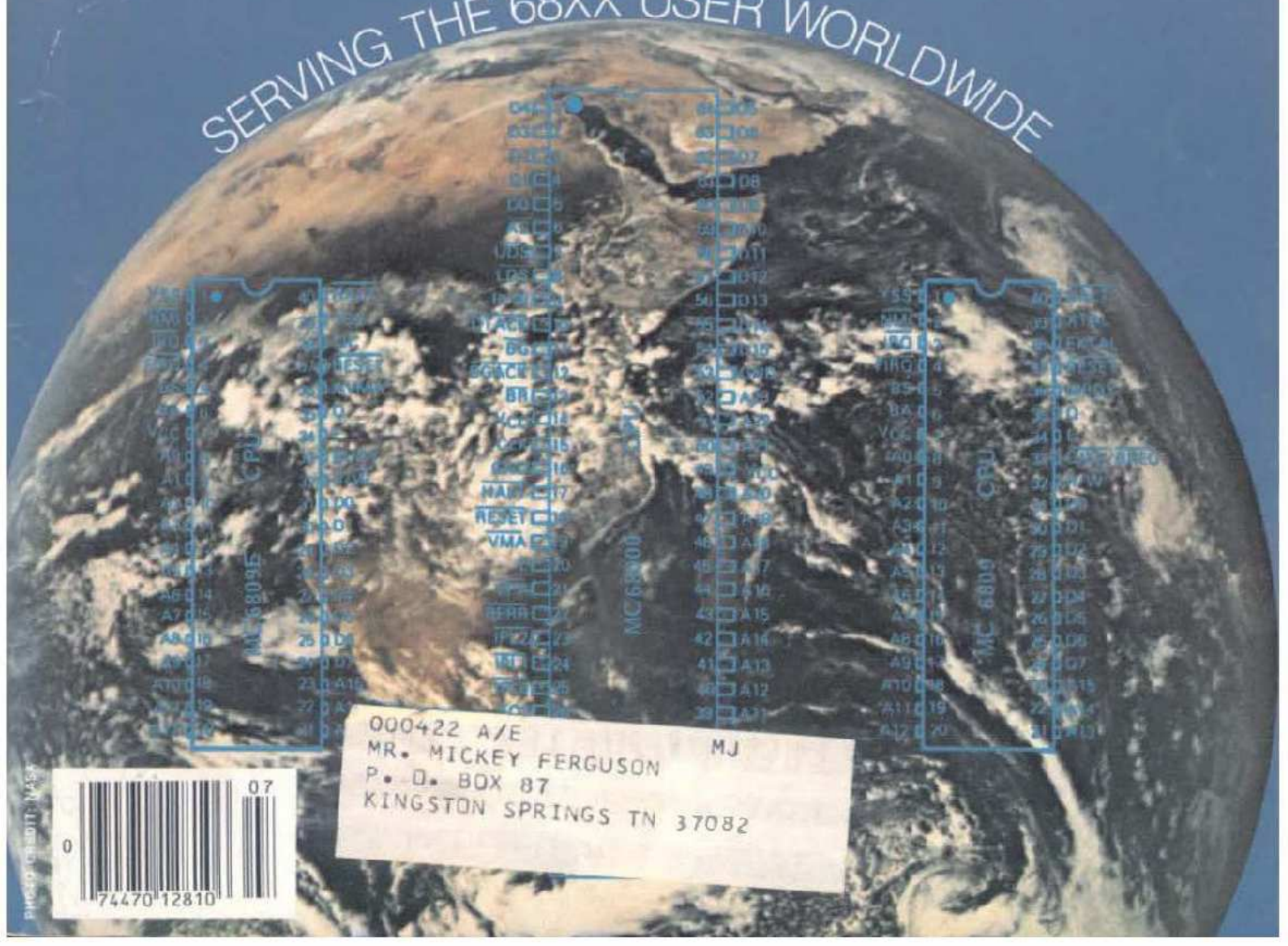
'68' MICRO JOURNAL

$2.95USA

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

VOLUME VI ISSUE VII • Devoted to the 68XX User • July 1984

"Small Computers Doing Big Things"



000422 A/E MJ
MR. MICKEY FERGUSON
P. O. BOX 87
KINGSTON SPRINGS TN 37082

0 74470 12810 07

WE DON'T PLAY GAMES

X-12+ A SERIOUS COMPUTER IN A DESKTOP PACKAGE

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

1.0 Megabyte Memory - Insures no limitation on programs

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

UniFlex* Operating System - The standard of comparison

Hardware Floating Point - Unmatched speed in a small system

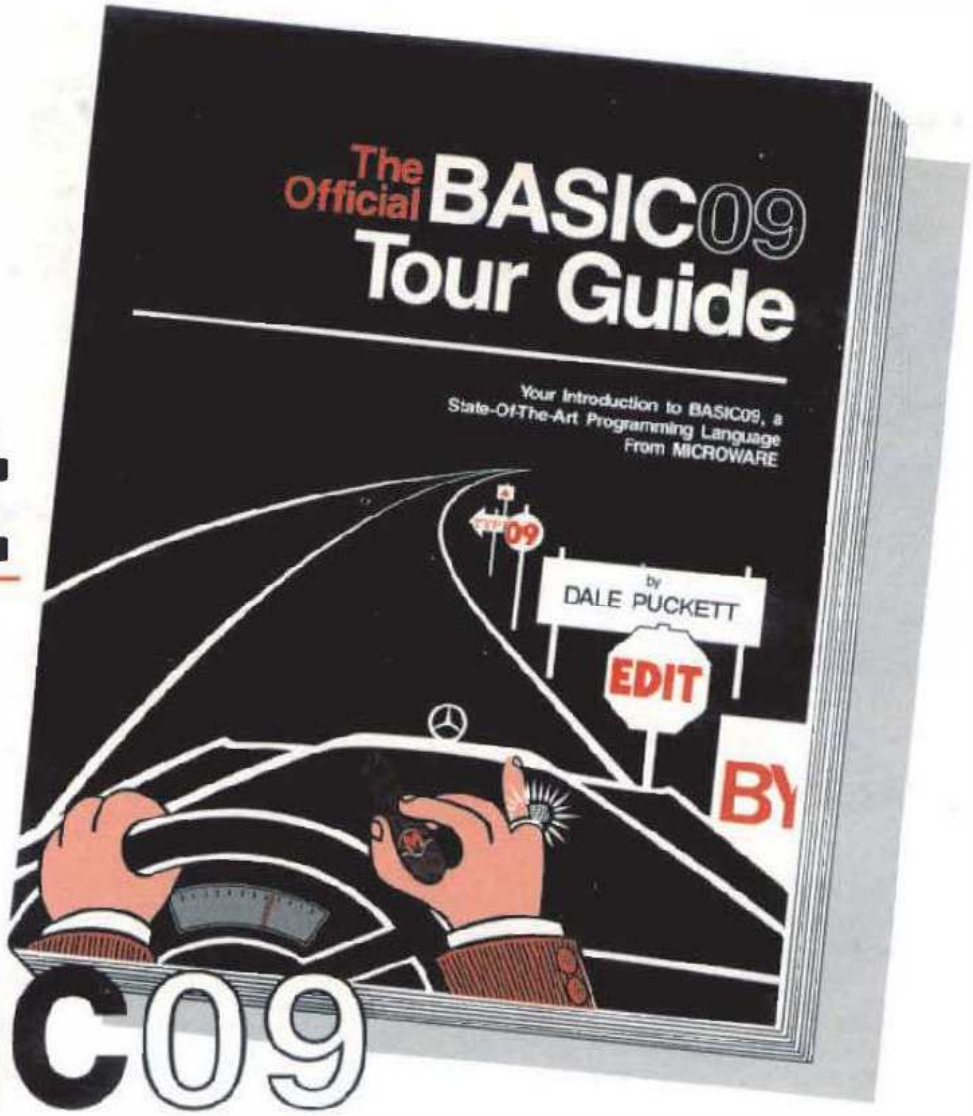
Up to Three Terminals - Instant expansion

*Trademark of Technical Systems Consultants

SWTPC

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

Get the most out of BASIC09



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

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

MICROWARE . . .

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

It's Easy to Order.

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

microware®

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

OS-9 and BASIC09 are trademarks of Microware and Motorola.

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

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

Name _____

Address _____

City _____

State _____ Zip _____

☐ I have enclosed a check

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

'68' MICRO JOURNAL

Portions of the text for 68 MICRO JOURNAL was prepared using the following furnished hard/software.

COMPUTERS-HARDWARE
Southwest Technical Products
219 W. Rhapsody
San Antonio, Texas 78216
S09-5/8 DMF disk-CDS1-8212W-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-Processor

Great Plains Computer Company, Inc.
PO Box 916
Idaho Falls, ID 83401
STYLO-Mail Merge

Editorial Staff

Don Williams Sr.	Publisher
Larry E. Williams	Executive Editor
Tom E. Williams	Production Editor
Robert (Bob) Nay	Color Editor

Administrative Staff

Mary Robertson	Office Manager
Penny Williams	Subscriptions
Michael Westfall	Shipping/Rec.
Christine Kocher	Accounting

Contributing Editors

Ron Anderson
Norm Commo
Peter Dibble
Dr. Theo Elbert
William E. Fisher
Dr. E.M. Pass

Special Technical Projects

Clay Abrams K6AEP
Tom Hunt

CONTENTS

Vol.VI,Issue VII July'84

Rumors and Such	8	OMW
FLEX USER Notes	8	Anderson
OS9 USER Notes	11	Dibble
C USER Notes	14	Pass
68000 User Notes	16	Lucido
Windrush IEEE-488 Review	18	McDaniel
VDisk Review	20	Pass
Disassembler	21	Stock
Reading Non-Flex Disks	31	Fraser
Bit Bucket	42	
Metalab Review	43	Nunn
ELEKTRA Motherboard Review	44	Sims
Corrections	48	
Help	48	
Classifieds	49	

Send All Correspondence To:

Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343
615 842-4600

Copyrighted 1984 by
Computer Publishing Inc. (CPI)

68' Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage Paid ISSN 0194-5025 at Hixson, Tenn. and additional entries. Postmaster: send Form 3579 to 68' Micro Journal, PO Box 849, Hixson, Tennessee.

SUBSCRIPTION RATES
USA
1-Year $24.50 2-Years $42.50 3-Years $64.50
FOREIGN
See Page 60

Items Submitted for Publication

Articles submitted for publication should be accompanied by the authors **full name, address, date and telephone number.** It is preferred that articles be submitted on either 5 or 8 inch diskette in TSC Editor format or STYLO format. All diskettes will be returned.

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

STYLO commands are all acceptable except the .pg page command, we print edited text files in continous text.

