strong enough to survive, service and maintain any products I might have bought from them. Nothing is more disheartening than to spend hard earned money and then find out that I am alone with an unreliable product. No support, no communication, NO SALE!

And as you say, *"I am happy to see that he has surfaced again."* Well, so were we. Some listen a little better than others. As for the article - I and thousands of reader/users would appreciate your sharing your efforts with us.

DMN

- - -

MICRONICS RESEARCH CORP.

Microcomputers - Hardware and Software
GIMIX® Sales, Service and Support

33383 LYNN AVENUE,
ABBOTSFORD,
BRITISH COLUMBIA,
CANADA, V2S 1E2

Don Williams,
68 MICRO JOURNAL,
5900 Cassandra Smith Road,
Hixson, TN 37343

Dear Don,

Yesterday's mail brought me both the July issue of 68MJ and also your acknowledgement of my recent mods to FLEX to encode the time of a file's creation. I really appreciated your kind comments which came with it.

I notice that in your editing of my letter re the patch to EDIT command that, apart from a few minor typos, my symbol for backslash (\) somehow got changed to a forward slash (/). May be you could clear this up for your readers. More importantly, since using this patch rather extensively, I've discovered a flaw in FLEX's DOCMND. Suppose, for example, that we exercised LIST <filename> under DOCMND, and part-way through - having found the section of text we were after - we chose to exit by pausing the display with ESC and then aborting with a CR. It seems that DOCMND also aborts, and does not proceed to its normal close. Instead of returning to the address stored in $CC43 the program returns instead to the address stored in $CC16. Even worse, it leaves the CMDFLG at $CC28 uncleared, so although EDIT is still apparently working OK, peculiar things happen when we try to leave EDIT, as FLEX thinks it is still executing DOCMD.

In order to get DOCMND to terminate correctly under the conditions set out above it becomes necessary to re-direct the Warm-Start of EDIT so that it will automatically clear the CMDFLG. As a result, I have re-written the EDIT patch to accomplish this. Here it is in full :

1. GET O.EDIT.CMD, then use Monitor's Memory Examine/Change to
2. Change the code at $02E3 from 30 8D 00 A6 to 30 8D 16 C8, which will stuff a new warm-start address of $19AF into ESCRTN at $CC16.
3. Append the following to EDIT (starting in my case at $19A3)

```
19A3 ... 8E 19B5 BD CD1E BD CD1B BD CD4B (call OOCMND)
19AF ... (WARM-ST re-direction) 7F CC28 (CLR CMDFLG)
         7E 038D (back to official Warm-st)
19B5 ... 0D 0A 46 4C 45 58 20 43 6F 6D 6D 61 6E 64 20
         2E 2E 2E 20 04 (CR LF FLEX Command ... )
```

4. Back to FLEX and execute SAVE O.EDIT.CMD 0000 1A07 0000. and that's it!!!

While I'm about it, I should mention that since sending you my disk of stuff, I've also carried out a minor modification to the DATE command, to make it, too, compatible with the time-encoding scheme. Now, when using the option to change the file's creation date to the current date, it will encode the current time as well. You should therefore substitute the new DATE.ASM for the previous one on your disk #19. For the enlightenment of my fellow-hackers out there, I would like to add that the overlay patch to FLEX has been written as a subroutine, which, when called, will return with the time encoded as a single byte in Accumulator A.

This will have to be regarded as my submission for July, as I shall be going on vacation soon. Will try and pick up again in August or September with some new stuff. Best wishes to you and all at 68MJ.

Sincerely,

Bob

R. Jones
President

Editor's Note: Thanks Bob, you folks at MICRONICS are certainly among our most active supporters. Your input is certainly appreciated.

I will let your letter speak for itself. The many games and other programs will be offered on a Readers Service Disk (number 21) for the benefit of all. Hope you have a most enjoyable vacation (whatever that is)! Looking forward to your next offering. Thousands of us appreciate your effort and sharing.

Thanks again Bob, and have a good 'un.

DMN

- - -

Dear Editor,

This is a request for help or more plainly stated ..."how do I adjust to living with a computer pack-rat?"

Last fall I married Chuck Adams knowing little about computer people. I have found them to be warm, friendly, bright but adverse to throwing out anything SOFT or HARD !

In Chuck's article, August 1985 issue, he said, "...it took a great deal of soul searching to put an old friend away. I needed more room on

the desk." It is true....there is no room left!

Any helpful hints from veterans in my dilemma?

Judy Adams

Judy Adams
P.O. Box 6809
Denton, Texas 76203

*Editor's Note: Gosh Judy, I dunno. I've known 'Bear' for quite a few years now, but I do remember receiving a letter from a reader a few years back. It was to the point - "PLEASE, cancel immediately my subscription. My wife has given me an final warning, - It's either get rid of all the computer stuff, or she goes!" A few days later we received another letter from the same reader, this time is was Air Mail, Special Delivery, Special Handling, you know, the URGENT type. It simply stated - "Had second thoughts, kept the computer, keep my subscription coming as I have a lot more time for hacking."*

*Actually, honest injun, we did receive a letter as above. I keep checking the subs file, from time to time, and he is still current. So Judy, I leave it up to some experienced reader to guide you in the proper manner.*

*DMN*

*P.S. Congratulations and we all wish you and 'Bear' many years of pleasant 'puting.*

- - -

## A Computer Odyssey

I bought a Southwest Technical Products 6800 system in 1976. I started out using an old Teletype ASR-33 as a terminal, relying on the paper tape punch reader on the TTY for program and data storage. About a year later Percom Data ran some ads for a cassette tape adapter, I had to have one. Next their ad was for the LFD-400 disk system. Again I had to move up. With the disk system and an adapter program, which also came from Percom, I moved up again to Flex-2. I then thought I had everything. Up until my 6800, I had been using mainframe systems. Now I had my very own computer. Immediately I modified every piece of software available to run on that 6800. Many days and almost every night was passed writing every utility I could think of using.

I have the very first issue of almost every micro computer magazine since 1976, including 68 Micro Journal.

Things were great for a few years. I use 8080 based systems at work, we build alot of proto-type systems (data collection and controllers etc.), so I know how good the 6800 series of chips really are. Then everything began to go 6809. I just had to have one. This time I bought boards, blanks and wire wrap, and built a 6809 system from scratch based on the SWTPC 6809 system. I did use a Southeastern Micro Systems, Inc. disk controller and an F&D mother board. Flex 9.0 version 2.6 came from SWTPC. I was all set. Everything worked flawlessly, for a while. The crash came when I tried to use the print spooler. I used the 6800 print spooler all the time and not having a spooler on the 6809 was intolerable.

What to do? The best thing to do is go right to the source of Flex, right? Wrong! Technical Systems Consultants may have written the original Flex systems, but, SWTPC modified them for their systems so they politely told me to call SWTPC. Ok, makes sense to go to the next source, so I called them. First it was a hardware problem, (their decision) so I talked to the hardware section. After explaining exactly what equipment I was using and hearing the man at the other end if the line groan, it became a software problem. This time the decision of what to do was quick, I needed the new version of Flex 9. Version 2.8:3 came after my check and disk were sent in and I was all fixed up and ready to spool. Wrong! I had the same problem.

I looked through my 68 Micro Journals and there it was, a fix for my problem (page 35, Feb. 1982, A letter from Jim Cordill and John Moses). I had to do a little hunting around for the right locations to change, but I typed it in using the disk edit utility and my print spooler worked with my old parallel printer, but not with a serial printer. The parallel printer was not letter quality so this was no good. Another call to SWTPC. After being shuffled around to half a dozen people, it was decided I had a problem. The problem was definitely mine. Their Flex 9 worked! No one else had complained with this problem so it was my hardware or something.

I definitely had a problem. I deleted all the print spooler utilities off my disk an used Flex 9 without them. Occasionally I would send print files to my 6800 and let it do the printing. This was the way it would be, but I couldn't stand it. It had to work.

What to do? After thinking things through, I hooked an oscilloscope to the IRQ line. The timer was interrupting the processor, but the interrupt was not being reset. My 6809 was in an endless loop. Where? Next thing to do was to write a routine to tell me where the processor was, and set up the NMI vector in ROM (see Abort Switch for 6809, 68 MICRO Journal, August 1981, page 35). Now when it would go off to never land I could go with it. It turned out that an interrupt handling routine for the print spooler starting at $DB9F was where the processor was going. After looking at the code in this area I found a BEQ ($27) instruction at address $D3AA, which could never be satisfied. I couldn't find a reason for it being there so I converted it to a BNR ($21) instruction. The tension builds. Needless to say I was excited. Had this one instruction been hanging my system all this time? RIGHT! The print spooler now works the way I expected it to work. I still don't know why the BEQ was there but NOPs or BNR seem to work equally well. With this change you don't even have to change the timer board port location. I'm again very pleased with my 6809 system and my 6800 system. Also I enjoy reading through the 68 Micro Journal. I am sorry to see Flex on the decline, but I won't be moving to OS9. What I have now will have to do.

I guess the moral of this little story is maybe we should believe in what we have and get to know it as beat we can before we jump into something else. I enjoy my 68XX systems. I have PL9 also, a fantastic little language, but that's another story.

Howard Thomas
5118 Russett Rd.
Rockville, Md.
20853

d.p. johnson

microcomputer consulting

7655 southwest cedarcrest street • portland, oregon 97223 • (503) 244-8152

**NEW PRODUCT ANNOUNCEMENT**

A 512K RAM DISK cartridge for the Radio Shack Color Computer is now available for $298. The CCRD cartridge is designed to plug into the multipak expansion bus of a CoCo running OS-9. OS-9 drivers for the device are available separately on disk for $20. The CCRD cartridge together with the drivers provide a half megabyte of ramdisk (/r device) which can be used by any program just like a disk drive but provides several times faster access. Two cartridges may be used in one system to provide a single device with 1 Megabyte of storage. A ram-disk is especially useful for speeding up compiles of the C or Pascal compilers by moving programs, source code and libraries to the ram disk. For more information or to order contact:
D. P. Johnson (503) 244-8152.

***

data⇌link ag

Computer-Anwendungen

8128 Hinteregg Zürich Switzerland
Tel 01-984 2984

Stiefelwis 10
Telex 59 887

Dear Don,

I read your magazine for years and like it more and more. Hereby I send you an article for your magazine.

HOW TO BURN 27128 EPROMs ON THE WINDRUSH UNIVERSAL EPROM PROGRAMMER ?

Originally the Windrush Universal EPROM-Programmer can only be used to burn Eproms up to 2528 ( a not very usefull device :).

A small modification makes it possible to burn the well known 27128 EPROM of any made.

Hardware: strap a Wire-Jumper from IC3 pin15 to the 28-Way Eprom-Socket SKTB pin26 (A13).

Software: replace the Eprom-Identification-String "2528" by "27128". Change the Switch-Settings to the same as the 2764-Eprom. Thata all !

Further enhancements have been made to the Software like: 8-Bit Checksum-Calculation, Data-Verification ... . Send $10.- for the Patches on disk.

Is there anybody who has modified this Programmer to support the Intelligent Programming Algorythm ?

Kind regarda

Peter Keller

Mr Don Williams ***

This is a first for me. I've been reading technical journals since WW2. and this is the first time I've ever sent a letter to the editor.

First let me join the bandwagon and congratulate you on a fine Journal. Second I did not want you to think that one of our readers was abandoning you. I've had a subscrition in my company name (Serial Electronics Inc.) since Sept.79 and a subscrition to Color Micro Journal in my own name since it was started. The extra journal went to a friend, now I'm letting one subscription run out.

Some background. I do electronics for a research co. and I run my own company (Serial Electronics Inc.)on the side. In those capacities I monitored MCU's. The 4004,4040,8008,8080 and then the 6800 were introduced. I could not wait any longer. In the spring of 76 I bought a E.P.A. 6800 based computer. It had a CPU bd. with a hex key pad, 7 seg. readout for the address & data buss and a 8K memory board. It also had the origional mikbug monitor. All of that for only $1200 WOW. Also bought a $900 Adam 3A terminal.I was in my seventh heaven. With the addition of 2 8K memory bd.'s, the system was used untill the spring of 1980.

At that time I purcheased a M.S.I. 32K, dual disk 6800 based computer also a Viewpoint terminal. This combination is still seving well for the design of monitor and control systems. Oh yea a few months back I bought a COCO 2 for fun. Now it is growing.

Again thanks for Publishing the 6800 Micro Journal all these years.

Sincerely

Sam Reidelberger

S.Reidelberger
Sr. Des. Engr.
Serial Electronics Inc.
5214 Cerritos
Orange Cal. 92669

***

Don Williams
68' Micro Journal
5988 Cassandra Smith Rd.
Hixson, TN 37343

19 Davis Road, MB-6
Acton, MA 01720
29 June 1985

Don,

I'm sorry to hear you had a bad experience with the OS-9 User's Group. In all fairness though, the User's Group has come a long way since those early days two years ago. I have received seven issues of MOTO, the User's Group news letter, and several disks from the software exchange library. As far as I can tell, the User's Group is doing just fine. I'm sure if you wrote them a note explaining the problem, they would be happy help.

The Software Library Peter Dibble mentioned in his column is for real. There is also a lot of excellent information being exchanged in the 09 Forum on Compuserve.

As you pointed out in your comments, the User's Group is run entirely by volunteers who do this in their spare time. If we all pitch in, we can make it work.

Sincerely,

Peter Sichel

Peter Sichel, W8PS

P.S. I really enjoy 68' Micro Journal, and appreciate that you "call'em as you see 'em". Keep up the good work.

*Editor's Note: Thanks Peter for the prompt reply to my editorial. I believe 1,000% in volunteers. We are sustained by volunteers! GOD BLESS 'EM ALL!. However, you have to understand that not everyone marches to the same drummer. But most all are at least pointed and marching in the same direction. I have found that you just need to keep your promises and schedules flexible. After all, for most, payday just don't come around in a volunteer organization. It is for the mutual benefit of all.*

*We monitor Compuserve quite often, but don't see a whole lot going on. Seems to be the same folks mainly, most of the time. It appears to be hash and rehash of the same old things, or past discussions, with quite a bit of peddling*

going on by some. Oh well, you pays you phone bill, and modem where you want.

Frankly, I know most everyone in the group, if not on a first name basis, at least by reputation. Most are fine folks who really enjoy their computers and have become quite proficient in the use of OS-9. However, all our surveys indicate that 'long line' rates are simply too expensive to really use a system such as Compuserve, unless you have some type of special phone privileges.

To down load data files at 300 baud, that is files of any size, probably cost more than it is worth. It is far easier to swap a disk with someone by mail or UPS, or order a 68 Micro Journal reader service disk. However, the cost is not too bad if you live in the Midwest, where the system is located. And of course, the OS-9 Users Group disk service is a swell deal, and the best, cost wise. Like I said before, we never did receive ours, so I can only go on what you and several others have relayed on to me. The support that Microware lends to the group is an important factor. But I don't believe that Microware exerts pressure, just cooperation. And that is a good sign of potential success. With technical input from Microware and others, along with the efforts of the volunteers, it should be a robust organization. Time will tell. Of all the thousands of OS-9 users, I wonder how many are members of the Users Group? I would recommend that ALL OS-9 users belong and contribute. New blood keeps things pumping.

Again thanks Peter for your kind and prompt reply. If they can get those with 'special' interest to quit trying to use the group as their own personal apple crate or sales group, and really devote the energies of the group to the group as a whole without any 'SPECIAL or profit motivation, I predict that your rolls will grow faster than warts on a pickle.

Best of luck and keep in touch.

DMW

***

**CREATIVE MICRO SYSTEMS**
3622 CERRITOS AVENUE LOS ALAMITOS CALIFORNIA 90720 U.S.A. (213) 493-2464

EXORbus* Compatible 64K Non-Volatile Memory Module

The 9636 is a 64K Non-Volatile Memory Module that is organized as eight 8K contiguous blocks of memory. Each block can accomodate the standard 8K byte wide EPROMS or RAMS in sockets which are individually enabled by an eight position dip switch.
An onboard lithium battery and a power fail detect circuit are provided to save the data in the RAMS in the event of power interruptions. The 9636 also provides extended address decoding as an option for use in systems requiring memory management.

The 9636 is specifically designed for compatibility with the M6800/M6809 microprocessor bus. It is pin and outline compatible with the EXORciser* and other industry standard cards. This module is intended for use with either 1 mHz or 2 mHz systems.

The 9636 is priced at $270.00 without EPROMS or RAMS in single quantity. Delivery is from stock.

Additional information is available from Creative Micro Systems, 3622 Cerritos Ave., Los Alamitos, California 90720. Telephone 213/493-2464 or 714/761-2695

***

| Mail | New Login |
|---|---|
| Version 5.0 | Version 1.0 |
| by MSM Software | by MSM Software |

**Description**

Mail is a better version of the electronic mail program delivered with UniFLEX*. It is compatible with the existing program, and adds many features and conveniences. Among these are the ability to read and delete messages one at a time, greater flexibility in preparing and sending mail, plus the ability to tailor the program to personal tastes.

**Selected New Features**

- Commands while reading mail include: delete, echo, exit, forward, mail, print, quit, reply, scan, undelete, write, vset. Pipe message though command. Execute command in sub-shell.
- Commands while sending mail: copy, delete, edit, print, read file, set subject, set to, exit, execute sub-shell command.
- User defined variables for named groups of users, file names or commands.
- Automatic initialization through a setup file.
- The ability to read a mailbox other than ".mail".

**Description**

The login program on a UniFLEX* system is used to allow users access to the computer system. It provides a password security for individual users on a multi-user system. The New Login program provides the same features, plus enhancements providing for more security and flexibility.

**New Features**

- The ability to establish "secure" terminal lines where the system manager is allowed to log in. Logins as "system" will be refused at any other terminals.
- Individual terminal lines can be configured to limit the number of tries, and maximum time that can be spent trying to complete the login procedure.
- Logins may be disabled prior to shutting a system down, or when the system is being heavily used.
- A record of when and where a user last logged into the computer, which is displayed each time the user logs in.
- The message of the day can be suppressed once a user has seen it. Login will not print the message unless it has changed since the last time the user logged in.
- A quiet login ability where no messages will print.

For more information contact:

MSM Software
572 Freeman Street
Corning, NY 14830

***

**MOTOROLA INC.**

Microprocessor Products Group
3501 Ed Bluestein Blvd.
Austin, Texas 78721

For further information contact:

Editorial Contact: Jackie Marsh

512 928-6223

Reader Contact: Dean Mosley

512 440-2839

## MOTOROLA INTRODUCES THE M68000KIT, A DESIGN KIT FOR THE M68000 FAMILY

Austin, Texas, July 1, 1985... Motorola Microprocessor Products Group has assembled the M68000KIT, a design package which enables the design engineer to develop superior M68000-based systems. The design kit contains the MC68000 and MC68008 microprocessors, six peripheral devices, related data sheets, application notes, and supportive documentation.

Included in the M68000KIT are the following devices:

*MC68000 - 16-Bit Microprocessor

*MC68008 - 8-Bit external data bus version of the MC68000

*MC68230 - Parallel Interface Timer

*MC68901 - Multifunction Peripheral

*MC68440 - Dual DMA (Direct Memory Access)

*MC68652 - Multi-Protocol Communications Control

*MC68661 - Enhanced Peripheral Communications Interface

*MC68681 - Dual Universal Asynchronous Receiver/Transmitter

DYNACALC (Radio Shack Version)
**A DYNAMITE CALCULATOR**
by
Ron Voigts

It is exciting to seeing all the new software coming out for the Color Computer. I'm also happy to see good quality software. One of the newest releases from Radio Shack is DYNACALC, an electronic spread-sheet. It makes a wonderful "What if" forecaster for all your numerical needs. It is written for OS-9, but you don't need OS-9 to run it! (More about that later.)

For many, the concept of an electronic spread-sheet or worksheet calculator needs no explanation. Others wonder what the excitement is about. Everyone has had experience with worksheet calculations, only they may not be aware of it. If you've found yourself jotting figures and calculations on a piece of paper, you know about them. If you've done income tax, you've worked a worksheet before. If you've ever filled out a credit application, you know how to use one. In fact, anytime you have put a number on a piece of paper and did something with it, a worksheet was being used.

DYNACALC removes the sheet of paper and lets you use the TV screen for calculations, manipulations, and just plain ciphering. It gives you a matrix off 256 x 256 cells that let you write numbers in them, or labels, or equations based on what is in other cells. Only 45 of the cells show on the Coco's screen at one time, but you can scroll in any direction using the arrows on the keyboard. Entering a ">" will let you move to any cell instantly. Here's what a sample screen looks like:

```
C12
C/A  7892  READY
     [   A   ][   B   ][   C   ]
  1- MONTH     SALES
  2- ********************
  3- JAN         1200.00
  4- FEB         1578.35
  5- MAR         2692.40
  6- APR          588.88
  7- MAY          499.32
  8- JUN          789.72
  9- ********************
 10- AVERAGE     1224.78
```

All cells are referenced by a letter and number. A1 has "MONTH" in it. A3 has "JAN". B6 has the number 588.88. The power of DYNACALC is not just being able to write down numbers and labels, but to do calculations with them. In cell B10, I told it to average cells B3 through B8.

DYNACALC has 27 math functions. They are ABS, ACOS, ASIN, ATAN, AVERAGE, COS, COUNT, ERROR, EXP, INT, LN, LOG, LOOKUP, MAX, MIN, NA, NPV, PI, RND, ROUND, SIN, SQRT, SUM, and TAN. Most everyone is familiar with ones like COS, SIN and LOG. They may not be so familiar with ones like NPV -- net present value for all the economists. Or there is STDDEV -- standard deviation for the statisticians. With the math functions are the standard operators -- addition, subtraction, multiplication, division and expoonentiation.

There are also 9 logical functions. They include TRUE, FALSE, ISERROR, ISNA, IF, NOT, OR, AND, and EOR. And to use with these, there are the logical operators, which are =, <>, >, <, >= and <=. This brings the tally to 36 functions and 11 operators. That is a lot of calculating power!

Besides all the functions, there are commands available that let you control the worksheet. Here are the main commands:

A - Attribute
B - Blank Cell
C - Clear Work Sheet
D - Delete a Row or Column
E - Edit a Cell
F - Format a Cell
I - Insert a Row or Column
K - Keysaver
L - Locates a Label
M - Moves Cells
P - Printer Output
Q - Quit
R - Replicates Cells
S - System Commands
T - Titles
W - Windows

There are 16 main commands in all and under many of these are subcommands. For example, under Attribute you can select from 18 subcommands, such as column width, printer/textfile attributes and characters used for printing graphics. Some have one function, like "B" which clears a cell of any information. One of the more interesting ones is "W" which adjusts the screen's window. Among the many things it does, it permits you to split the screen into two windows. You can scroll the contents of one window while keeping the other fixed. This is good if you want to bring together two rows or columns that are normally not next to each other.