All articles submitted on diskettes should be in TSC FLEX™ format, either FLEX2 6800, or FLEX9 6809 any version.

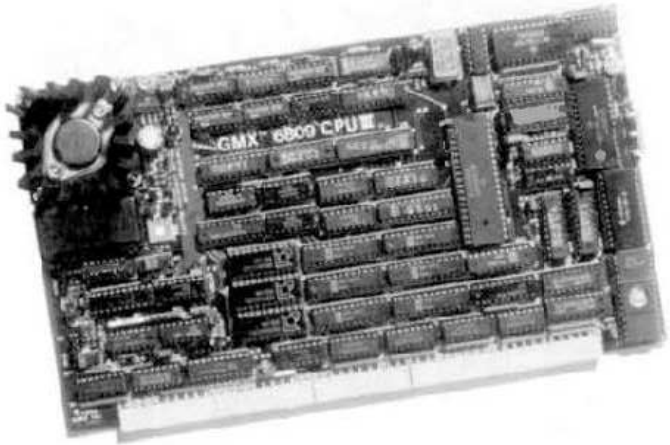
If articles are submitted on paper they should be on white 8X11 bond or better grade paper. No hand written articles (hand written or drawn art accepted). All paper submitted articles will be photo reproduced. This requires that they be typed or produced with a dark ribbon (no blue), single spaced and type font no smaller than 'elite' or 12 pitch. Typed text should be approximately 7 inches wide (will be reduced to column width of 3 1/2 inches). **Please use a dark ribbon!**

All letters to the editor should also comply with the above and bear a signature. Letters of 'gripes' as well as 'praise' are solicited. We attempt to publish all letters to the editor verbatim, however, we reserve the right to reject any submission for lack of 'good taste'. We reserve the right to define what constitutes 'good taste'.

Advertising: Commercial advertisers please contact the 68 Micro Journal advertising department for current rate sheet and requirements.

Classified: All classified must be non-commercial. Maximum 20 words per classified ad. Those consisting of more than 20 words should be figured at .35 cents per word. 20 words or less $7.50 minimum, one time, paid in advance. No classified ads accepted by telephone.

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



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

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

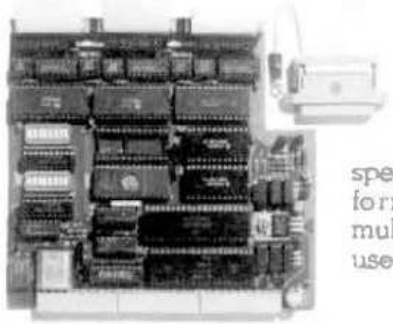
GIMIX has single user systems that can run both FLEX and OS-9 or Multi user systems for use with UniFLEX or OS-9.

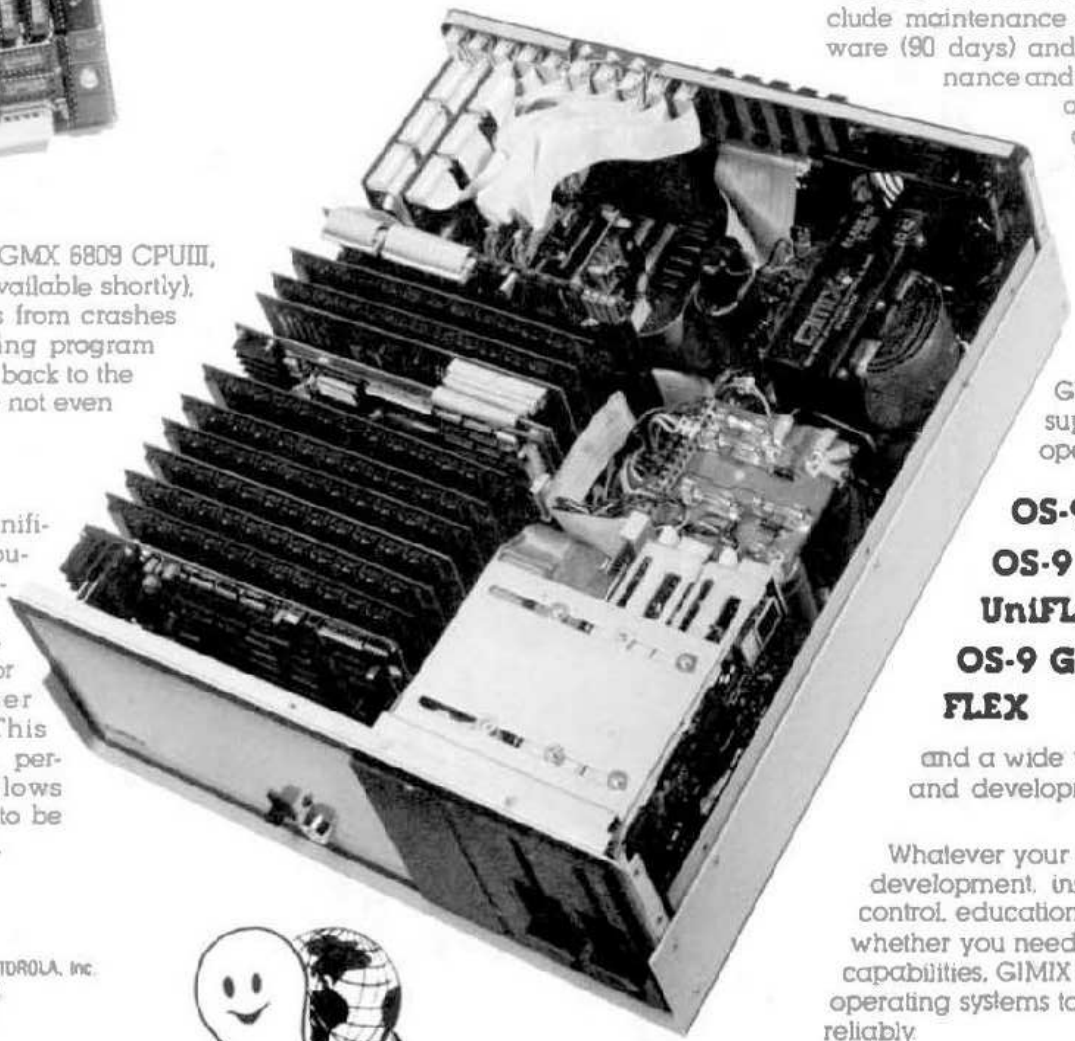
GIMIX versions of OS9 and UniFLEX include maintenance and support by Microware (90 days) and TSC (1 year). Maintenance and support after this period are available at extra cost.

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

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

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





GIMIX 6809 systems support five predominant operating systems:

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

and a wide variety of languages and development software.

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

Please phone or write if you need further information.

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

GIMIX INC.

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

©1983 GIMIX Inc.

Microware presents 4 new OS-9 software packages.

1 LEVEL II PRINT SPOOLING SYSTEM

This versatile package gives your OS-9 Level Two System a complete print spooling management capability for time-sharing applications. Features of the spooling system are:

- *Handles up to seven independent spooling devices and queues with "print on first available device" feature.*
- *Prints large block header pages between listings with date, time, user name and job name.*
- *Multiple listing copy option.*
- *Complete forms change capability for each job and device.*
- *Prints formatted or unformatted listings.*
- *Status command displays print queues and status.*
- *User can kill or change priority of queued jobs.*

*Available **only** for OS-9 Level Two Systems.*

Suggested List Price: $150.00 Manual Only: $15.00

2 RMA RELOCATABLE MACRO ASSEMBLER

At last — a full feature relocatable macro assembler and linkage editor for OS-9. RMA permits sections of assembly language programs to be independently assembled to "relocatable object files". The linkage editor takes any number of program sections and/or library sections and combines them into a single executable OS-9 memory module. Global data (including indexed and direct addressing modes) and program references are automatically resolved in the process. The macro facility permits commonly used statement sequences to be defined, then used within the program with appropriate parameter substitution. RMA also supports conditional assembly and library source files.

Suggested List Price: $200.00 Manual Only: $20.00

3 OS-9 FILE HANDLER TOOLBOX

Introducing a special toolbox for OS-9 users who do a lot of file manipulation! A collection of 12 useful OS-9 command programs. Most can be used as "filters" using OS-9 pipeline facilities. Included are:

***D** — unformatted directory listing with "wild card" matching*
***Compress** — does character compression on text files.*
***Expand** — restores a "compressed" file to the original state.*
***Split** — breaks a file into smaller files.*
***Space** — indents lines with optional spacing between lines.*
***Code** — decodes any key on a keyboard to hex.*
***Qsort** — quick sort for small files, directories, etc.*
***Pr** — versatile formatted file printing utility.*
***Tr** — transliterates text pattern to substitution pattern.*
***Grep** — searches file for a pattern and prints matching lines.*
***Xmode** — same "tmode" except changes are made to the device descriptor.*
***Count** — counts words, lines, or characters within a text file.*

Suggested List Price $85.00

4 ENTERTAINMENT PACK I

A collection of games and other interesting programs that are not only entertaining but serve as good instructional examples of Basic09 programming techniques. All programs include complete Basic09 source files and can be easily edited to run on standard alphanumeric or graphics terminals.

***Blkjak** — A Vegas-rules blackjack game.*
***Clk** — graphical display of a wall clock on your terminal.*
***Dogs** — Greyhound racing with simulated graphics.*
***Eliza** — Basic09 version of the famous artificial intelligence simulation of natural language dialogue with a psychiatrist.*
***Haiku** — Program that creates original "haiku" prose.*
***Quest** — a mini-"Adventure" game.*
***Rats** — find your way out of a computer-generated maze — from a rat's point of view.*
***Towers** — a graphical display of the solution to the "Tower of Hanoi" puzzle.*

Suggested List Price: $85.00

MICROWARE®

Microware Systems Corporation
P.O. Box 4865 • Des Moines, IA 50304
515-279-8844 • Telex 910-520-2535

OS-9 and Basic09 are trademarks of Microware® and Motorola.
Unix is a trademark of Bell Labs.

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. JBO

FLEX™ USER NOTES
THE 6800-6809 BOOK

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

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

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 idea. Thanks and good luck, you guys are the best! TOA

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

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

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

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

™FLEX is a trademark of Technical Systems Consultants

$4,325
FOR A WORLD-CLASS SS-50 COMPUTER

Smoke Signal's VAR/68™ gives you:

- Fabled Chieftain performance that led the pack in tough Benchmark surveys
- Integrated, easy-to-use software that covers your *complete* business needs
- Proven reliability backed by our exclusive Endurance-Certification Program
- Extremely good looks and unsurpassed operator comfort

$4,325: A PRICE CALCULATED TO GET YOU HOOKED ON THIS BLOCKBUSTER SS-50

That price buys you a VAR/68 computer with multi-user, multi-tasking capabilities, and an ergonomically designed terminal. You get 128K RAM—expandable to 1mb. Eight serial ports, up to 16 if desired. Two parallel ports—and more are available. Plus a long list of other impressive capabilities.

Smoke Signal's experience allows us to offer OS-9 and other UNIX-like, and multi-user operating systems.

The styling is completely new—fashioned for the utmost in operator comfort. And it's remarkably compact. VAR/68 is a combination of great performance and good looks demanded by the office of today.

VAR/68 IS TOUGH, BUT SMOKE SIGNAL GIVES YOU *EXTRA* PROTECTION

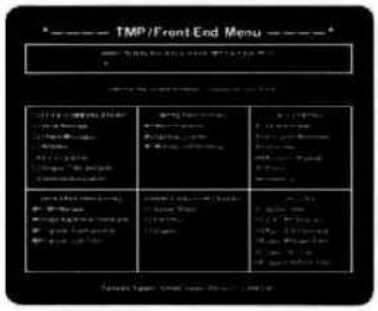
(1) Your new computer is Endurance-Certified before delivery. That's an exclusive quality-assurance process that guarantees perfect operations from day one.

(2) Our Advance-Replacement program is yours for a low fixed charge. (3) You get instant diagnostic service by telephone. It's free. (4) Normal repairs are handled with super speed. (5) Software *and* hardware support are part of doing business with Smoke Signal.

TOTAL INTEGRATED SOFTWARE GIVES YOUR BUSINESS *SOLUTIONS* INSTEAD OF PROBLEMS

Powerful business application programs are ingeniously interlinked to give even untrained operators a quick, smooth upper hand. The VAR/68 is a joy for first-time users, and an unprecedented productivity tool for *anyone* who wants new dimensions of control over critical business matters.

This screen tells part of the story:



GET A BIG DISCOUNT ON YOUR INITIAL ORDER

Most re-sellers can save up to 42 percent—even on small orders. Smoke Signal's price schedule is a powerful profit-maker for dealers of almost every description.

CALL SMOKE SIGNAL OR WRITE FOR MORE INFORMATION ON THE VAR/68 COMPUTER FAMILY

SMOKE SIGNAL
Products and Support for VARs

VAR/68™ BY SMOKE SIGNAL
The computer designed for your specialized application software

31336 Via Colinas • Westlake Village, CA 91362-3984 • (818) 889-9340

THE 68000 FROM SMOKE SIGNAL!

ADD 68000 AND UNIX™ TO YOUR EXISTING SS-50 COMPUTER AT PRICES 50% TO 75% OFF LIST

THANK YOU

Seven years ago, Smoke Signal was founded to sell state-of-the-art computer products, by mail, to individual professional programmers and hardware engineers. At that time, most big companies did not believe in the power or future of microcomputers for serious computing applications. Only after you, the individual computer user, proved the viability of the microcomputer was Smoke Signal able to sell systems for business uses. However, as we progressed to become the leader in SS-50 systems, we had to add the sales and technical support services demanded by these business customers — and our prices for complete systems reflected these added costs.

With the introduction of our 68000 products, we wanted to find a way to say thanks to you, our original customers, the individual computer users, and still offer complete sales and technical support to our business customers for complete systems. We think this offer accomplishes both of these goals. We are offering you a choice of upgrade kits that will bring any SS-50 computer up to the electrical equivalent of our complete 68000 computer systems at prices far below complete system prices. In fact, the prices offered are 50% or more off our normally low prices for the components contained in the upgrade kits.

This special offer is limited to one upgrade kit per customer and is our way of saying thanks to those of you who had confidence in us from the beginning.

THE UPGRADES

The following upgrade kits were designed so that any SS-50 system can be upgraded to 68000/UNIX.

SWTP UPGRADE..... $2,800.00
Contains: LMB-1A SS-50C Motherboard, DCB-4A floppy controller, PSA-1 Winchester/Tape DMA interface, SCB-68K 68000 CPU, SER-2 dual serial board, 5Mb Winchester and controller, power supply, all cables, and REGULUS.

GIMIX UPGRADE..... $2,500.00
Contains: Same as SWTP Upgrade except allows you to use your GIMIX motherboard, serial board and Winchester power supply.

Users of standard SMOKE SIGNAL systems may choose one of the following upgrade kits:

For SSB floppy based systems:

SS-FD UPGRADE $2,100.00
Contains: SCB-68K 68008 CPU, PSA-1 Winchester/Tape DMA interface, 5Mb Winchester and controller, power supply, all cables, and REGULUS.

For SSB Winchester based systems:

SS-HD UPGRADE..... $500.00
Contains: SCB-68K 68008 CPU and REGULUS.

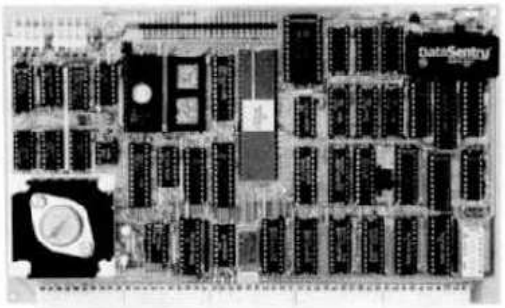
COMPLETE SYSTEMS

SMOKE SIGNAL is also making available complete VAR/68K™ systems at dramatic discounts. This offer is only available through SMOKE SIGNAL dealers. Contact SMOKE SIGNAL directly for information about how to order a complete VAR/68K system.

RULES OF THE OFFER

1) Limit, one upgrade system per customer.
2) Prices valid through December 31, 1984.
3) Orders must be accompanied by full payment in the form of individual check or credit card authorization.
4) Support will only be provided for systems containing the following SMOKE SIGNAL boards: SCB-68K, DCB-4A, PSA-1, and a motherboard such as the LMB-1A with extended addressing and main terminal I/O at FF7E8.
5) While we feel that most static RAM boards will work with these upgrades, we only guaranty compatibility with systems containing SMOKE SIGNAL static or dynamic RAM.

VAR/68K is a trademark of Smoke Signal.
REGULUS is a registered trademark of Alcyon Corp. UNIX is a registered trademark of Bell Laboratories. OS9 and OS9/68K are trademarks of Microware. MACSBUG is a trademark of Motorola Inc.



PRODUCTS

The heart of all these upgrade kits is SMOKE SIGNAL'S new SCB-68K 8 MHz 68008 CPU Board. This standard (5½" x 9") board will replace a SCB-69 CPU Board in any SMOKE SIGNAL computer with current revision boards. This board contains a real-time clock with battery back-up, 2 EPROM slots for up to 64K bytes of storage, a MACSBUG™ type monitor along with an auto boot loader and a mnemonic disassembler, plus many more features.

All upgrades also come standard with REGULUS™, a UNIX like operating system which is totally compatible with UNIX. REGULUS supports real-time tasks, shared memory, record locking and contains a shell similar to the Berkeley C shell. Along with the operating system, you get C, an editor, assembler, linking loader, interactive debugger and a word processor.

SMOKE SIGNAL is also including in many of the kits the DCB-4A double density floppy controller which can handle up to four 5" and four 8" floppies and contains 1K of buffer RAM for fast disk transfers; the PSA-1 Winchester/Tape DMA interface board which has taps for SASI and Priam disk interfaces as well as a tap for 90 ips tape streamers which are supported under both REGULUS and OS9™; either a M-256-X or M-512-X dynamic RAM board with over two years of field proven reliability; and the LMB-1A heavy duty motherboard with gold plated connectors, extended addressing and on-board baud rate generator with ten selectable baud rates.

SOFTWARE

Software and Software Support is available only from Smoke Signal dealers. Spread Sheet, Word-Processing, Relational Database, C, Basic and Cobol are all available now. Additional system's software is becoming available every day because of the UNIX compatibility.

SMOKE SIGNAL dealers are also offering Microware's OS9/68K™ to purchasers of these upgrade kits. SMOKE SIGNAL will offer other Microware 68000 products as they become available.

SUPPORT

Even at these "lower than PC" prices, we're not going to leave you with "PC" type support. We've arranged with one of our very technically qualified dealers to provide you with add-on software and technical support. In addition to answering your questions on how to convert your system to the 68000, he has a group of his customers who are themselves computer experts who are joining in a network that will help with even the most technical questions. We hope you will contribute your ideas to the network so that we all can benefit from new and fresh thinking. Complete details of the support available are included with the upgrade systems.

ORDER FORM

Fill in your name, address and phone number below. Your order will be shipped UPS so please do not use P.O. Box. Check items being ordered on form. Add prices for all items selected. CA residents must add 6% for sales tax. Total the amount for your order and check payment method below.

Name _____
Address _____
City, State, Zip _____
_____
Phone _____

Payment: ☐ Enclosed Check
☐ VISA
Mastercard

Card # _____
Exp. Date _____
Signature _____

☐ SS-FD UPGRADE	$2100	_____
☐ SS HD UPGRADE	500	_____
☐ SWTP UPGRADE	2600	_____
☐ GIMIX UPGRADE	2500	_____
☐ M-256X RAM	648	_____
☐ M-512-X RAM	948	_____
☐ SER-2 I/O	65	_____
☐ 20Mb HARD DISK (Instead of 5Mb)	800	_____
Sub Total		_____
CA residents add 6%		_____
Total		_____

SEND COMPLETED ORDER FORM TO:
SMOKE SIGNAL
31336 Via Colinas, Westlake Village, CA
91362-3884
(818) 889-9340

RUMORS & SUCH

This month not much to rumor about. However, some important items to announce and some to ponder.

First, in the next month or so we will have a regular 68000 column. As the quality and actual availability of 68000 applications software becomes more available to the average end-user, we will expand our 68000 coverage.

Our 68000 column author is well known in the 68XX community, not by name but by the software products he has brought us in the past. I know that you will want to follow this column. It seems that many of you will be converting over to the 68000 in the foreseeable future.

It will be as our usual practice: more gut level information and less flowerly garbage. Of **ALL** the computer magazines, now or in the past, we - 68 MICRO JOURNAL - have published **MORE** useful code and **how-to** information than any, I mean **ANY!!** We have had to publish less corrections than **ANY** other!! We have had a larger renewal rate than **ANY** other!! And we will continue to do the same with the 68000. We have survived in the smallest computer community market for over 6 years. I have seen 7 magazines or Journals(?) fail in our marketplace. Yet we remain, stronger than ever - we delivered!! And that my dear readers is the answer - deliver. I have watched hardware and software companies grow or go, over the past 8 or 9 years. For the most part it all boiled down to deliver and support.

Now, understand this we **WILL NOT - WILL NOT** abandon the Standard S50 Bus and all of you who are not inclined to go 68000. We are expanding, not replacing. Fact is I can cite you instances of those who were with us once but followed the lure of greener grass, only to falter in the larger more competitive pasture. We know where we **came from, and WHO got us where we are - YOU users and readers!** I have not forgotten.