Another nice feature is that there is help at any point when working on Dynacalc. To enter a command a "/" is first entered from the keyboard. Immediately a list of letters appear giving possible choices, like "A" for Attribute, "B" for Blank Cell, etc. One choice is "?". Typing this key will print what each of the Commands does. Now let's say you choose "F". The next list of choices will be what is available from Format. Can't remember what the letters mean? Type a "?" and a new list of help appears. In fact, once you've read the manual, you should be able to run Dynacalc without having to go back to reread things. If you get into trouble, just use the handy help feature.

I think an important feature of any software is the manual. The DYNACALC manual is not exceptionally large; some of the previous manuals for OS-9 from Radio Shack were at least 100 pages, plus. This one is only 76 pages. But the manual covers all the commands and details of DYNACALC. Chapter 1 contains a sample session using it to set up a household budget. There is also an advanced session at the end of the book. It is the same budget program, but with more advanced features. The best way I've found to learn it is to sit down at the computer with the manual and try things as they come up. Also, read the appendixes at the end of the book. There are some valuable things there.

If you still need help, try your local library or bookstore for a book on Visicalc. Visicalc is a worksheet calculator by Visicorp. The commands in DYNACALC are almost identical to Visicalc's. I got a book from the library and tried examples from it. They worked the first time! There are a few small differences. For example, I've noticed that Visicalc uses "/GF" for global format, while DYNACALC uses "/WF" for window format. They both have the same effect. There are some others, but I don't believe you'll have any trouble translating them.

Another nice feature is that the joystick or "mouse" can move the cursor around the screen. Just plug it into either port, press the "fire" button and you've got remote screen control. For moving the cursor off the screen, you'll still need the arrow keys or you can use the ">" to travel to off screen cells.

Finally, you don't need OS-9 to run OS-9 DYNACALC. The disk comes set up and ready to run. All you need to do is put the disk in drive 0 and enter "DOS". Before you know it, you're in OS-9 and ready to run DYNACALC. For those of you that have the older disk controller, the appendix of the manual has a small Basic program that will boot it for you. Information for backing up the original disk is included in an appendix, too. If you have OS-9 already, you'll feel right at home with it. If you don't, after running DYNACALC, I'll bet you want to get it.

DYNACALC is great for all your calculating needs. Whether it's science, business, school, home, or fun, you'll find a use for it. You can use worksheets for financial statements, budgeting, cash management, balancing checkbooks, doing school work, and whatever your imagination can create. DYNACALC is available from Radio Shack for $99.95. It would make a nice addition to your OS-9 program library. And it's a dynamite calculator!

# CoCo OS-9 PASCAL Review

A NEW MEMBER TO THE
RADIO SHACK OS-9 FAMILY
by
Ron Voigts

Years ago there was a need for a computer language that would be relatively easy to learn and use, and yet would be able to handle complex algorithms. Fortran was around, but it had developed a complex set of routines that made it cumbersome to use or learn. Basic was inspired by Fortran. It was easy to use, but often inadequate for writing complex programs. Algol could handle the complex algorithms, but it was difficult to learn and implement. Then in 1971 at ETH, Zurich, Switzerland, Professor Niklaus Wirth developed a language that fit the need. It was easy to learn, since it was designed specifically for teaching Structured Programming and to force the students to be cognizant of the data types used, etc., and yet it could handle complex algorithms. Instead of naming it by acronym as had been done before, he named it after a man, the French mathematician, Blaise Pascal. That year the PASCAL Programming Language was born.

Pascal became the starting language of students in colleges and universities. It was easy to implement on today's computers. Complex algorithms could be written using Pascal's control statements. And it allowed the development of well-organized and well-structured programs. Since the students learned Pascal in school, it naturally carried over from the academic world to the outside world. Everyone used it. It became the language of choice.

Now Radio Shack has brought Pascal to the Color Computer. What makes this even better is you run it on OS-9! The OS-9 PASCAL is licensed through Microware. It conforms to the ISO standard 7185.1 Level 0. It also acts according to standards set by Wirth (remember Nick from a few paragraphs ago?) and Jensen. The system comes on two disks, since there is so much in it. Be of good cheer, you can run it on one disk drive. I ran my first program on only one drive. However, I do recommend using two drives if you can.

The manual points out, "Either you already know Pascal, or you don't." I have to admit this is true. I've had some experience with it. I entered my first program and was running in minutes. The little program I entered was:

```
program myhello(output);
{ This program will echo a greeting
  to the screen.                    }
begin
   writeln('HELLO, THERE!')
end.
```

If this doesn't make sense, then you're probably one of those who doesn't know Pascal. Fear not! There are many good books around that cover it. The OS-9 Pascal Manual suggests three of them. I recommend checking your local library for more.

Most Pascal's have one, or maybe two, modes of executing programs. OS-9 Pascal has three! Before you can use any of them you must compile your source into P-Code. The system comes with a compiler appropriately named PASCAL. Compiling is as easy as entering:

```
PASCAL <MYFILE
```

It starts to compile MYFILE, while it checks for proper syntax. It generates useful information like pseudo-line numbers and P-Code bytes created. When it is through you have a file called PCODEP, which is the compiled version of your program. PCODEP is a default name and you can name it whatever you like.

The first option to run your program is PASCALN. It is the normal Pascal interpreter, hence the "N" in its name. Its usage looks like:

```
PASCALN PCODEP
```

Besides running your program, it is checking for run-time errors. If it finds an error, it can tell the type, the procedure it occurred in, and the statement line in your source it occurred on. Pretty smart, huh?

Most Pascal's have one, or maybe two, modes of executing programs. OS-9 Pascal has three! Before you can use any of them you must compile your source into P-Code. The system comes with a compiler appropriately named PASCAL. Compiling is as easy as entering:

```
PASCAL <MYFILE
```

It starts to compile MYFILE, while it checks for proper syntax. It generates useful information like pseudo-line numbers and P-Code bytes created. When it is through you have a file called PCODEP, which is the compiled version of your program. PCODEP is a default name and you can name it whatever you like.

The first option to run your program is PASCALN. It is the normal Pascal interpreter, hence the "N" in its name. Its usage looks like:

```
PASCALN PCODEP
```

Besides running your program, it is checking for run-time errors. If it finds an error, it can tell the type, the procedure it occurred in, and the statement line in your source it occurred on. Pretty smart, heh?

What if your program is too large for memory? This is a problem for the Color Computer, which runs OS-9 Level I and is limited to 64K. To solve this problem, there is the second option, PASCALS. The "S" stands for "Swap". It is used like Pascaln, except this interpreter swaps parts of your P-Code file in and out of memory as they are needed. A temporary disk file is kept with the "virtual memory" pages on it. The trade-off is speed. All that swapping takes time, so Pascals programs will run slower.

You say, you want speed? PASCALT is our third option. It is used to translate P-Code into 6809 machine language. It will run faster and is reenterable under OS-9. Space used by the interpreter is freed, since the machine code has everything it needs to run. The actual process takes a few steps. First, the source code is transformed to P-Code by Pascal, the compiler. Next Pascalt is used to generate an assembly language source code. Finally, you use your assembler from the original OS-9 package to convert your Pascal program to a reenterable OS-9 module.

A big complaint in the past was Pascal's input and output operations. OS-9 Pascal overcomes this with a rich set of I/O functions to take full advantage of the OS-9 system. Besides the regular I/O's like READ and WRITE, there are special ones like SEEKEOF, REPOSTION, and SYSREPORT. In fact, there are 22 procedures in all. Who says it has to be hard to write a file in Pascal?

The entire set of Pascal commands are supported, with a few exceptions. You can use WHILE...DO, REPEAT...UNTIL and FOR...DO loops. There is also IF...THEN...ELSE and CASE...OF. It supports all of the standard variable types like integer, real, boolean and char. You can also create your own variable type if you wish. The Pascal library comes complete with all the standard functions and procedures. There are a lot of other things it offers. You can select compile-time options and run-time options. There is also PASCALE which is linkage editor. You can use it to merge main programs with

external procedures. OS-9 Pascal is a very complete system.

As with the other OS-9 manuals from Radio Shack, this one is a system manual and not a tutorial. The manual tells how to use OS-9 Pascal. To learn Pascal, you'll still need to get a good book on programming it. But I do believe the manual is fairly well written. By chapter 2 you'll be compiling and running Pascal programs. Most chapters start with a synopsis of what's in it, which is helpful when you're looking for something special. One thing I like is that the manual is bound like a real book. Radio Shack got rid of the wire spiral binder that they used on previous manuals.

I like the OS-9 Pascal for the Color Computer. Whether you want to learn Pascal, or just program for the fun of it, I think you'll enjoy it. So, if you're interested in the newest member to the Color Computer OS-9 family, you can get it from your local Radio Shack for $99.95. If they don't have it on the shelf, ask them to order it for you. I'll be running Pascal programs in my BASIC OS-9 Column from time-to-time. Should you have any questions about Pascal, drop me a line. I'll try to answer them. Happy Pascal programming.

***

E.A.P. CO.
P.O. BOX 14
KELLER, TX 76248
(817) 498-4242

If you process mail orders, or telephone orders requiring charge card orders you can now use your computer to process the charge card orders completely.

Enclosed is a sample of the continous credit card forms that you can use in your computer for processing any major credit card including American Express, Diners Club, Carte Blanche, Visa, And Master Card. You already use the computer to process the order and generate the invoicing, you can now use the same data to print the charge card depositry slip as well. This is much simpler than individually typing, or writing the charge card depositry slips. These universal charge cards are approved by all major clearing houses for the charge cards. These continous charge slips are being used by many companies throughout the country to process their credit card orders.

E.A.P. Co. can ship these to your location in a matter of days so you can be in operation almost immediatly.

The Universal charge card is available in two formats one is standard 9 1/2" wide and can be used on small printers or a double wide version for use on 15" printers. The double wide gives a customer receipt or file copy.

The forms are packaged 5000/ per carton and prices at $37.25/1000 for either the one wide or two wide versions.

These forms will comply with the new regulations requiring carbonless depositry slips since they do not produce duplicate copies. Most merchants have an invoice that produces an acceptable customer receipt and file copy for their records.

If you are tired of duplicating the work required to process the charge cards it is time to coputerize the charge slips.

E.A.P. Co. can also provide UPS approved continous COD labels for UPS cod shipments. If you have requirements for these please call. We can provide continous pre printed checks, letterheads, statements, invoices, and many other custom forms. If you need any of these call and ask for our catalog.

Call Bd at E.A.P. Co. (817) 498-4242 for prompt service.

# Classified Advertising

1-GIMIX #79 System OS/9 III, with 19 Mega Byte Mini-Hard Disk plus 5" DSDD Drive, also Dual 8" DSDD Disk System with Controller, 6-Serial Ports, 2-Parallel Ports, 256K #72 Card and 4-#67 64K Cards.

1-PT-69 complete with Dual 5" DSDD Disk System and Controller, includes FLEX DOS.

!!! Make me an offer !!!

Call Tom or Larry (615) 842-4600 M-F 9 A.M. to 5 P.M. EST

***

TELETYPE Model 43 PRINTER - with aerial (RS232) interface, and full ASCII keyboard. LIKE NEW - New cost $1295.00 - ONLY $559.00 ready to run - Call Tom or Larry, CPI 615 842-4600

***

S/09 with Motorola 128K RAM, 1-MPS2, 1-Parallel Port, MP-09 CPU Card $1990. 1-DMAF2 Dual 8" Drives with Controller $2190. 1-CDS1 20 Meg Hard Disk System with Controller $2400.