Have recently, last week, returned from the annual Southwest Technical Products Corp dealer meeting. Will have some things to report to you about that but will have to wait as time constraints precluded its inclusion in this months issue.

Also we have received our LSI 68000 CPU board, actually two boards, and hope to have some preliminary info as we bring this system up running CPM 68K. It seems to be well done and we have high-hopes for those 68000 boards that fit the Standard S50 Bus. This one looks good.

Also next month we hope to have some insight to what went on at the Spring/Comdex. We have two reporters from our local staff there now and they report some exciting 68000 products.

GIMIX AWARDED "E"

We received a notification from the U.S. Dept. of Commerce, as follows:

. . . .

GIMIX Inc., has been selected to receive the President's "E" award for exports. In recognition of outstanding contributions to the increase of U.S. trade abroad.

Signed:
s/Malcolm Baldrige, Secretary of Commerce

. . . .

Congratulatons, Bobby, Richard, Mike and all!

We understand that the actual award will either be made by the Chicago District Office or at NCC later this year. Either way it is about time some of **OUR** folks received the credit due the quality of materials and services afforded by Standard S50 Bus suppliers.

DMW

- - -

Ellen W. Commo Dies

Right after last months edition went down, I received word that Ellen Commo, wife of Norm Commo, had died, after a prolonged illness.

It was a shock to myself, our family and employees. You see, we here at CPI knew for quite some time of Ellen's illness. We marveled at her patience and understanding as Norm took time from the family to write the "C" User Notes column for us. As or Norm, well, I can never express my gratitude for his devotion to all of us. I have learned a lot by this experience, and not all confined to the C language!

Around the first of the year she seemed to improve and we all rejoiced. Her sudden passing came as a sad reminder that all of us are but a breath away from death. And our prayers have gone out for Norm and the children, in these heart-burdened times. I know that all of you join me in saying, **"Thanks Norm, may God give you and the children peace, understanding and strength in these times."**

DMW

- - -

Flex User Notes

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

Reader Response on Debate

I've gotten very busy and the last column didn't get transmitted to Dan Farnsworth in time to have his further response here (if he has any). I would like to quote another letter and comment on it, however. Just so you know that all the letters I've received are not in agreement with each other (or with me), I will present the following, received from Geoffrey A. Gass. Of course I have some comments to make in response. The letter follows:

"The assemble/compile controversy in FLEX User Notes ('68' MJ April 1984) is just my meat. Allow me to pull up a chair...

"As it happens, I started out on 68xx machines much the same as you did, poking hex code from a home-made terminal into my SWTPC 6800 for several years, even after Dan Meyer sent me a copy of Co-RES as an honorarium for a full-featured LIFE I sent him. It took many months of debugging (some of the bugs went all the way back to Motorola -- see DDJ, May 1979) and adaptation to make the assembler usable, and it wasn't until January 1980 that I had the beast running under FLEX and capable of directly generating executable FLEX object-code files (MSI did a version that would save source to their disk, but put out object code only to tape ... !). At that point I had already hand-coded programs as large as 12K (8K resident with a 4K overlay), but could now rely on an assembler to handle the chore with a minimum of aggravation.

(LIFE refers to "The Game of Life, an exercise in simulating bacteria culture colony growth according to a simple set of rules ...RWA)

"From the very start, however, I had been writing programs in assembler format with symbolic references and Motorola Mnemonics. All the assembler did was take over the mechanical coding chore, calculate branches and pretty print the documentation. There was essentially no change in what programs could do or how they did it.

"When it comes to high-level languages, though, it's an entirely different story. Although by the Turing criterion, ANSI BASIC is no doubt capable of calculating PI to 100 decimal places, it's hardly worthwhile even to think about approaching the problem this way. (I once tried to get DEC's FOCAL to compute the first 20 digits of PI on a PDP-8. After 20 minutes of grinding away, it had the first digit right!) No doubt your PL/9 with its 32 bit "REAL" numbers could be made to compute the compounded daily interest on a savings account of $37,215.65 -- but certainly not by any straightforward use of its built-in arithmetic functions.

"It is rare that a programmer is so intimately aware of the inner workings of a high-level language as to be certain of its exact limitations under marginal conditions -- such as knowing the variation in execution times for various operations as a function of data values, or even what the result would be of trying to exit a loop early by setting the loop variable to its limit value.

"High-level languages all have their little idiosyncrasies, awkwardnesses, limitations, and just plain bugs. To avoid these and optimize a program's "friendliness" with a user, smoothness and timeliness of execution, and accuracy and reliability to the degree demanded by the application, a programmer must often devise his own data types, establish his own primitive algorithms, and develop his own error-handling in a way that facilitates the proper continuation of the program, and does not bring it to a screeching halt with a silly "ERROR 207 in line 260" or a bomb-out.

"As Jerry Pournelle discovered (Byte, April 1984, p 70), a newly-enhanced version of an old familiar compiler may have some former safety-nets deleted -- and when you come to rely on them, wham! Broken back.

"Because HLL's are often not well equipped to manipulate a variety of interface devices, the programmer must add special segments and routines to handle them in machine code, to be called by the HLL program. The most outrageous example I know of is the Commodore 64 machine, which runs under a Microsoft Kernel and BASIC having no extensions to perform the graphics and sound functions of the machine. The display and sound generators must be manipulated by BASIC POKE and PEEK statements. A look through Compute's Gazette shows pages and pages of DATA statements which are decimal translations of machine code instructions to be POKED into memory and executed by SYS statements.

"Here then, is a self documenting high-level language executing totally un-annotated (not even mnemonics!) machine language instructions, to get done what the HLL cannot properly do by itself (in any timely way). How much superior in every way it would be to have a properly annotated assembly listing and no HLL at all!

"If a programmer does not get carried away by the attractions of every new chip that comes down the pike, he eventually develops a library of routines that perform commonly required functions -- like parsing command tables, initializing a PIA or performing a 64 bit multiply -- and can use these to put together a reliable skeleton for nearly any type of program. These become like a compiler's runtime library, but with this marvelous difference: the routines are not chiseled in granite, and are adaptable to whatever the needs of a new program may be. If that 64 bit multiply needs to handle only 24 bit data, it can be trimmed down to size in a wink. If the initialization routine needs to check the printer's handshake (to make sure it's plugged in and turned on), so be it: the necessary code is easily added.

"No function need be abandoned, compromised or made so slow as to be an annoyance or even useless just because the HLL was written in a particular way.

"Don't get me wrong: I do not say that HLL's are useless. I use them all the time, for the three T's:

1. For trivial programs: After doing my income tax this year, I became curious as to what happened to all that income I was paying tax on, so I put together an expense distribution program called "Budgeteer", which maintains a random access disk data array distributing each entered transaction among nine "accounts" for expenses and three for income. The program turned out to run 6 pages of BASIC code, but that total included 60 odd lines to allow the operator to enter the filename for the data array (Uiterwyk's BASICs do not allow string variables to be used in an OPEN statement).

"The program answered my questions and provided a means of keeping running accounts for the current year as well. It would not have been worth it to put together such a program in assembly language. BASIC was quicker and could handle the trivial chore quite adequately -- except for the filename awkwardness. (There's a similar program in the April '68' MH, I see.)

2. For temporary programs: This past year, I've written several programs for student drill: one for typing, one for Roman numerals (fun!), and one for arithmetic (you don't know what fun is until you do reduction of improper fractions in BASIC!). Again, it was hardly worthwhile to go to a full excavation and poured foundation for a palatial assembly language program here: a little tarpaper shack in BASIC was all that was required for a limited time usage, and that's all that got built.

3. For trial programs: Sometimes it's worthwhile to try out a program idea in a high level language. If it looks successful, but just needs some cleanup and speedup that the HLL can't handle, the HLL version can be used as a flowchart for writing the good version in assembler for actual use.

"Last year I wanted to develop an INDEX program for maintaining data files of document abstracts, with the capability of searching these abstracts quickly for keywords or combinations of words, to provide the names and locations of the original documents.

"Fast search dictated that as large a file as possible be resident in memory. It also dictated some form of text compression -- something HLL's don't handle well -- and dedicated routines (i.e., assembly language) for the searching.

"What I wound up with is a program which substitutes tokens for the 250 or so commonest words and phrases used in the field covered by the file. The program used 6K, the keyword file 4K, and the memory resident data file up to 18K per block. A complete (i.e., failed) search of 700 odd entries representing some 150 documents (just under 18K) takes less than a second -- a tenth of the time sleepy old FLEX takes to download a block from the DMAFI disk (FLEX is NOT well adapted to handling large blocks of data).

"Execution of this program is adequate, but it would certainly not be so if it had to battle with the data types, array restrictions and slow data handling of a HLL. Having to perform multiple disk accesses for every few hundred entries would make the program an exercise in frustration.

"(After I rewrite the I/O and overhaul the 20 page manual, I'll pass it on to South East Media: it could be saleable).

"In the end, I guess we'd eventually agree on this: Each high level language has a certain range of applications within which it does entirely adequately. And, generally speaking, programs within that range will be more quickly written, sooner debugged, more reliable, and more easily

maintained than assembly language versions, given the same level of programmer skill and experience. Where we might tend to disagree is the placement of that range.

"For each serious application, I find that I have to write a new HLL optimized for that application. That HLL is called the "program"!"

Response

First of all, read the second paragraph up, once more. I couldn't (and wouldn't) have put it any differently myself. I suspect that one of the reasons for our disagreement at where to draw the line, is the economic facts of life concerning writing software for a product in my case. The software has to work and work reliably. The quantity of units shipped with any one program is very small. We frequently do one of a kind programs for special applications that are similar to our standard instrument, but have some additional capability. One of a kind applications, if the resulting code is sufficiently compact and fast, certainly merrit the use of a compiler rather than assembler.

Geoffrey, thanks for this well thought out and well written discussion. To start at the top of your letter, my early machine language programming was on a 6502, and I also used mnemonics to write the programs and did the coding by hand and liberal use of the instruction set summary card.

In the next paragraph, you dive in and give several fine examples of misapplications of high level languages as reasons why they shouldn't be used at all. Of course you shouldn't try to do applications software to calculate anything to 100 places in BASIC. Though it is possible, such applications require a great deal of calculation (number crunching) and most BASIC interpreters are many times slower than some of the true complied languages. Certainly trying to use a compiler with a 7 digit math package to calculate the compounded daily interest on a savings account with 7 digits value, (daily interest requiring another 6 places to provide sufficient accuracy) would be a total misapplication.

Now let me give you a couple examples of good applications for these languages. TSC Extended BASIC has 64 bit arithmetic (56 bit mantissa and 8 bit exponent). It has scientific functions good to 15 or 16 digits. I'm writing a runtime package for a compiler that has a scientific function package. What better way could I devise to test my algorithms than with that 16 digit accuracy BASIC, if my package is to be a 9 digit one? This is a real life example. I've coded these algorithms in BASIC and tested them, then coded them in assembler, PL/9, Pascal, and "C" for fun and entertainment.

I have an instrumentation application in which the only input is from function switches on the front panel, and a 12 bit Analog to Digital converter. Is 24 bit arithmetic good enough for handling and processing data from a 12 bit A/D converter? Of course it is!

The point is that all HLLs are NOT created equal. I'll go a step farther. Not all implementations of the same HLL are created equal. You can't just grab one and do whatever you wish with it. Take Pascal as an example. One of the implementations available for FLEX9 is a "P-code" implementation. That is, the compiler translates the program into a set of machine instructions for a "hypothetical" machine. The runtime interpreter makes the particular processor act like the hypothetical machine. In a sense, a P-code compiler is somewhere between an interpreter and a native code compiler in the way it operates. Generally P-code interpreters generate very little code (they are code efficient in terms of object code generated) but the penalty is that they don't run as fast as a native code compiler's output. The implementation I have in mind (Lucidata) has 9 digit arithmetic, and its output runs about twice as fast as TSC Extended BASIC. I must mention that the compile step is VERY simple and VERY fast, an important consideration in the selection of a compiler.

OmegaSoft has an implementation of Pascal that is a native code compiler. It has 6 digit floating point capability. Its output code runs several times faster than Lucidata's, but the compile process takes much longer. Output code generation is quite efficient also. Both of these compilers have provision for declaring a variable "AT" a particular address, so that handling I/O via a PIA or ACIA is quite simple. Both have very good file handling capabilities, allowing the user to specify the filename.

Which one is best? That depends ENTIRELY on what you want to do. If you want to learn Pascal, you will opt for the easy and quick compile capabilities of Lucidata and live with the slower execution times. If you want to write programs for applications that are speed critical, you will want OmegaSoft Pascal and you will live with the slower compile time to get the faster execution time.

I must agree with your (and Jerry Pournelle's) observations about new versions of compilers. I am ALWAYS very wary of a new version. We once were "bit" by a new version that wiped out what we thought was a clever way of identifying the level and scope of variables. The Pascal compiler not only allowed upper and lower case, but distinguished between cases. We had the bright idea of using upper case names for GLOBAL variables, first letter capitalized for next level Local variables, and all lower case for local variables beyond the first nest level of procedures. The idea worked out fine, but the next version of the compiler was changed to allow both upper and lower case, but not to distinguish between cases. We therefore had dozens of "Multiply Declared" errors when we tried compiling our program. We had to go back and rename those variables to make them different in another way than case!

I've recently received new versions of a compiler that I use at work. It seems that very frequently, a bug repair closes a loophole and makes a statement that has been incorrect all along (but ran with the old version with the bug) suddenly produce the wrong results. This has happened to me several times with various compilers, and it is most assuredly a problem. Once again, if your range of applications is limited, you will settle on a couple of compilers, and find out what their idiosyncrasies are, just as you learn what the idiosyncrasies of the instruction set of the 6800 and 6809 are. Are you all aware that the 6800 and 6809 handle stack pointers differently? The 6800's stack pointer always points at the next available location on the stack, while the 6809's points at the last byte that was pushed onto the stack. In the 6800, the TSX and TXS instructions compensate for this offset, so that if you TSX, the value in X is one larger than the value in S and vice versa. That is fine, but if you STS SPVAL, and then LDX SPVAL, you won't get the compensation, and you find that you have to INX. The reason for the difference in the 6809 was to overcome that problem, and to make it possible to reference an item on the stack "indexed off the stack pointer". In other words, LDA 0,S will get you the same value as PULS A.

That is all fine, but a program written for the 6800 that uses STS and LDX with INX as compensation, will simply BOMB when reassembled with the 6809 assembler! Things aren't all that simple in assembler land either!

Your remarks about interface with hardware devices are pertinent. All the compilers that I consider useful, can handle hardware via the dodge of allowing a variable to be declared AT an address, as I mentioned above. "C" does not allow this, but any address in memory may be accessed very simply via a pointer. Pure Jensen and Wirth Pascal has no such provisions and is useful only for the purpose for which it was originally designed, that of teaching structured programming. Even Niklaus Wirth would not disagree with that statement. Anyone attempting to program a Commodore 64 using POKE and PEEK for graphics and sound, certainly has my sympathy.

Now let me paraphrase a paragraph of Geoffrey's. If a programmer does not get carried away by the attractions of EVERY NEW LANGUAGE or COMPILER that comes down the pike, he eventually develops a library of routines that perform commonly required functions.... Certainly the same thing can be said about compilers as can be said about processors.

I must agree with the remainder of Goeffrey's letter with regard to the uses he mentions for HLL's, though I would substitute BASIC where he uses the term HLL in this regard. (All his examples are of the use of BASIC). I frequently use BASIC for his three T's, and some less trivial applications. For example, I maintain a mailing list for a friend who is on the staff of Campus Crusade for Christ. He mails out about 320 letters a month to the folks who are interested in what he is doing, and provide support for him and his family. The effort on my part entails making corrections and printing a couple sets of labels every two months or so. The whole project probably involves about 6 hours a year of my time.

I wrote the set of programs that I use to manipulate the data file in BASIC. I have programs for adding to the address file, deleting records from it, changing records, sorting by name or zip code, and printing labels. All were written in a week of evenings or so. Sure I could code these in a compiled language or assembler, but the effort would exceed several years of time spent using the program. Since the limiting factor in the case of changing a record is my typing speed, and the limiting factor in the printing of labels is the printer, I would not gain any speed by recoding in a lower level language. Sorting might be improved, but most of the sorting time is spent reading and writing disk files, so the improvement would be slight. Generally, the sorting algorithm used has a much greater effect on sorting time than the language in which the sort is written.

Geoffrey mentions having had to program around Ulterwyk's BASIC limitation of not allowing use of string variables in an OPEN statement. My answer to that, of course, is that TSC Extended BASIC has been available in both 6800 and 6809 versions for several years now. It runs 10 to 50 times faster than Ulterwyk's original BASIC for the 6800, and it DOES allow use of string variables for filenames!

I certainly agree that BASIC is excellent for trying out an idea for an algorithm, and I have frequently used a debugged BASIC program as a flowchart or plan for writing code in Pascal, PL/9, "C", or assembler.

With regard to the INDEX program, that sounds like a case where assembler is obviously the only way to go. I'll go further, and agree that MOST utilities (for FLEX at least) should be written in Assembler for speed and compactness.

Final (probably) Remarks

Since I've been on the defensive here for several columns, I'd like to make a few positive assertions before I let this go once and for all. My claim is (and it has been clarified and reaffirmed by all the discussion that has come between the column that started all this and the present one) not only that I can write a program faster using a higher level language, but that it will be a better program, containing fewer hidden bugs. It will have cost less to write, will cost less to maintain, and it will be maintainable by someone else without my help. All the arguments about subroutine library files for assembler programs apply to programs written in HLLs as well, and the use of standard procedures and functions in HLLs further reduce the time necessary to write a program, and further increase the probability of minimal errors since library routines tend to have been debugged thoroughly previously.

All the above presupposes a GOOD MATCH of the HLL to the application. Given such a match, I will say that the speed penalty will in most cases be minimal, and that in the few cases where speed is the limiting factor, assembler subroutines can be used to optimize the program. The only possible remaining objection is that of memory usage. In my applications I consider a few EPROMs a small price to pay for the advantages.

Now, I suspect that any further discussion would be just so much redundancy, so let's discontinue the "Great Debate". I think after all is said and done, the difference of opinion is really just the one mentioned above, of where we draw the line between Assembler applications and HLL applications.

Since I've now revealed that I do program in Assembler, I'll leave this forum open to Dan Farnsworth for publication of more of his math routines if he would like to publish them.

OS9 USER NOTES

by Peter Dibble
("OS-9 Users Notes" Columnist;
'68' Micro Journal)

Standards

Several months ago I mentioned Smoke's special version of OS-9 Level Two in this column. The questions I posed about its compatibility with Microware OS-9 stirred up a lot of commotion, but thanks to Don Williams' intervention no blood was shed. Smoke Signal has agreed to give customers a choice of the accelerated Smoke version of OS-9 or the Microware version. I think Smoke Signal deserves much credit for offering their customers this alternative. Some, perhaps most, people who use OS-9 need extra speed enough to take the risk associated with a version of OS-9 not just like everyone else's. Cautious people (like me) can ask Smoke to send them the Microware version of OS-9.

It probably seems strange that I, a person who likes to fuss with operating systems, should get so worked up about changes to OS-9. After all, I enjoy adding non-standard features to OS-9; I even publish some of them in this column.

Plan to Attend the 3rd ANNUAL OS-9 USERS SEMINAR

August 17, 18, 19, 20
Pre-Registration Only!

- MORE INFORMATION
- MORE EXHIBITS
- MORE SPEAKERS
- HARDWARE
- SOFTWARE

- TECHNICAL SESSIONS FOR 6809 & 68K

Plan now to attend the 3rd Annual OS-9 User Seminar. This is an event you won't want to miss if you use, sell or are interested in systems that use Micr ware 6809/68000 software. Informative r und-table discussions on almost every aspect of the design and use of Microware software will be held. A bigger and better exhibit area will have display bo ths from many of the leading suppliers of OS-9 compatible hardware and s ftware. Don't miss this chance to increase y ur knowledge and skill in the latest micr computer software technology — Register today!!!

Fee: $125

Location: Marriott Hotel
Des Moines, Iowa

Don't Miss It: Pre-Register Now!
Call: 515/279-8844 or Write:

microware®

MICROWARE SYSTEMS CORPORATION
P.O. Box 4865 • Des Moines IA 50304

OS-9 and BASIC09 are trademarks of Microware and Motorola.

Let me examine the question of standards from a few points of view. There are things to be said for ignoring standards: mostly that ignoring existing standards is the way new, improved ones are born. However, consumers find standards convenient, and producers typically find standards crucial.

Good examples of standards that beg to be ignored can be found in the busses invented in the early days of microcomputers. Engineers I know agree that the S-100 bus is poorly designed. They would love to be able to make a few changes to its specifications. Our own SS-50 bus has gone through some evolution, but extending the address space beyond a megabyte will require further changes to the standard.

I don't know hardware very well, but I imagine electrical engineers learn to work around standards about the same way programmers do. Strict adherence to standards even when they have been outgrown often results in a "kludge." Either the old code is left there and a new structure built on top of it, or it is entirely replaced with code that does things "right," and adherence to the standard is added on as a special case, something ugly hanging off the side of the new idea. Both of these solutions look like poor design.