1-S+ System with 256K RAM, 1-MPS4, 1-Parallel Port, MPU Processor Board, UniFLEX included $2700. 1-QMI 10 Meg 5" Hard Disk & 8" Disk Drive with DMAF3 Controller Board $3600. 1-X12 Terminal 12" $1260. 1-Cabinet for S+ System w/Filler Plate $500. 2-8212 Terminals $495.

**I will accept any reasonable counter-offer !!**

Call Tom (615) 842-4600 M-F 9 A.M. to 5 P.M. E.S.T.

***

Several SWTP S+ Systems Available: Below dealer cost, new equipment warranty, lots of software available. Finders Fee. 512-828-2679

***

Complete 2 mhz 6809 system: SSB boards & OS; 80K RAM; dual 8" 500K drives; custom cabinets; software. Attractivaly priced. John Haugeland, Department of Philosophy, Univeraity of Pittaburgh, Pittaburgh, PA 15260; or call (412) 661-9196 evenings.

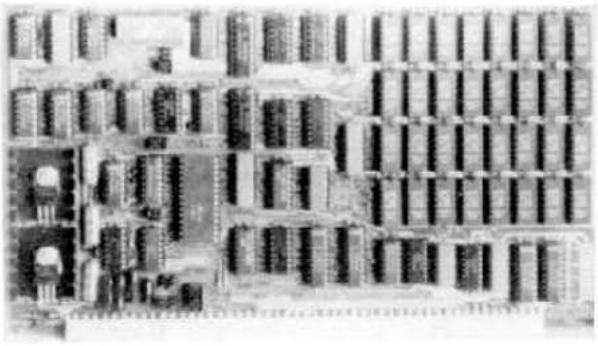
***

MUST SELL! 6809 CIMIX 128K System. 5" Drives, Clock, Flex, OS9, Forth, CPM80, $1000. Dual 8" Qume's in Case, $200. VT100 Terminal, $450. HP161IA Analizer, 6809 Module, $1500. All Offers Considered. (303)651-9402 Evenings.

***

Please publish the fact that the "C. Dragon" system at (316) 943-9716 is going "off the air" as of 8/1/85. The system will be put back into private service with a new phone #. Any questions can be forwarded to: Thomas L. Gilchrist 1450 N. Clarence #108 Wichita, Kansas 67203 (316) 943-2901


**Complete Board (No RAM) $495**



**256K, 512K, 1 MEG MEMORY SYSTEM**

Now compatible with DMA controllers. Runs at up to 2Mhz without generating MRDY or interrupts. Has an optional on board DAT for use with CPU cards without a DAT. 128K, 256K, 512K or 1M byte per card. Field upgradable. Optional configuration allows 4M byte address reach (using memory board DAT) without CPU changes or cables. 1 year limited warranty.

TURBO virtual disk software and memory diagnostics supplied with the system.

**Prepaid: 128K:$575, 256K:$695, 512K:$745, 1024K:$895**

Domestic shipping and handling $10.00. Users manual: $35.00, applicable toward system purchase. Cashiers check, COD, personal checks must clear before shipment. Fla. residence add 5% sales tax. Shipped stock to 30 days. Dealer and quantity discounts available.

COMPUTER EXCELLENCE INC.
P.O. BOX 8442
CORAL SPRINGS, FL 33065
(305) 752-8321

# FINALLY, A PORTABLE OFFICE

## (And A Way to Add IBM-PC/OS9 Compatibility)

Are you trying to get your personal and business affairs in order when your desk is out of order? Are you attempting to operate swiftly, efficiently and effectively when your reports, records and data are buried under a Matterhorn of clutter?

Now, you can clean up your act, by cleaning up your desk—because we've put together a Corona Portable Computer with Alpha Software's "Electric Desk". It's Corona's comprehensive, yet simple to use, Portable Office Package that allows you to concentrate on your work, instead of concentrating on trying to find your work.

You get the Corona PPC-400, the most advanced portable computer that's IBM* PC compatible. It features a high resolution screen (twice the resolution of the IBM-PC), a 360K floppy disk drive and 256K RAM (with room for more drives and RAM). And, as you grow, the Corona PPC-400 will grow with you. It has expansion slots for additional PC boards, and built-in serial and parallel ports so you can add other peripherals.

## $1195 AT A VERY SPECIAL PRICE

What's more, the system comes complete with "Electric Desk" ... a powerful, easy-to-learn and use package that includes word processing, spreadsheets, database management and communications (suggested retail value is $345).

**IBM/OS9 COMPATIBILITY** — Smoke Signal offers a set of programs that brings compatibility to your OS9 computer. When you order a Corona PC, we can offer you two sets of programs at a very special price that provide compatibility between the Corona and your OS9 system. The first set of programs allows your Smoke Signal OS9 system to list the directory, print the contents of a file and copy files to and from a disk created on the Corona. The second set of programs allows the Corona to be used as a terminal on Smoke Signal and many other OS9 systems and to transmit files back and forth between systems. This permits the OS9 system to be used as a file server for the PC. These two sets of programs are normally sold by their authors for over $600. With the purchase of a Corona PC, we can offer them to you for just $199.

**HARD DISK AND DESK TOP PC's** are available at very special prices available only to Micro-Journal readers and Smoke Signal customers through October 15. Call (818) 889-9340 for complete information on pricing and available configurations.

SMOKE SIGNAL corona data systems inc. / Electric Desk

31336 Via Colinas, Westlake Village, CA 91362
Phone (818) 889-9340 • TLX 910-494-4965

*IBM is a registered trademark of International Business Machines, Inc.

**COMPILER EVALUATION SERVICES**
**By: Ron Anderson**

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

**COMPILER COMPARISON AND EVALUATION REPORT**

Due to the constant and rapid updating and enhancement 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 WHIMSICAL PL/9

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

**S.E. MEDIA - C.P.I.**
**5900 Cassandra Smith Rd.**
**Hixson, Tn. 37343**
**(615) 842-4601**



# OS-9™ SOFTWARE

SDISK—Standard disk driver module allows the use of 35, 40, or 80 track double sided drives with COCO OS-9 plus you can read/write/format the OS-9 formats used by other OS-9 systems. **$29.95**

SDISK + BOOTFIX—As above plus boot directly from a double sided diskette **$35.95**

FILTER KIT #1—Eleven OS-9 utilities for "wild card" directory lists, copies, moves, deletes, sorts, etc. Now includes disk sector edit utility also. **$29.95 ($31.95)**

FILTER KIT #2—Macgen command macro generator builds new commands by combining old ones with parameter substitution, 10 other utilities. **$29.95 ($31.95)**

HACKER'S KIT #1—Disassembler and related utilities allow disassembly from memory, file. **$24.95 ($26.95)**

PC-XFER UTILITIES —Utilities to read/write and format MS-DOS™ diskettes on CoCo under OS-9. Also transfer files between RS disk basic and OS-9 (requires sdisk). **$45.00**

BOLD prices are CoCo OS-9 format disk, other formats (in parenthesis) specify format and OS-9 level. All orders prepaid or COD, VISA and MasterCard accepted. Add $1.50 S&H on prepaid, COD actual charges added.

## SS-50C
## 1 MEGABYTE RAM BOARD

Full megabyte of ram with disable options to suit any SS-50 6809 system. High reliability, can replace static ram for a fraction of the cost, $895 for 2 Mhz or $995 for 2.25 Mhz board assembled, tested and fully populated. (Add $6 shipping and insurance, quantity discounts available.)

**D.P. Johnson, 7655 S.W. Cedarcrest St.**
**Portland, OR 97223 (503) 244-8152**
(For best service call between 9-11 AM Pacific Time.)

OS-9 is a trademark of Microware and Motorola Inc.
MS-DOS is a trademark of Microsoft, Inc.



| | ASSEMBLED | BARE |
|---|---|---|
| Elektra Super Bus (Fits Current Elektra Chassis) | Coming | |
| Elektra Super 68OXX Boards | Coming | |
| Cabinet w/Power Supply (Largest SS-50 Cab. Available) | 650.00 | |
| Disk Regulator | 50.00 | |
| Disk Regulator (Heavy Duty) | 75.00 | |
| Motherboard (Gold Connectors) | 395.00 | |
| 2MHz 6802/6809 CPU | 275.00 | 55.00 |
| Dual Port Serial Card | 95.00 | 40.00 |
| Dual Port Parallel Card | 80.00 | 40.00 |
| 54K Static Ram Card | 250.00 | N/A |
| 5"/8" 2MHz Super Floppy Controller (The Best) | 295.00 | 100.00 |
| Flex/Star-DOS Drivers | 30.00 | |
| OS-9 Drivers | 50.00 | |
| Elektra OS-9 w/Editor, Assembler, Debugger | 250.00 | |
| Elektra Star-DOS | 75.00 | |

| | ASSEMBLED | BARE |
|---|---|---|
| 4K Kumbug | 75.00 | |
| 2K Microbug | 40.00 | |
| 7-84 MByte Winchester Systems | -Phone- | |
| Removable Cartridge Winchester $ stems | -Phone- | |
| DMA Winchester/Floppy SS-50 Host Interface | 395.00 | 250.00 |
| HD-5 Dual Floppy Cabinet with Power Supply | 150.00 | |
| HD-5W Floppy/Winchester Cabinet with Power Supply | 199.00 | |
| HD-5DH Dual Winchester Cabinet with Power Supply | 250.00 | |
| Ribbon Cable for Dual Outboard Floppies | 40.00 | |
| Ribbon Cable for Dual Inboard Floppies | 35.00 | |
| HD-8 Dual 8" Floppy Cabinet with Power Supply | 350.00 | |
| Ribbon Cable for Dual 8" Floppies | 45.00 | |
| Custom Cables On Request | -Phone- | |
| 30 Pin Prototyping Board | 30.00 | |
| 40 Pin Prototyping Board | 50.00 | |
| CONNECTORS | | |
| Gold Plated 10 Pin Male (Square Posts) or Female | 1.50 | |
| Tin Plated 10 Pin Male (Square Posts) or Female | 1.50 | |

Write or phone for current pricing

**AAA Chicago Computer Center**
**120 Chestnut Lane — Wheeling IL 60090**
**(312) 459-0450**

*Technical Consultation available most weekdays from 4 PM to 6 PM CST*

ELEKTRA is a trademark of AAA Chicago Computer Center
FLEX is a trademark of Technical Systems Consultants, Inc.
OS-9 and BasicO9 are trademarks of Microware Systems Corp.
UniFLEX is a registered trademark of Technical Systems Consultants, Inc.

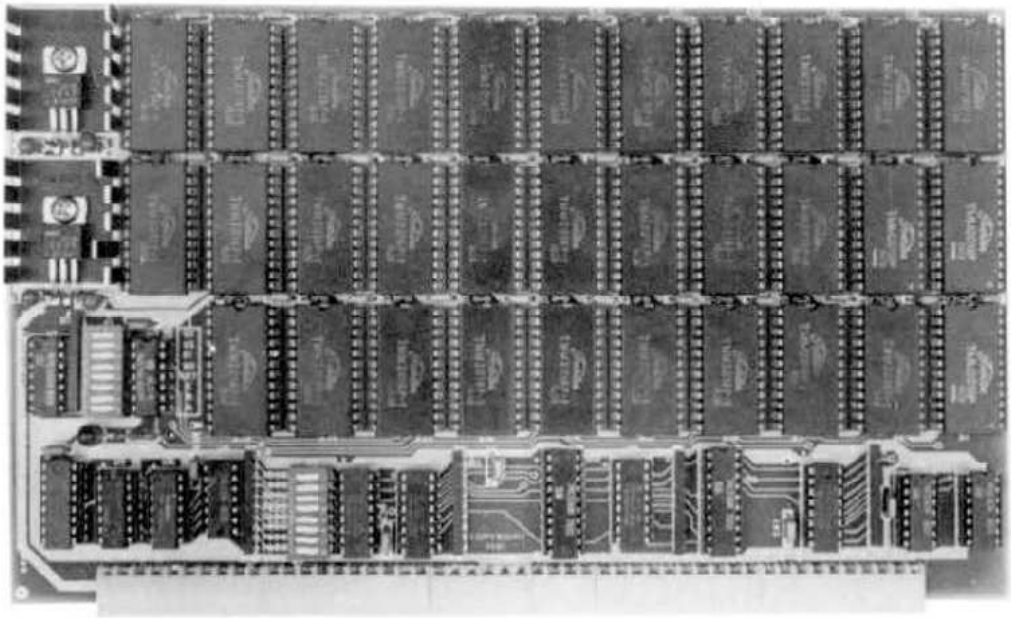
# 64K SS-50 STATIC RAM

PRICE CUT!!