IBM is a good example of a company, in fact an industry, caught on a horns of a standard. Years ago they invented the 360 architecture, a computer architecture that they used for all their computers. The idea of having a line of compatible computers caught on nicely. Later, they extended the 360 architecture to include virtual memory and a few other goodies, giving the 370 architecture. It was also quite successful. Customers seemed to appreciate being able to move to more powerful computers without rewriting any software. Most recently, IBM produced XA, an extension of the 370 architecture which 370 customers can move to relatively painlessly.

While these hardware changes were going on, operating systems were being improved. Programs that ran under MFT (an old operating system for 360s) should run with no important changes under the latest version of MVS. This level of compatibility exists only because IBM has stuck grimly to its standards. This practice has brought them success, but not critical acclaim. I know operating system experts who pretend to feel sick when MVS is mentioned -- with some justification. That operating system contains layer after layer of history. In some places the complexity is so thick it is practically impossible to figure out what the programmer was trying to do. I imagine that, if the effort which goes into adapting MVS and 370 architecture to modern needs were directed toward designing new hardware and software, the result would be much faster and more useful than IBM's current 370-type products. I bet there are numerous engineers and computer scientists at IBM who yearn to junk the old standards in favor of something better.

Standards like S-100, SS-50, and 360/370 architecture have tied manufacturers to dinosaurs. They can't depart from their standards without hurting, and perhaps losing customers. The big computer and software manufacturers probably have mixed feeling about standards. The consumers of their products feel about the same way.

It is hard to resist a sexy new computer or piece of software. The non-standard offerings are frequently faster and in various ways better than the more conservative ones. The problem is that non-standard computers or operating systems are

risky. The excitement of being the only person in the state with some fast, elegant operating system fades fast when you have troubles with software availability.

We are lucky to be using hardware and software that have good standards. CoCo users are dealing with only one vendor and one machine. It is a shame Tandy didn't decide to use the same disk format all the other OS-9 systems do, but at least that problem is well known. It should be easy to exchange software and hardware between CoCos.

The SS-50 bus is also a good standard which has been carefully respected by the vendors that support it. I ran my Gimix disk controller board with a SWTPc CPU board and memory boards from three different sources for about a year with no trouble. If all those manufacturers hadn't respected the SS-50 standard, I couldn't have done that.

Microware OS-9 is solid across all the machines I know of. It is even possible to move from Level One to Level Two without changing software (provided the programs were written to appropriate standards). An OS-9 user can trade from a CoCo to a Hellx to a Gimix III system without rewriting any programs except where they use special I/O features of each computer (like graphics on the CoCo). A software house can use their Gimix III system with its high speed and debugging facilities to develop software which will run on a CoCo. Usually we can order software without paying attention to the manufacturer of our machine.

The standards within OS-9 are as important as the interface to user programs. The device drivers and other system modules I include with the column occasionally should run on any OS-9 system with suitable hardware. I rely on Microware to stick with the interfaces between system modules that they have specified. If I ever find the money for it, I will be able to buy a graphics board for my system. If the vendor is selling it for the OS-9 market, it will come with software to hook it into my system. That software will almost certainly work because its author wrote it and tested it on a system with the same interfaces between system modules as mine.

Programmers have the most to gain from carefully followed standards. If someone buys a program that doesn't run on his computer, he will complain -- maybe return the program. This is a problem for the consumer, but for the author of that program it is a disaster. Imagine what it would feel like to spend thousands of hours creating a masterpiece of a program, then discover that it would only run on a few of the computers you had counted on for your market. With Microware OS-9 on any supported computer a programmer can be confident that that won't happen.

Programmers would like to see more standards in the OS-9 world. I have wished and worked for a standard terminal interface for a year now. It is a shame that each programmer who wants to sell his programs has to invent a way to adapt his program to whatever kind of terminal it might encounter. A standard here would save days in program development time for each program that used it, encourage more programmers to use terminal features supported by the standard, and give purchasers confidence that a program would work with their terminals.

Standards that are the User's Responsibility

If your system comes to you non-standard in some way, you should complain to the person responsible. Once you have it, it's your baby. You can generate additional standards to simplify your system, or let chaos grow in your system.

Several areas come to mind as good places to institute standards. Directory structure is an especially good place to devise a standard. If you write a lot of programs, you may need a naming convention. A set of standards for documentation might help keep it up-to-date.

There are two policies that can be used to guide the construction of directory structures. The directories can be arranged by what the contents are (programs, text, spread sheet info.), or by what they are for (sort programs, household, User Group files). Each method has its charm. I use both, each where it seems appropriate, but I wish I had decided early which way I wanted to go and stuck with it. Sometimes I have to search for minutes before I find a file I haven't used in a few months.

It is a good question whether documentation for a project should be in the same directory with the source of programs for that project, in a sibling of that directory dedicated to documentation for several projects (or just for a single project), or in a directory which is the child of the directory with the source in it.

Some people think that directories should contain either only other directories, or only data files. I don't think I like that idea, but I can see some value in it.

Program names deserve serious thought. The shorter they are the faster they can be typed. It is easier to type L than LIST, but the shorter names are the more cryptic they become. LOOK or LOGOFF could also be abbreviated L. It has to be clear what the abbreviation stands for. It makes sense to me to give short names to frequently used programs. The names of the commands will stay fresh in the mind if they are frequently used even if they aren't very mnemonic. Less frequently used programs should have longer names both to save short names for more frequently used commands, and to jog the memory about their function.

The Users Group

The OS-9 Users Group plans to submit a list of "requirements" to Microware at the OS-9 Seminar this summer. If you have spotted a flaw in Microware's software that you think is of general interest, or would like to suggest that a new feature should be added to one of their products, this would be a good way to bring it to Microware's attention. Submit your suggestion in writing to the Users Group early enough that it will reach us at least a few weeks before the Seminar. Please keep it to about a page or less. We will have copies of all the suggestions available at the Users Group booth at the seminar. The suggestions will be discussed at the Users Group meeting and those about which we can reach a consensus will be given to Microware. We will try to get an official response to each suggestion from Microware -- something like: Impossible, not interested, will do, wonderful suggestion, or already done.

SUPPORT YOUR ADVERTISERS

"C" User Notes

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

INTRODUCTION

The column this month provides the names of several C language books and a C group. It also discusses the code-generating aspects of several 6809 C compilers. The example program is from Introl and displays one or more files in hexadecimal. It is, of course, written for the Introl C compiler.

C BOOKS

Due to the growing importance of UNIX, there is an increasing number of books being printed and marketed on the subjects of UNIX and C. As with all books, most of them are too specific or too poorly done to be recommended. Five of the best are described below.

The C Programming Language
by Kernighan and Ritchie
c 1978 by Prentice-Hall
Englewood Cliffs, NJ 07632
$19.95 retail

This book is a "must-have" for anyone wanting to learn or to use the C language. However, it is a reference manual, not a primer, and only very experienced, multi-lingual programmers will be able to learn the C language from this book. Thus, most people will need primers to become proficient in the use of C.

The C Primer
by Hancock and Krieger
c 1983 by McGraw-Hill
N Y, N Y 10020
$16.95 retail

This book attempts to provide access to the C language for beginners. It starts with the basics and works up to more advanced concepts. It covers all of the components of the C language itself; however, it does not cover the standard C library functions, which provide input and output. The more difficult concepts, such as pointers and structures, are covered at some length and with many diagrams. The book's primary shortcoming is the lack of exercises, worked or otherwise.

Learning to Program in C
by Plum
c 1983 by Plum Hall
Cardiff, NJ 08232
$25.00 retail

This book was written and published by Thomas Plum, who teaches professional seminars on the C language and on the UNIX operating system. Like the primer just described, it starts with the basic concepts and builds to more advanced concepts. It describes each language feature, the cases in which it may be used appropriately, and provides a style of coding each feature. Since it is taken from the seminars, it provides a large number of worked and unworked examples for the student to study. Since it it privately distributed, the book may be somewhat hard to get; a bookstore should be willing to special-order it from the source.

C Programming Guide
by Purdom
c 1983 by QUE Corporation
Indianapolis, IN 46250
$17.95 retail

This book is similar to the previous one. It is a primer for beginners which starts with the simple and works up to the complex concepts of C. It is written by a consultant who is a retired computer science professor. Its really strong point is the inclusion of a large number of examples, explained line-by-line. Also like the previous book, this book is privately distributed, so a bookstore must special-order it from the source.

The C Puzzle Book
by Feuer
c 1982 by Prentice-Hall
Englewood Cliffs, NJ 07632
$13.95 retail

This book is all examples, as the name implies. It is basically a workbook for becoming really proficient in the C language. Some of the examples are simple, but many are real puzzles, as the name suggests. It starts with basic arithmetic examples, works thru data types and coersions, then thru control flow, style, and other concepts, to pointers, arrays, and structures, which provide the most complex puzzles. It also covers the preprocessor commands. The book provides solutions for all the puzzles presented.

C GROUP

Although the 6809 community has no C user group of its own, it does have a representative in the BDS C Users' Group. The BDS C Users' Group Newsletter is published about four times per year by the C Users' Group, Inc., Box 287, Yates Center, KS 66783. Circulation is by subscription only, at $10 per six issues.

Subscribers are entitled to purchase diskettes from its public-domain C library, usually at $8 each; however, they are in CP/M format, which presents a problem to most 6809 users. The 6909 representative is Sidney Thompson, who has managed to get some of the disks and I have managed to convert a small number of them to 6809 disk format. Unfortunately, most of the programs have BDOS calls, complicating the conversion problems. I will put note any progress in this area, and in the efforts of the Motorola group of the Atlanta Computer Society to establish its own public-domain C library.

CODE GENERATION IN C COMPILERS

In order to generate code in his 8080 Small C compiler, Ron Cain devised a simplified model of the 8080 itself. This model has two 16-bit registers, one of which is addressable as two 8-bit bytes. It has up to 65536 bytes of addressable 8-bit bytes. It also has one 16-bit stack register which is capable of directly addressing only the two bytes pointed to by the register. The instruction set is highly limited, and is (of course) a subset of the 8080 instruction set.

Since they are supported by the 8080, this model supports 16-bit pointers and signed or unsigned integers, and 8-bit signed characters. It does not support longs, floats, or double data types, nor does it support structure or union composite data types.

Since most of the 6809 Small C compilers are heavily based upon Ron Cain's Small C compiler, the object code they generate often causes a 6809 to emulate an 8080. The code generated by this process would be rated by knowledgeable 6809 assembler language

programmers as very poor, as it is very wasteful of space and time.

For example, given the following trivial C program:

```
main()
{
        int i,j;
        i=0;
}
```

several 6089 Small C compilers would produce code similar to the following (for the "i=0" statement only):

```
* int i,j;
* i=0;
    LEAY 2,S
    TFR Y,D
    PSHS D
    LDD #0
    STD ,S++
```

whereas Introl C and McCosh C would produce code similar to the following for the same C statement:

```
    CLRA
    CLRB
    STD 2,S
```

which is similar, but not necessarily identical, to what a 6809 assembly language programmer would have coded.

Although this example was intended to be simple, the point is that the production of good object code by a given 6809 C compiler may not be assumed. Although the Full C compilers produce better low-level object code, in general, than do the Small C compilers, they also have higher overhead and are usually more difficult to use.

In one test recently performed by a friend, the same program was processed thru two Full C compilers and one Small C compiler. After making minor syntax changes required to successfully compile the program, the resulting object program was 30% smaller and faster when compiled with a Small C than when compiled with one of the Full C compilers. The other Full C compiler's loader would not load the program without error. However, the discussion of the problems and incompatibilities among the C compilers for the 6809 will be saved for a future column.

CODE OPTIMIZATION IN C COMPILERS

In terms of the Full C compilers, both Introl and McCosh C compilers provide optimization. It is integral to the Introl compiler, whereas it is provided as an external program in the McCosh compiler, although it may be invoked from the command line. The McCosh optimizer is claimed to save about 11% of the code and time required to process the object code, when compared to the original code.

Rather than correct the compiler to emit better code, Wordsworth and Everhart provide optimization programs (written in C) to scan the assembler file output from the compiler and correct the most common and blatant sequences of inefficient code.

The Wordsworth optimizer is a peephole type of optimizer, utilizing a 4-line peephole. It processes the following sequences:

```
OPTIMIZED       ORIGINAL

TFR X,D         (Removed)
TFR D,X

TFR X,D         PSHS X
PSHS A,B

LEAX p,S        LDD p,S
TFR X,D
TFR D,X
LDD D,X

TFR X,D         PSHS X
PSHS A,B
TFR D,X

TFR S,D         TFR S,X
TFR D,X
```

Wordsworth's optimizer saves about 20% to 30% of the instructions and time when compared to the code produced by his C compiler.

The Everhart optimizer is also a peephole type of optimizer, although it has a variable size peephole. It processes the following sequences:

```
OPTIMIZED       ORIGINAL

LEAS -2,S       LEAS -n,S
:
LEAS -2,S

LEAY p,S        LDD p,S
TFR Y,D
PSHS D
LDD [,S++]

LEAY p,S        LDB p,S
TFR Y,D
PSHS D
LDB [,S++]
```

Everhart's optimizer saves about 20% of the instructions and time when compared to the code produced by his C compiler.

Wordsworth's Middle C and Everhart's Small C have corrected some of the worst code generated by earlier versions of C for the 6809. They still require optimizers to correct the remaining sequences of bad code, however. Wordsworth is apparently no longer marketing either Small C or Middle C. Everhart's Small C is in the public domain.

EXAMPLE C PROGRAM

Following is this month's example C program; it is from Introl, and outputs a hex listing of files named on its command line to STDOUT, which may be the user's terminal or may be redirected to another device.

```
/*                                                      */
/*      hex - display a file in hex                     */
/*                                                      */
#include "stdio.h"
main(argc,argv)
        int     argc;
        char    **argv;
        {
        FILE    *fp;    /* file to output               */
```

```
        int     i;      /* command line index variable          */
        int     _ofst[];
        i=_ofst;
        if(argc < 2)
                {
                fprintf(stderr,
                   "hex: command line is hex <file> [<file>]\n");
                exit(1);
                }
        for(i=1; i < argc; ++i)
                {
#ifdef  STDIO
                if((fp=fopen(argv[i],"r")) == NULL)
#endif

#ifdef  FLEX
                if((fp=fopen(argv[i],"br")) == ERROR)
#endif
                        {
                        fprintf(stderr,
                                "hex: can't open %s\n",argv[i]);
                        exit(1);
                        }
                hex(fp,0L);
                }
        }
/*                                                              */
/*      hex - output a file in hex                              */
/*                                                              */
hex(fp,g)
        FILE    *fp;    /* file pointer                         */
        long    g;      /* starting address                     */
        {
        char    row;    /* row being displayed (modulo 16)      */
        char    buffer[16]; /* data in column                   */
        int     inbuffer;   /* number of valid bytes in column  */
        int     i;
        char    ch;
        row=0;
        FOREVER
                {
                inbuffer=getbuf(buffer,fp);
                if(inbuffer == 0)
                        break;          /* end of file          */
                if((row++ & 0x0f) == 0)
                        {
                        printf("\n       0  1  2  3   4  5  6");
                        printf("  7   8  9  A  B   C  D  E  F\n");
                        printf("------ -----------");
                        printf("  ----------- -----------  ");
                        printf("-----------  ----------------\n");
                        }
                printf("%04x",g);
                for(i=0; i < inbuffer; ++i)
                        {
                        if(!(i & 0x03) == 0)
                                printf(" ");
                        printf("%02x ",buffer[i]);
                        }
                printf(" ");
                for(i=0; i < inbuffer; ++i)
                        {
                        ch=buffer[i] & 0x7f;
                        if(ch < ' ' || ch >= 127)
                                printf(".");
                        else
                                printf("%c",ch);
                        }
                printf("\n");
                g = g + inbuffer;
                }
        printf("\n");
        }
getbuf(buf,fp)
        char    *buf;
        FILE    *fp;
        {
        int     cb;
        int     i;
        for(i=0; i < 16; ++i)
                {
                if((ch=getc(fp)) == ERROR)
                        return i;
                *buf++=ch;
                }
return i;
}
```

68000 USER NOTES

Phillip Lucido
2320 Saratoga Drive
Sharpville, PA 16150

Welcome to the first installment in a new regular column concerning the 68000. For many months now, 68MJ has been asking "Where is the 68000?" While it has been around for quite a while on non-SS-50 machines, these have generally been large, expensive computers using the Unix operating system, or small evaluation boards with no software at all, other than a monitor ROM. Finally, though, things seem to be moving for the 68000 on the SS-50, and so this column was begun.

For the record, I work as a systems programmer for a small company. Most of the work here has been with 8080 and Z-80 computers, using CP/M, though the company is now looking at 68000 and 8086 systems. My involvement with the 6809 and 68000 has been mostly on my own, though. I started with a SWPTC 6809, in kit form, in 1980. The first major program I wrote on that computer was a 6809 disassembler, which is now being sold under the name 'Dynamite'.

That original machine is now gone from the scene. In its place is a Hazelwood Computer System Helix. The Helix has both a 2 MHz 6809 and a 10 MHz 68008, with the active CPU selected via a toggle switch. The 6809 runs OS-9 Level Two and Flex, while the 68008 runs OS-9/68K Level One. There is a 256K dynamic RAM board inside, and a disk controller board that handles a single 5" floppy disk drive and a 19MB Winchester hard disk drive, both boards from Hazelwood. The disk controller is actually a 6809 with enough intelligence to logically divide the hard disk into a number of segments, so that OS-9, Flex, and any other operating system I get can all run on the same drive.

So Now What?

This column is supposed to cover the 68000. That is a fairly large arena, and I doubt that I will be able to do justice to all of it. Right off the bat, then, I am asking for your help. What do you want to see? I could probably find enough material writing about OS-9/68K alone, though that would not be fair to the other operating systems which are around. I might stick with the SS-50 bus, but I probably shouldn't ignore the Macintosh, the Sage, Radio Shack's Model 16, the 68000 on the S-100 bus, etc. If I'm going to do this properly, I am going to need your comments and ideas.

Obviously, I will have some trouble covering what I don't have access to, so for now, this column will mostly be about OS-9/68K and the SS-50. I will be on the lookout for other information, though, and will probably pick up additional operating systems as they become available for my machine. If you have something that you think should be in this column, then by all means send the information to me. I can't write about what I haven't heard or read.

If too much information starts coming in, or if there are too many different topics to cover, then this column will likely be split up. After all, there are currently three other regular columns in 68MJ, and all of these deal mostly with various aspects of the 6809.

68000 Operating Systems

There seem to be three major operating systems available for the 68000 at the moment. These are Unix, the various Unix look-alikes, and CP/M-68K. It is unclear just which of these will win the battle for supremacy, but I have some ideas on that score.

First, CP/M-68K comes backed by Digital Research. As such, it is likely to be the first choice of those thousands whose experience has so far been with DR's CP/M on the 8080, Z-80, and 8086. This is unfortunate, because CP/M-68K does a poor job of using the power inherent in the 68000. It is a single user, non-multitasking operating system. While it is possible to work with such a DOS, a short encounter with a multitasking DOS such as OS-9 or Unix will quickly spoil a person. DR is supposedly working on Concurrent CP/M for the 68000, which would remedy some of these problems. For now, though, CP/M-68K just doesn't seem to be as good as it should be. It is fairly inexpensive, though. I believe that it costs around $350, and includes a C compiler, as well as the assembler and linker.

If CP/M-68K is not powerful enough for the 68000, Unix strikes me as being too powerful. Unix was originally developed for use on large minicomputers, and it may not be well-suited to a micro. For instance, Unix relies on swapping programs on and off disks when running multiple users, or so I have been led to believe. Also, Unix is a huge operating system. The company I work for has a Motorola VME-10, with 384K of RAM. We were considering moving to Unix, but found that we would need 256K more RAM, plus about $2500 for the Unix license. This is on a machine that would have only one or two users!

Now don't misunderstand me here. I fully believe that the 68000 is as powerful as many minicomputers, and can do an excellent job in a large machine with a lot of terminals hooked up at the same time. At that level, the distinctions between microcomputer and minicomputer get very blurry. Unix is probably a good idea if you need that much power, but what about a small business or high level home computer? If you only need a few terminals, then Unix is likely to prove an expensive proposition. Some people and businesses can get by on a machine that has less than a megabyte of RAM and a 30 MB hard disk drive.

This leaves the large group of Unix look-alikes. There are a number of these, such as Idris and Regulus. I don't know much about them, but from what I've seen, they might just fit the bill. They seem to be the Unix system, trimmed down to work better in a smaller microcomputer environment. It may be that they are also too large for use without a lot of RAM and a large hard disk, but hopefully that isn't so. Also, these operating systems are likely to be available for SS-50 computers. Smoke Signal's 68008 board is available with the Regulus operating system, and various other manufacturers are considering the Idris OS.

OS-9/68K

There is, of course, a fourth choice. Microware, of OS-9 fame, has introduced its 68000 operating system. I recieved my copy of OS-9/68K in late April (a month ago as I write this). According to Hazelwood Computers, my copy was only the second one sold. It is a preliminary version (Version 0.5) and there are still some bugs to be worked out in some of the utilities. For the most part, though, OS-9/68K is operational. It is certainly powerful enough to start development of application software, which was the reason (excuse) I used to get it.