**$119.00** (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:**

| | |
|---|---|
| 56K | $129 |
| 64K | $139 |

- ★ 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.

**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 SOLD FOR $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.

## SOFTWARE FOR 680x SYSTEMS

**SUPER SLEUTH DISASSEMBLERS**
**EACH $99-FLEX $101-OS/9 $100-UNIFLEX**
OBJECT-ONLY versions: EACH $50-FLEX,OS/9,COCO
Interactively generate source on disk with labels, include xref, binary editing
specify 6809,1,2,3,5,8,9/6502 version or Z80/6080,5 version
OS/9 version also processes FLEX format object file under OS/9
COCO DOS available in 6800,1,2,3,5,8,9/6502 version (not Z80/6080,5) only

**CROSS-ASSEMBLERS (TRUE ASSEMBLERS, NOT MACRO SETS)**
**EACH $50-FLEX,OS/9,UNIFLEX ANY 3 $100 ALL $200**
specify for 180x,6502,6801,6804,6805,6809,Z8,Z80,8048,8051,8085,68000
true, modular, free-standing cross-assemblers in C, with load/unload utilities
8-bit (not 68000) sources included with all cross-assemblers (for $200)

**DEBUGGING SIMULATORS FOR POPULAR MICROPROCESSORS**
**EACH $75-FLEX $100-OS/9 $80-UNIFLEX**
OBJECT-ONLY versions: EACH $50-COCO FLEX,COCO OS-9
interactively simulate processors, include disassembly formatting, binary editing
specify for 6800/1, (14)6805, 6502, 6809 OS/9, Z80 FLEX

**ASSEMBLER CODE TRANSLATORS FOR 6502, 6800/1, 6809**
6502 to 6809 $75-FLEX $85-OS/9 $80-UNIFL X
6800/1 to 6809 & 6809 to position-ind $50-FLEX $75-OS/9 $60-UNIFLEX

**FULL-SCREEN XBASIC PROGRAMS with cursor control**
**AVAILABLE FOR FLEX, UNIFLEX, AND MSDOS**

| | |
|---|---|
| DISPLAY GENERATOR/DOCUMENTOR | $50 w/source, $25 without |
| MAILING LIST SYSTEM | $100 w/source, $50 without |
| INVENTORY WITH MRP | $100 w/source, $50 without |
| TABULA RASA SPREADSHEET | $100 w/source, $50 without |

**DISK AND XBASIC UTILITY PROGRAM LIBRARY**
**$50-FLEX $30-UNIFLEX/MSDOS**
edit disk sectors, sort directory, maintain master catalog, do disk sorts,
resequence some or all of BASIC program, xref BASIC program, etc.
non-FLEX versions include sort and resequencer only

**CMODEM TELECOMMUNICATIONS PROGRAM**
**$100-FLEX,OS/9,UNIFLEX**
OBJECT-ONLY versions: EACH $50-FLEX,OS/9
menu-driven with terminal mode, file transfer, MODEM7, XON-XOFF, etc.
for COCO and non-COCO, drives internal COCO modem port up to 2400 Baud

## HARDWARE & SERVICES

**5.25" DISKETTES**
**EACH 10-PACK $12.50-SSSD/SSDD/OSDD $20-DSQD**
American-made, guaranteed 100% quality, with Tyvek jackets, hub rings, and labels

**S8-50C 256K 1.5MHz MEMORY BOARDS**
**EACH BLANK $80 ASSEMBLED AND TESTED $350**
with instruction manual, schematics, and delay line; all parts readily available

**ADDITIONAL SERVICES FOR THE COMPUTING COMMUNITY**
**CUSTOMIZED PROGRAMMING**
we will customize any of the programs described in this advertisement or in our brochure for specialized customer use or to cover new processors; the charge for such customization depends upon the marketability of the modifications
**CONTRACT PROGRAMMING**
we will create new programs or modify existing programs on a contract basis, a service we have provided for over twenty years; the computers on which we have performed contract programming include most popular models of mainframes, including IBM, Burroughs, Univac, Honeywell, most popular models of minicomputers, including DEC, IBM, DG, HP, AT&T, and most popular brands of microcomputers, including 6800/1, 6809, Z80, 8502, 68000, using most appropriate languages and operating systems, on systems ranging in size from large telecommunications to single board controllers; the charge for contract programming is usually by the hour or by the task.
**CONSULTING**
we offer a wide range of business and technical consulting services, including seminars, advice, training, and design, on any topic related to computers; the charge for consulting is normally based upon time, travel, and expenses.

**Computer Systems Consultants, Inc.**
**1454 Latta Lane, Conyers, GA 30207**
**Telephone 404-483-1717 or 4570**

**We take orders at any time, but please plan discussions after 6, if possible.**

**Contact us about catalog, dealer, discounts, and services. Most programs in source: give computer, OS, disk size. 25% off multiple purchases of same program on one order. VISA and MASTER CARD accepted; US funds only, please. Add GA sales tax (if in GA) and 5% shipping.**
FLEX is a trademark of Technical Systems Consultants, OS/9 is a trademark of Microware, COCO is a trademark of Tandy

## SOFTWARE FOR THE HARDCORE

** FORTH PROGRAMMING TOOLS from the 68XX&X **
** FORTH specialists — get the best!! **

NOW AVAILABLE — A variety of rom and disk FORTH systems to run on and/or do TARGET COMPILATION for

6800, 6301/6801, 6809, 68000, 8080, Z80

Write or call for information on a special system to fit your requirement.

Standard systems available for these hardware—

EPSON HX-20 rom system and target compiler
6809 rom systems for SS-50, EXORCISER, STD, ETC.
COLOR COMPUTER
6800/6809 FLEX or EXORCISER disk systems.
68000 rom based systems
68000 CP/M-68K disk systems, MODEL II/12/16

tFORTH is a refined version of FORTH Interest Group standard FORTH, faster than FIG-FORTH. FORTH is both a compiler and an interpreter. It executes orders of magnitudes faster than interpretive BASIC. MORE IMPORTANT, CODE DEVELOPMENT AND TESTING is much, much faster than complied languages such as PASCAL and C. If Software DEVELOPMENT COSTS are an important concern for you, you need FORTH!

firmFORTH℠ is for the programmer who needs to squeeze the most into roms. It is a professional programmer's tool for compact rommable code for controller applications.

* tFORTH and firmFORTH are trademarks of Talbot Microsystems
* FLEX is a trademark of Technical Systems Consultants, Inc.
* CP/M-68K is trademark of Digital Research, Inc.

### tFORTH™ from TALBOT MICROSYSTEMS NEW SYSTEMS FOR 6301/6801, 6809, and 68000

---> tFORTH SYSTEMS <---

For all FLEX systems: GIMIX, SWTP, SSB, or EXORcisor Specify 5 or 8 inch diskette, hardware type, and 6800 or 6809.

** tFORTH — extended fig FORTH (1 disk) $100 ($15)
with fig line editor.
** tFORTH+ — more! (3 5" or 2 8" disks) $250 ($25)
adds screen editor, assembler, extended data types, utilities, games, and debugging aids.
** TRS-80 COLORFORTH — available from The Micro Works
** firm FORTH — 6809 only. $350 ($10)
For target compilations to rommable code.
Automatically deletes unused code. Includes HOST system source and target nucleus source. No royalty on targets. Requires but does not include tFORTH+.
** FORTH PROGRAMMING AIDS — elaborate decompiler $150

** tFORTH for HX-20, in 16K roms for expansion unit or replace BASIC $170

** tFORTH/68K for CP/M-68K 8" disk system $290
Makes Model 16 a super software development system.

** Nautilus Systems Cross Compiler
— Requires: tFORTH + HOST + at least one TARGET:
— HOST system code (6809 or 68000) $200
— TARGET source code: 6800-$200, 6301/6801—$200
same plus HX-20 extensions— $300
6809—$300, 8080/Z80—$200, 68000—$350

Manuals available separately — price in ( ).
Add $6 system for shipping, $15 for foreign air.

TALBOT MICROSYSTEMS 1927 Curtis Ave., Redondo Beach, CA 90278 (213) 376 9941

# WINDRUSH MICRO SYSTEMS

## UPROM II

PROGRAMS and VERIFIES: I2758, T2508, I2716, T2516, I2732/2732A, MCM68764/6, I2764/2764A, T2564, I27128/27128A, and I27256. I=Intel, T=Texas, M=Motorola.

NO PERSONALITY MODULES REQUIRED!

TMS-VOLT EPROMS ARE NOT SUPPORTED

INTEL's intelligent programming (ta) 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 pod 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 1.2).

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 syst .
i. 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-activ environment where you have INSTANT access to the Editor, the Compiler, and the Trace-deDugger, 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/ZOR)
- 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).
- PL/9 supports the MC6809 RESET, NMI, FIRQ, IRQ, SWI, SWI2, and SWI3 vectors. Writing a self-starting (from powerup) 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 'RACE' 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, BITIO, HARDIO, HEXIO, FLEXIO, SCIPACK, STRSUBS, BASTRING, and REALCON.

'... 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 tedius 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 V7 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, optimiser, 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) | $ 90.00 |
| ASM05 | (6809 FLEX only) | $ 90.00 |
| PL/9 | (6809 FLEX only) | $190.00 |
| 'C' | (6809 FLEX only) | $295.00 |
| IEEE-488 | with IEEE-488 cable assembly | $290.00 |
| 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 | S-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 (ta) is a trademark of Technical Systems Consultants, OS-9 (ta) is a trademark of Microware Systems Corporation, MDOS (ta) and EXORciser (tm) are trademarks of Motorola Incorporated.

# '68' MICRO JOURNAL

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Master Charge ☐ — VISA ☐

Card # _____ Exp. Date _____

For ☐ 1-Year ☐ 2 Years ☐ 3 Years

Enclosed: $_____

Name _____

Street _____

City _____ State _____ Zip _____

My Computer Is: _____

**Subscription Rates**
(Effective March 3, 1985)

**U.S.A.: 1 Year $24.50, 2 Years $42.50, 3 Years $64.50**

- **Foreign Surface: Add $12.00 per Year to USA Price.**
- **Foreign Airmail: Add $48.00 per Year to USA Price.**
- **Canada & Mexico: Add $ 9.50 per Year to USA Price.**
- **U.S. Currency Cash or Check Drawn on a USA Bank !!**

**68 Micro Journal**
**5900 Cassandra Smith Rd.**
**Hixson, TN 37343**

☎ **(615)842-4600**

**TELEX 558 414 PVT BTN**

VISA MasterCard

# STAR-DOS LEVEL I

Whenever a new DOS is introduced, there's always the problem of developing software to work with it. So we did it the opposite way — we analyzed the requirements of software that already exists and developed a DOS that met them... and exceeded them! The result is STAR-DOS Level I, a new DOS for 6809 systems, ideal for single-user industrial, control, and advanced hobbyist applications. This includes SS-50 systems and single-board computers from a variety of vendors.

Level I is compatible with most current 6809 hardware and software. On the hardware side, it allows up to ten floppy or Winchester drives with appropriate controllers. On the software side, it runs existing 6809 software from all the major 6809 software suppliers, including TSC, Star-Kits, Introl, and others.

Write or call for more information. STAR-KITS Software Systems Corporation. P.O. Box 209, Mt. Kisco N.Y. 10549 (914) 241-0287.

STAR-KITS

**ANDERSON COMPUTER CONSULTANTS**
**&**
**Associates**

Ron Anderson, respected author and columnist for 68 MICRO JOURNAL announces the **Anderson Computer Consultants & Associates,** a consulting firm dealing primarily in 68XX(X) software design. Our wide experience in designing 6809 based control systems for machine tools is now available on a consultation basis.

Our experience includes programming machine control functions, signal analysis, multi-axis servo control (CNC) and general software design and development. We have extensive experience in instrumentation and analysis of specialized software. We support all popular languages pertaining to the 6809 and other 68XX(X) processors.

If you are a manufacturer of a control or measuring package that you believe could benefit from efficient software, write or call Ron Anderson. The fact that any calculation you can do with pencil and paper, can be done much better with a microcomputer. We will be happy to review your problem and offer a modern, state-of-the-art microcomputer solution. We can do the entire job or work with your software or hardware engineers.

**Anderson Computer Consultants & Associates**
**3540 Sturbridge Court**
**Ann Arbor, MI 48105**

# INDUSTRIAL PASCAL FOR THE 68000

If you're looking for a language to write real-time process control software, look no further. With the rising cost of labor, it is becoming critical that a high level language be used whenever possible. Find out why over 1400 companies have switched to OmegaSoft Pascal for their demanding applications.

OmegaSoft Pascal takes the Pascal framework and expands the basic data types, operators, functions, and memory allocation to fit the needs of real-time systems. These additions fit in the same structure as Pascal and enhance its usefulness without impairing the excellent readability, ease of maintenance, and structured design.

The compiler generates assembly language for assembly and link to run on the target system. Since a true relocating assembler and linking loader is used, only those runtime modules required are automatically linked in, providing a smaller object module than other compilers.

Large Pascal programs can be split up into conveniently sized modules to speed the development process. Procedures, functions, and variables can be referenced between Pascal modules and assembly language modules by using Pascal directives.

The compiler package includes an interactive, symbolic debugger. The debugger allows setting of breakpoints, displaying and changing variables, and tracing statements. Debugging can also be done at the assembly language level when needed. The debugger allows very fast turnaround for programs to be run on the host system (target system debugger coming soon).

The compiler package also includes a full relocatable macro assembler and linking loader. These are designed to support the compiler but may also be used for general assembly language development. In addition, a full screen editor is included which can be used with a variety of intelligent terminals.

Full source code is included for the runtime library, the debugger, the screen editor, and other support utilities.

Versions to run under the OS-9/68000 and VERSAdos operating systems are currently available to end-users and OEM's. End user price is $900 (domestic) or $925 (international). A version for CP/M-68K is available for OEM use, with OEM versions for UNIX type operating systems to follow.

Similar products to run on a 6809 system and generate 6809 code are also available for most major 6809 operating systems.

T.M. OmegaSoft is a trademark of Certified Software Corporation. OS-9/68000 is a trademark of Microware. VERSAdos is a trademark of Motorola. CP/M-68K is a trademark of DRI. UNIX is a trademark of Bell Labs.

**CERTIFIED SOFTWARE CORPORATION**

616 Camino Caballo, Nipomo, CA 93444
Telephone: (805) 349-0202; Telex: 467013

TRS-80+ MOD I, III, COCO, TI99/4a
TIMEX 1000, OSBORNE, others

# GOLD PLUG - 80

Eliminate disk reboots and data loss due to oxidized contacts at the card edge connectors.
**GOLD PLUG 80** solders to the board edge connector. Use your existing cables. (if gold plated)

| | | |
|---|---|---|
| **GOLD PLUG 80** Mod I (6) | $44.95 | ~~$54.95~~ |
| Keyboard/EI (mod I) | 15.95 | ~~18.95~~ |
| Individual connectors | 7.95 | ~~9.95~~ |
| COCO Disk Module (2) | 16.95 | ~~18.95~~ |
| Ground tab extensions | INCL | ~~1.00~~ |
| Disk Drives (all R.S.) | 7.95 | ~~9.95~~ |
| Gold Disk Cable 2 Drive | | 29.95 |
| Four Drive Cable | | 39.95 |
| **GOLD PLUG 80** Mod III (6) | | 54.95 |
| Internal 2 Drive Cable | | 29.95 |
| Mod III Expansion port | | 10.95 |

SPECIAL PRICES

COCO MODULE INSTALLATION AVAILABLE

USA shipping $1.45 — Can/Mex $4.
Foreign $7. — TEXAS 5% TAX

Ask your favorite dealer or order direct

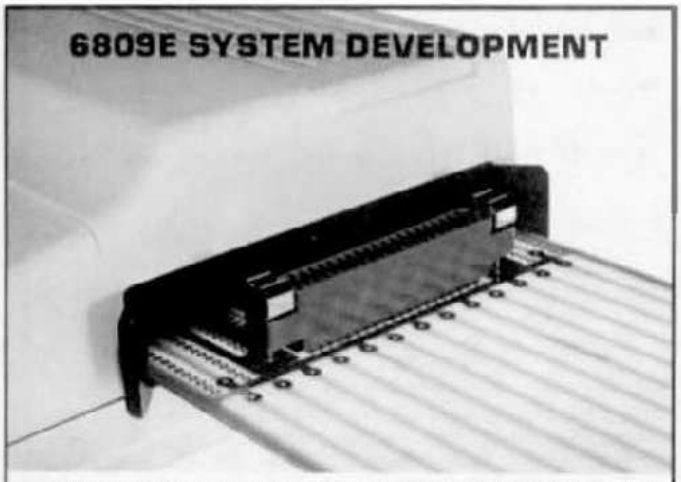
ORDER TODAY!

**E.A.P. CO.**
P.O. BOX 14
KELLER, TEXAS 76248

(817) 498-4242 — MC/VISA

+ trademark Tandy Corp

## 6809E SYSTEM DEVELOPMENT



The XPNOR2 project card extends the CoCo cartridge port signals to wirewrap pins on the bottom of a 40 pin disk controller/ROM pak connector. In one efficient package you have over 24 square inches of drilled card for interface or prototype circuits plus the full on-line power of any software development languages or tools available for the CoCo. Made with all the right stuff. Assembled, tested and ready to run. Shown above with our cartridge port card guide.

XPNOR2 $39.95 each or 2/$76
SuperGuide $ 3.95 each

To order, or for technical information, call (206) 782-6809, weekdays 8 am to noon. We pay shipping on prepaid orders. For immediate shipment send check, money order or the number and expiration date of your VISA or MASTERCARD to:

ROBOTIC MICROSYSTEMS

BOX 30807 SEATTLE, WA 98103

# LLOYD I/O™

*Computer Engineers*

**19535 NE GLISAN • PORTLAND, OR 97230 (USA)**
**PHONE: (503) 666-1097 • TELEX: 910 380 5448 LLOYD I O**

## K-BASIC™ *IS HERE*

**K-BASIC is a TSC XBASIC (XPC) compatible COMPILER**
*for OS9 & FLEX... price $199*

Here at last is a compiler for BASIC that will compile all your XBASIC programs. K-BASIC compiles TSC's XBASIC and XPC programs to machine code. K-BASIC is ready now to save you money and time by teaching your computer to:
• Think Faster • Conserve Memory • Be Friendlier

**Call (503) 666-1097 for our CATALOG.**
**We have many programs for serious software developers!**

## DO™

*Micro BASIC for OS9...$149*

A structured mirco BASIC for general system control featuring: Parameter passing, 10 string variables, 26 numeric variables, subroutines, nested loops, interactive I/O, sequential files, and time variables (for applications executing in the background required to execute procedures such as disk or file backups.) Includes the SEARCH and RESCUE UTILITIES™. [For OS9 ONLY.]

## SEARCH and RESCUE UTILITIES™

*for OS9...$35*

A super directory search utility. Output may be p'ped to the included utilities to perform file: COPIES, DELETES, MOVES, LISTING (pagination), and FILTERING. Some filtering utility programs are included, of interest is the FILE DATE CHECKING utilities: YOUNGER and DRAFT (Level 2). (For OS9 Level 1 and 2)

## PATCH™

*Modem Communications for OS9...$39*

PATCH is a modem communications program for OS9 featuring: KEY MACROS, ASCII TEXT AND BINARY FILE UP/DOWN LOADING, PRINTER COPY, and HELP MENUS. We use it several times each day with our TELEX service. PATCH is convenient and easy to use. Key macros may be pre-stored and loaded at any time.

## CRASMB™

**CROSS ASSEMBLER PACKAGE**
*for OS9 & FLEX...all for $399*
*Motorola CPU's...$150*
*Intel CPU's...$150, Others...$150*

CRASMB is the highly acclaimed cross assembler package for OS9 and FLEX systems. It turns your 6809 computer into a development station for these target CPUs:

6800 6801 6804 6805 6809 6811 6502
7000 1802 8048 8051 8080 8085 Z8 Z80

(68000 16/32 bit cross assembler...$249)

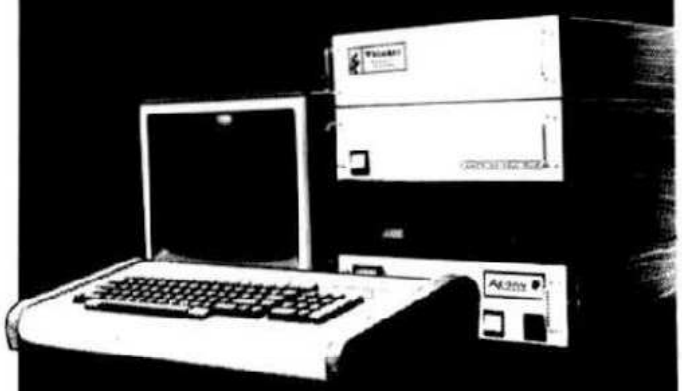
CRASMB features: Macros, Conditionals, Long symbol names, Symbol cross reference tables, Object code in 4 formats (OS9, FLEX, S-1-S9, INTEL HEX).

**VISA, MC, BOD, CHECKS ACCEPTED**

| | | |
|---|---|---|
| USA: | LLOYD I/O (503 666 1097), | S.E. MEDIA (800 338 6800) |
| England: | Vivaway (0582 423425), | Windrush (0692 405489) |
| Germany: | Zacher Computer (65 25 299), | Keil Software (06203 6741) |
| Australia: | Paris Radio Electronics (344 9111) | |

*K-BASIC, DO, SEARCH and RESCUE UTILITIES, PATCH, CRASMB and CRASMB 16/32 are trademarks of LLOYD I/O*
*OS9 is a ™ of Motorola, FLEX is a ™ of TSC*

# ACORN
COMPUTER SYSTEMS SS-50C



MODULES - BARE CARDS - KITS - ASSEMBLED & TESTED

| Stackable Modules | | KIT | A&T |
|---|---|---|---|
| 20 amp POWER SUPPLY w/fan w/Disk protect relay | | 350.00 | 400.00 |
| DISK CABINET w/rege. & cables less DRIVES | | 200.00 | 250.00 |
| MOTHER BOARD, 8 SS-50c, 8 SS-30c NMI button | | 225.00 | 335.00 |
| **Item** | **Bare** | **KIT** | **A&T** |
| IT3 - INTERRUPT TIMER 1, 10, 100 per sec. | 19.95 | 29.95 | 39.95 |
| PB4 - INTELLIGENT PORT BUFFER Single board comput. | 39.95 | 114.95 | 139.95 |
| DPIA - Dual PIA parallel port, 4 buffered I/Os | 24.95 | 69.95 | 89.95 |
| XADR - Extended Addressing BAUD gen. PIA port | 29.95 | 89.95 | 89.95 |
| MB6 - MOTHER BOARD SS-50c w/BAUD gen. | 64.95 | 149.95 | 199.95 |
| P16K - 16K PROM DISK 21, 2764 EPROMs | 39.95 | 79.95 | 109.95 |
| FD8S - Firmware development 2, 8K blocks | 39.95 | 84.95 | 114.95 |
| XMPR - 2764 PROM burner adapt. for 2716 BURNER | | 19.95 | ------ |
| CHERRY Keyboard w/Cabinet 96 key capacitive | | 249.95 | ------ |
| TAXAN 12", 16 Mhz MONITOR GREEN | | ------ | 149.95 |
| AMBER | | ------ | 159.95 |
| 4 MODULE CABINET - unfinished | | 150.00 | ------ |
| POWER SUPPLY w/disk protect | | 250.00 | ------ |

# Color Computer

| | | | |
|---|---|---|---|
| SOROLINK - 20 Mhz Monochrome video driver | | 15.00 | 20.00 |
| CC30 PORT BUS w/power supply 5 SS-30, 2 Cart | | 169.95 | 199.95 |
| POWER BOX 6 switched outlets transient suppression | | 39.95 | 39.95 |
| RS-232 3-switched ports for above | ADD | +20.00 | +25.00 |

Write for FREE Catalog
ADD $3.00 S&H PER ORDER
WIS. ADD 5% SALES TAX

ACS INC
11931 W. Bluemound Road
MILWAUKEE, WIS. 53226
(414) 257-0300


# 68' MICRO JOURNAL

**Disk-1** Filesort, Minicat, Minicopy, Minifms, **Lifetime, **Poetry, **Foodlist, **Diet.

**Disk-2** Diskedit w/ inst.& fixes, Prime, *Prmod, **Snoopy, **Football, **Hexpawn, **Lifetime

**Disk-3** Cbug09, Sec1, Sec2, Find, Table2, Intext, Disk-exp, *Diskasve.

**Disk-4** Mailing Program, *Finddat, *Change, *Testdisk.

**Disk-5** *DISKFIX 1, *DISKFIX 2, **LETTER, **LOVESIGN, **BLACKJAK, **BOWLING.

**Disk-6** **Purchase Order, Index (Disk file indx)

**Disk-7** Linking Loader, Rload, Harkness

**Disk-8** Crteat, Lanpher (May 82)

**Disk-9** Datecopy, Diskfix9 (Aug 82)

**Disk-10** Home Accounting (July 82)

**Disk-11** Disassembler (June 84)

**Disk-12** Modem68 (May 84)

**Disk-13** *Initmf68, Testmf68, *Cleanup, *Dskalign, Help, Date.Txt

**Disk-14** *Init, *Test, *Terminal, *Find, *Diskedit, Init.Lib

**Disk-15** Modem9 + Updates (Dec. 84 Gilchrist) to Modem9 (April 84 Combo)

**Disk-16** Copy.Txt, Copy.Doc, Cat.Txt, Cat.Doc

**Disk-17** Match Utility, RATBAS (A Basic Preprocessor) June 85

**Disk-18** Parse.Mod, Size.Cmd (Sept. 85 Armstrong), CMDCODE, CMD.Txt (Sept. 85 Spray)

**Disk-19** Clock, Date, Copy, Cat, PDEL.Asm & Doc., Errors.Sys, Do, Log.Asm & Doc.

**Disk-20** UNIX Like Tools (July & Sept. 85 Taylor & Gilchrist). Dragon.C, Grep.C, LS.C, PDUMP.C

**NOTE:**
This is a reader service **ONLY!** No Warranty is offered or implied, they are as **received** by "**68' Micro Journal**, and are for reader convenience ONLY (some MAY include fixes or patches). Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other.

PRICE: 8" Disk **$14.95** - 5" Disk **$12.95**

**68' Micro Journal**
5900 Cassandra Smith Rd.
Hixson, Tn. 37343
**(615)842-4600**

* Indicates 6800
** Indicates BASIC SWTPC or TSC 6809 no Indicator.

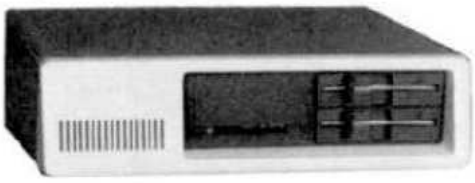
**MASTER CARD - VISA Accepted**
Foreign -- add **10%** for Surface
or **20%** for Airl

# PT-69 SINGLE BOARD COMPUTER SYSTEMS

## NOW WITH WINCHESTER OR FLOPPY DISK DRIVES

The proven PT-69 Single Board Computer line is expanding! Systems now can be Winchester or floppy-based. Available also in a smaller cabinet without drives for dedicated systems with no mass storage requirements.

* 1 MHZ 6809E Processor
* Time-of-Day Clock
* 2 RS 232 Serial Ports (6850)
* 56K RAM 2K/4K EPROM
* 2 8-bit Parallel Ports (6821)
* 2797 Floppy Disk Controller



Winchester System

*Custom Design Inquiries Welcome*

Floppy System

•PT69XT WINCHESTER SYSTEM — $1795.95
Includes 5 MF6 Winchester Drive, 2 40-track DS/DD Drives, Parallel Printer Interface • choice of OS/9 or STAR-DOS

•PT6952 FLOPPY SYSTEM — $895.95
Includes PT69 Board, 2 DS/DD 40-TRK 5 1/4" drives, cabinet, switching power supply, OS/9 or STAR-DOS

•PT-69A ASSEMBLED & TESTED BOARD $279.00
•OS/9 $200.00
•STAR-DOS $ 50.00

**PERIPHERAL TECHNOLOGY**

1480 Terrell Mill Rd., Suite 870
Marietta, Georgia 30067

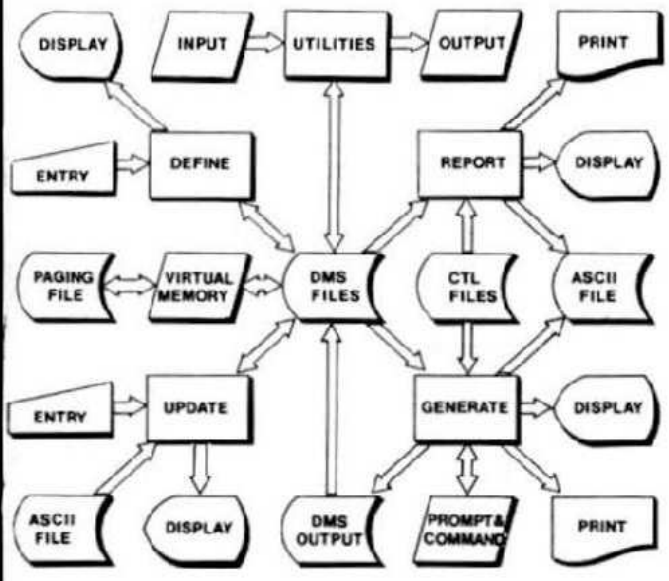
VISA/MASTERCARD/CHECK/COD — Telex #880584 — 404/973-0042 — CALL OR WRITE FOR ADDITIONAL CONFIGURATIONS

"OS/9 is a trademark of Microware and Motorola



# XDMS

## Data Management System




System Architecture

WESTCHESTER Applied Business Systems
Post Office Box 187
Briarcliff Manor, N.Y. 10510

### XDMS Data Management System

The XDMS Data Management System is available in three levels. Each level includes the XDMS nucleus, VMODE utility and System Documentation for level III. XDMS is one of the most powerful systems available for 6809 computers and may be used for a wide variety of applications. XDMS users are registered in our database to permit distribution of product announcements and validation of user upgrades and maintenance requests.

### XDMS Level I

XDMS Level I consists of DEFINE, UPDATE and REPORT facilities. This level is intended as an "entry level" system and permits entry and reporting of data on a "tabular" basis. The REPORT facility supports record and field selection, field merge, sorting, line calculations, column totals and report titling. Control is via a English-like language which is upward compatible with level II. XDMS Level I . . . . . $129.95

### XDMS Level II

Level II adds to Level I the powerful GENERATE facility. This facility can be thought of as a general file processor which can produce reports, forms and form letters as well as file output which may be re-input to the facility. GENERATE may be used in complex processing applications and is controlled by a English-like command language which encompasses that used by Level I. XDMS Level II . . . . . $199.95

### XDMS Level III

Level III includes all of level II plus a set of useful DMS Utilities. These utilities are designed to aid in the development and maintenance of user applications and permit modification of XDMS system parameters, input and output of XDMS files, display and modification of file format, graffitic display of numerical data and other functions. Level III is intended for advanced XDMS users. XDMS Level III . . . . . $269.95
XDMS System Documentation only ($10, credit toward purchase) . . $ 24.95

### XACC Accounting System

The XACC General Accounting System is designed for small business environments of up to 10,000 accounts and inventory items. The system integrates accounting functions and inventory plus the general ledger, accounts receivable and payable functions normally sold separately in other systems. Features user defined accounts, products (or services), transactions, invoicing, etc. Easily configured to most environments. XACC General Accounting System (Requires XDMS, prof. Lv. III) . $299.95
XACC System Documentation only ($10, credit toward purchase) . . $ 24.95

WESTCHESTER Applied Business Systems
Post Office Box 187, Briarcliff Manor, N.Y. 10510

All software is written in macro/assembler and runs under 6809 FLEX O/S.
Terms: Check, Money Order, Visa or Mastercharge. Shipment first class. Add P&H $2.50 ($7.50 Foreign Surface or $15.00 Foreign Air). NY Res add sales tax. Specify 5" or 8".

Sales: S. E. [redacted], (813) 842-4469, Consultation: 914-941-3552 (evens)

## ALL SYSTEMS INCLUDE:

* The CLASSY CHASSIS with a ferro-resonant, constant voltage power supply that provides + 8 volts at 30 Amps, + 16 volts at 5 Amps, and − 16 volts at 5 Amps.
* Gold plated bus connectors.
* Double density DMA floppy disk controllers.
* Complete hardware and software documentation.
* Necessary cables, filler plates.

## YOU CAN EXPAND YOUR SYSTEM WITH:

### MASS STORAGE

Dual 8" DSDD Floppies, Cabinet & Power Supply ..... **$1698.88**
20MB Streamer ..... (under development)
1.6MB Dual Speed Floppy ..... (under development)

### MEMORY

#67 Static RAM-64K NMOS (6809 Only) ..... **$349.57**
#64 Static RAM-64K CMOS w/battery (6809 Only) ..... **$398.64**
#72 256K CMOS Static RAM w/battery ..... **$998.72**
#31 16 Socket PROM/ROM/RAM Board (6809 only) ..... **$268.31**

### INTELLIGENT I/O PROCESSOR BOARDS

significantly reduce systems 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. For use with GMX III and 020 systems.

#113 Port Serial-30 Pin (OS9) ..... **$498.11**
#14 3 Port Serial-30 Pin (UniFLEX) ..... **$498.14**
#12 Parallel-50 Pin (UniFLEX-020) ..... **$538.12**
#13 4 Port Serial-50 Pin (OS9 & UniFLEX-020) ..... **$818.13**

### I/O BOARDS (6809 SYSTEMS ONLY)

#41 Serial, 1 Port ..... **$88.41**
#43 Serial, 2 Port ..... **$128.43**
#46 Serial, 8 Port (OS9/FLEX only) ..... **$318.46**
#42 Parallel, 2 Port ..... **$88.42**
#44 Parallel, 2 Port (Centronics pinout) ..... **$128.44**
#45 Parallel, 8 Port (OS9/FLEX only) ..... **$198.45**

### CABLES FOR I/O BOARDS—SPECIFY BOARD

#95 Cable sets (1 needed per port) ..... **$24.95**
#51 Cent. B.P. Cable for #12 & #44 ..... **$34.51**
#53 Cent. Cable Set ..... **$36.53**

### OTHER BOARDS

#66 Prototyping Board-50 Pin ..... **$56.66**
#33 Prototyping Board-30 Pin ..... **$38.33**
Windrush EPROM Programmer S30 (OS9/FLEX 6809 only) ..... **$545.00**

*CONTACT GIMIX FOR FURTHER DETAILS ON THESE AND OTHER BOARDS AND OPTIONS.*

EXPORT MODELS: ADD $30 FOR 50Hz. POWER SUPPLIES.
ALL PRICES ARE F.O.B. CHICAGO.

GIMIX DOES NOT GUARANTEE PERFORMANCE OF ANY GIMIX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

GIMIX, Inc. reserves the right to change pricing, terms, and products specifications at any time without further notice.

## GIMIX 2MHZ 6809 SYSTEMS

## GMX 68020 SYSTEMS

| Operating Systems Included | #49 OS9 GMX I/ and FLEX | #39 OS9 GMX II/ and FLEX | #79 OS9 GMX III/ and FLEX | #39 UniFLEX | #89 UniFLEX III | #020 OS9/68020 | #020 UniFLEX VM |
|---|---|---|---|---|---|---|---|
| CPU Included | #05 | #05 | GMX III | #05 | GMX III | GMX 020 | GMX 020 + MMU |
| Serial Ports Included | 2 | 2 | 3 Intelligent | 2 | 3 Intelligent | 3 Intelligent | 3 Intelligent |
| High Speed Static RAM | 64KB | 256KB | 256KB | 256KB | 256KB | 512KB | 1 Megabyte |
| **PRICES OF SYSTEMS WITH:** Dual 80 Track DSDD Drives | $2998.49 | $3398.39 | $4898.79 | N/A | N/A | N/A | N/A |
| 19MB Hard Disk and one 80 track Floppy Disk | $5598.49 | $5998.39 | $7798.79 | $5998.39 | $8098.89 | $11,680.20 | $13,680.20 |
| 72 MB Hard Disk and one 80 track | $7598.49 | $7998.39 | $9798.79 | $7998.39 | $10,098.89 | $13,680.20 | $15,680.20 |
| a 72MB + a 6MB removable pack hard disk and one 80 track floppy | $9098.49 | $9498.30 | N/A | $9498.39 | N/A | N/A | N/A |
| a 72MB + a 12MB removable pack hard disk and one 80 track floppy | N/A | N/A | $11,298.79 | N/A | $11,598.89 | $15,180.20 | $17,180.20 |
| **GMX 6809 OS9/ FLEX SYSTEMS SOFTWARE** OS9 + Editor, Assembler, Debugger | GMX I Included | GMX II Included | GMX III Included | | | | |
| FLEX | Included | Included | Included | | | | |
| GMXBUG Monitor | Included | Included | Included | | | | |
| Basic 09, Run8 (OS9) | Included | Included | Included | | | | |
| RMS (OS9) | Included | Included | Included | | | | |
| DO (OS9) | Included | Included | Included | | | | |
| VDisk for FLEX | N/A | Included | Included | | | | |
| RAMDisk for OS9 | N/A | $125 option | Included | | | | |
| O-FLEX | N/A | $250 option | Included | | | | |
| Support ROM | N/A | N/A | Included | | | | |
| Hardware CRC | N/A | N/A | Included | | | | |

Available: Wide variety of languages and other software for use with either OS-9 or FLEX.

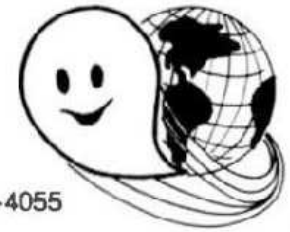
All GIMIX versions of OS9 can read and write RS color computer format OS9 disks, as well as the Microware/GIMIX standard format.

All OS9/FLEX systems allow you to software select either operating system.

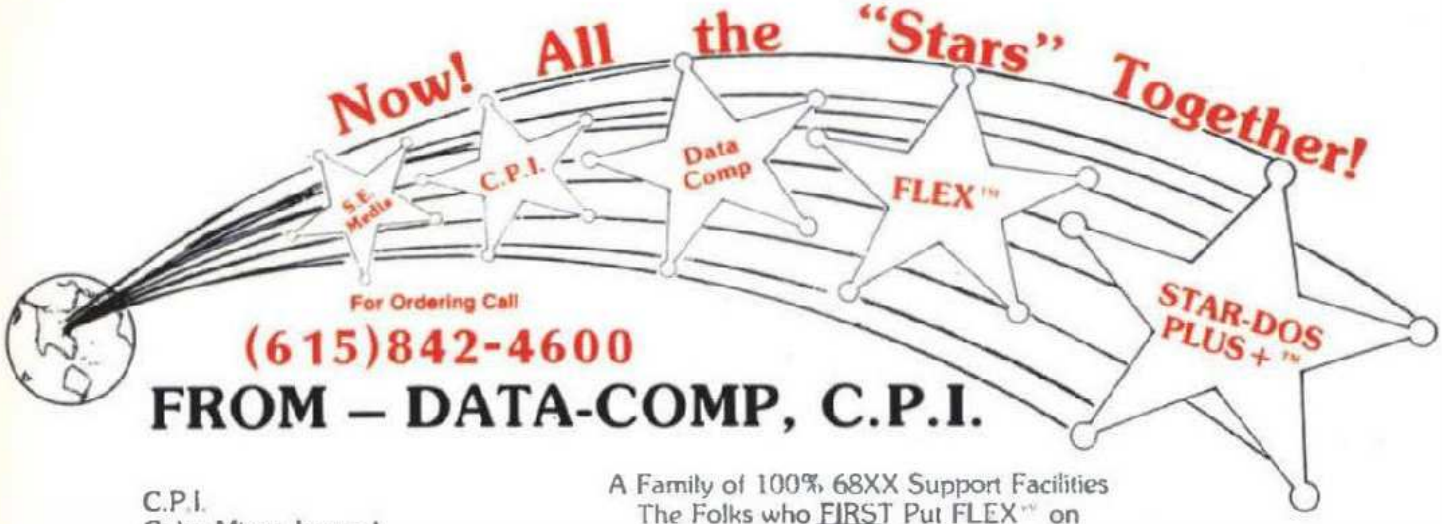
TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air 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 number 73-32033.

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



©1985 GIMIX, INC. 8-85




For Ordering Call

**(615)842-4600**

# 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
+ Many Many More!!!
+ External Terminal Program
+ Test Disk Program
+ Disk Examine & Repair Program
+ Memory Examine Program

TSC Editor
Reg $50.00
**NOW $35.00**

TSC Assembler
Reg $50.00
**NOW $35.00**

## CoCo Disk Drive Systems

NEW LOWER PRICES ON PAK #5, AND PRINTERS

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. | $349.95 |
| 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 | $499.95 |

## Controllers

| | |
|---|---|
| J&M DISK CONTROLLER W/ JDOS OR RADIO SHACK DISK BASIC, SPECIFY WHAT DISK BASIC. | $134.95 |
| RADIO SHACK DISK CONTROLLER 1.1 | $134.95 |

## Misc.

| | |
|---|---|
| 64K UPGRADE W/MOD. INSTRUCTIONS, C,O,E,F, AND COCO 2 | $ 44.95 |
| HJL KEYBOARDS | $ 74.95 |
| MICRO TECH LOWER CASE ROM ADAPTER | $ 74.95 |
| RADIO SHACK BASIC 1.2 | $ 24.95 |
| RADIO SHACK DISK BASIC 1.1 | $ 24.95 |
| 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 | $269.00 |
| EPSON RX-80FT | $369.00 |
| EPSON MX-100 | $499.00 |
| EPSON FX-100 | $799.00 |
| EPSON FX-80 | $549.00 |
| EPSON MX-70 | $200.00 |

## Disk Drive Cables

| | |
|---|---|
| CABLE FOR ONE DRIVE | $ 19.95 |
| CABLE FOR TWO DRIVES | $ 24.95 |

**DATA-COMP**
5900 Cassandra Smith Rd.
Hixson, TN 37343

MC VISA

SHIPPING
USA ADD 2%
FOREIGN ADD 5%
MIN. $2.50

(615)842-4600
For Ordering
TELEX 558 414 PVT BTH

68 Micro Journal
ISSN 0194-5025
5900 Cassandra Smith Rd.
Hixson, TN 37343

Second Class Postage Paid
At Hixson, TN
And Additional Mailing Offices


# Introducing

## S-50 BUS/68XX

**Board and/or Computer**

**Terminals-CRTs-Printers**

**Disk Drives-etc.**

## REPAIRS

**This**

**Not This**

**NOW AVAILABLE TO ALL S50/68XX USERS**

The Data-Comp Division of CPI is proud to announce the availability of their service department facilities to **'ALL'** S50 Bus and 68XX users. Including all brands, SWTPC - GIMIX - SS8 - HELIX and others, including the single board computers. * Please note that kit-built components are a special case, and will be handled on an individual basis, if accepted.

**1.** If you require service, the first thing you need to do is call the number below and describe your problem and **confirm a Data-Comp service & shipping number!** This is very important, Data-Comp will not accept or repair items not displaying this number! Also we cannot advise or help you troubleshoot on the telephone, we can give you a shipping number, but **NO** advice! Sorry!

**2.** All service shipments must include both a minimum $40.00 estimate/repair charge and pre-paid return shipping charges (should be same amount you pay to ship to Data-Comp).

**3.** If you desire a telephone estimate after your repair item is received, include an additional $5.00 to cover long distance charges. Otherwise an estimate will be mailed to you, if you requested an estimate. Estimates **must be** requested. Mailed estimates slow down the process considerably. However, if repairs are not desired, after the estimate is given, the $40.00 shall constitute the estimate charge, and the item(s) will be returned unrepaired providing sufficient return shipping charges were included with item to be serviced. Please note that estimates are given in dollar amounts only.

**4.** Data-Comp service is the oldest and most experienced general S50/68XX service department in the world. We have over $100,000.00 in parts in stock. We have the most complete set of service documents for the various S50/68XX systems of anyone - **YET, WE DO NOT HAVE EVERYTHING!** But we sure have more than anyone else. We repair about 90% of all items we receive. Call for additional information or shipping instructions.

**DATA-COMP**
5900 Cassandra Smith Rd.
Hixson, TN 37343

☎ **(615)842-4607**
Telex 558 414 PVT BTR

VISA