OS-9/68K is largely compatible with OS-9/6809. The file structure used is identical, except for one small change. The maximum length of a file name is now 28 characters, instead of 29. Other than that, files written by one operating system can be read by the other. I have used this capability extensively, since my early copy of OS-9/68K does not yet have a parallel printer driver (this is expected in a new version coming out in early June). To get a listing, I have to redirect the output to a disk file, then switch over to OS-9/6809 to do the actual printing. Many of the utilities have been improved, so the 68000 versions are more convenient than the 6809 versions, even when used on 6809 files. For instance, the 'dump' utility now allows a directory file to be dumped, and the dump can be started at any byte in the file, instead of at the start of the file only.

There are some changes to get used. The size of the entire system, including all of the utilities, is much larger. While the core of the operating system was written in assembler, all of the utilities were written in C, and are generally much larger than their 6809 counterparts, on the order of 10 to 16 times longer. Under OS-9/6809, I load about 20 different utilities into memory as part of the startup procedure, so they can be found without going to the disk. This only takes about 8K of memory. The same utilities, in OS-9/68K, would probably c e to over 100K.

The operating system was shipped to me on four single sided single density 5" disks. When I first recieved it, my computer had two 5" disk drives, with a capacity of about 350K per disk. It only took me a day or two to decide to splurge and send off for the hard disk drive, since the entire operating system would not fit on a single floppy. This is not to say that OS-9/68K cannot be run without a hard disk. It could probably do quite well with 8" disks, or with 80 track 5" disks. But since the price of a hard disk has dropped so much, I decided to go that route, instead of sending away for 8" drives (about $1500 for a 19MB hard disk, as opposed to $1200 or so for dual 8" drives).

While the size of the utilities is a disadvantage of writing everything in C, there are advantages. First of all, since it is generally easier to add all sorts of bells and whistles to a program written in a high-level language (not withstanding that debate over in Ron Anderson's column), the utilities tend to be a little more powerful here. The 'dir' command, for instance, includes a wildcard option, so only certain programs in a directory are listed. The command

```
dir -w *.c
```

would list all files whose names end in '.c' Many of the utilities, which expect one or more filenames, include options to allow those names to be supplied by a file or the standard input. In conjunction with the wildcard option in 'dir' and a pipe, this can be extremely useful. To delete all files which start with the letters 'test', the command to use is:

```
dir -wu test* ! del -z
```

The 'dir' command finds all of the names of the files to be deleted, which are then piped to the 'del' command over the standard input channel. The -z option tells 'del' to take its list of files to delete from the standard input (note - due to a bug in del, the option actually has to be '-z=' - this should be fixed by the time you read this).

Most of the utilities now use a uniform method for setting option flags. Options can be anywhere in the command line, and are preceded by a minus sign. This is the same method used in Unix commands. Also, most commands accept an option of '-?', which causes an explanation of the command and its various options to be printed.

So where does OS-9/68K fit in the great 68000 operating system race? Well, it does seem better fitted to a microcomputer, when compared with Unix or the like. In talking with the folks at Microware, I learned that they are not out, yet, to compete with Unix on the big systems. They believe, as I have been saying, that there is a large middle ground, ill suited to Unix, but perfectly suited to OS-9/68K.

As far as the future goes, Microware is actively working toward improving OS-9/68K. The C compiler is available now (my copy came in a week ago). Basic09 is just about ready for beta testing, and should be out at the end of June. Pascal and Fortran are loosely targeted for the third quarter. The version out now is Level 1, in which all of the memory space is accessable to all processes. Level 2, requiring an MMU (memory management unit), should begin beta tests in August. This version will give the protection from other processes that is important in a multiuser computer. Microware is also working on a new Level 3 operating system, which will implement virtual memory. This is a minicomputer concept, which allows programs which are larger than available RAM to run. This should be around sometime late this year.

I must admit to some chauvinism here. Those of us using SS-50 machines know what we've got, and are frustrated by the acceptance of the various 8080 and 8086 systems. If only we had that much software, we keep saying. We've already got the hardware. Microware is a company which grew up around our machines, and I guess its normal for me to hope that it will do well in the wider world of the 68000. Hopefully, this is the opportunity for the software developers using the SS-50 to gain some access to a much larger market than the one we've had so far.

What About Hardware?

I have been calling around to the various SS-50 bus hardware manufacturers, trying to determine their plans for the 68000. I'll have more information later on, when written materials start to arrive here. For now, this is the information I have from those phone calls.

LSI Enterprises seems to have been the first to advertise a board in 68MJ. Their ad first appeared in the September 83 issue. The LSI S68K/08 board uses an 8 MHz 68008, and can be bought in kit form or assembled and tested. The computer uses the CP/M-68K operating system. LSI will possibly license OS-9/68K later on, and are also investigating Idris (a Unix look-alike). They also are advertising an LSI 680 users group, which entitles you to a monthly newsletter, public domain software, and updates, as well as a discount on your first LSI hardware purchase.

Hazelwood Computer Systems first advertised their CP-08 in the June issue of 68MJ. The board uses a 68008, which may be set for 4, 8, or 10 MHz. Hazelwood is also working on a full 16 bit 68 0, as well as a 16 bit-wide memory board. These will allow word size data fetches using Hazelwood's special SS-64 bus structure, which has extra address and data bus lines, but will also work in normal SS-50 bus computers. The memory board will have 512K of RAM, expandable to 2M when the 256K-bit dynamic RAMs become cheap enough. Hazelwood's 68008 board runs OS-9/68K Level 1, while the 68000 will be tailored for Level 2. Prototypes of the new boards should be at the August OS-9 Seminar. Hazelwood is also giving serious thought to Idris and Unix.

Smoke Signal also first advertised their board in the June issue. Their SCB-68K CPU board uses an 8 MHz 68008. The operating system is Regulus, a Unix look-alike from Alcyon. A new version of Regulus, compatible with Unix System-V, will be available soon. Smoke Signal is working on software for Regulus, too, and have various packages available. These include Unify, a data base manager which is popular on Unix systems. Microware is currently working on a version of OS-9/68K for Smoke Signal's board, which should be finished soon. There is also an I/O processor for 68000 systems in the works, to be ready sometime in late summer or early fall, and other hardware projects are in the works.

Gimix has a board which is almost completed. They would not say what chip would be used, other than to say that it would not be a 68 8 or a 68000. That probably means the 68010, though it might be a 68020. Gimix is waiting for OS-9/68K Level 2 to release the board, since the protection and memory segmentation important for multiple users is not available under Level 1. Since this board will be using a 16 bit chip, instead of the 68008, it will have to split up the memory accesses into 8 bit chunks to accommodate the data bus of the SS-50. According to Gimix, this will be done without any wait states in only one bus cycle, when used in their GMX III. Gimix hopes to have a board ready for the August OS-9 seminar.

That's a Wrap!

That seems to be plenty for the first column. I hope I've managed to keep your interest. By next month, I should have version 0.6 of OS-9/68K, and can talk a little about the various utilities, the two editors supplied with the package (a screen editor and a line editor) and whatever else might turn up in my wandering through the system. I have also run the sieve of Eratosthenes prime number benchmarks, in both C and assembler, and will report on the results next month.

- - -

WINDRUSH IEEE-488 REVIEW

THE WINDRUSH IEEE-488 INTERFACE BOARD,
A REVIEW

By: J Albert McDaniel
54 Perham Street
Farmington, Maine 04938

HARDWARE:

The IEEE-488 talker/listener/controller board for the S30 bus is manufactured by Windrush Micro Systems Limited under license from Fleet Electronics Consultancy. The board is based on a design by John Moore of Fleet Electronics, and the software was developed by him. The particular one reviewed is Revision E, 1983, and came equipped with a Motorola MC68488L General Purpose Interface Adapter. The board is made of high quality glass epoxy with all IC's socketed, is double sided with plated through holes and gold plated connectors. The IEEE connector is attached by a cable which mates with a double row header; thus the cable may be lengthened if necessary. There is also a terminal block with AUX PORT, TRIG and common. (The AUX PORT is used only if your computer has 4 addresses per I/O block. The TRIG is used to send a GET, Group Execute Trigger, command to those instruments which can use it.)

The hardware was designed to meet the following IEEE-488A-1980 standards. The * indicates that no instrument was available to perform tests of this function; the & indicates that no instrument was available, but procedures are so similar that the board must function: all others were tested.

	SH1	Source Handshake; complete capability.
	AH1	Acceptor Handshake; complete capability.
	T5	Talker; complete capability
	L3	Listener; complete capability
	SR1	Service Request/Serial Poll; complete capability.
	RL1	Remote-Local; complete capability, including Local Lockout.
	DC1	Clear/Selective Device Clear; complete capability.
	DT1	Device Trigger; complete capability.
&	TE5	Extended Talker; complete capability.
&	LE3	Extended Listener; complete capability.
*	PP1	Parallel Poll Response.

C1,2,3,4,7 Controller Functions:

	System Controller
	Send IFC & Take Charge
	Send REN
	Respond to SRQ
*	Conduct Parallel Poll
	Send I/F messages
*	Receive Control
*	Pass Control
	Take Control synchronously or asynchronously.

I do have one complaint with the construction of the PCB. In order to use the IRQ hardware interrupt, a link must be set. (The manual indicates at one point that a switch is available and at another suggests setting the link.) However, on the board reviewed a programming link was not installed so a wire or link had to be soldered to the board. An earlier version of the board had a switch to turn the IRQ on or off as needed.

The hardware, however, functions very well indeed. The MC68488L apparently has the capability of running at 2 MHz, for the board would USUALLY function at that speed even though the slow I/O stretch facility was not enabled. If you can stand an occasional error in transmission or reception, you might try running this board at 2 MHz with the slow I/O turned off. The performance of the board was determined with the computer running at 2 MHz and the slow I/O enabled.

HARDWARE NOTE:

One annoying, potentially ruinous effect the author repeatedly observed (some people never learn) was that if instruments connected to the interface were turned off and on while a disk access was in progress the system would usually experience an interrupt. This usually led to the destruction of the file but never the entire disk. I am not certain whether this effect is a function of the particular software (my editor), the cables, the proximity of the instruments to the computer or to the interface itself. Cycling the instruments while no disk access was taking place had no effect of any kind. At any

rate, I finally learned!

SOFTWARE:

The IEEE-488 board comes with a generally excellent instruction manual which will be of continuous use. Included in the manual are (1) a general description of the capabilities, circuit, and operation of the interface, (2) a rather complete set of driver routines (in 6800 assembler) which allow communications with devices on the bus, (3) a series of back-to-back test routines (in 6809 assembler) to test one board against a known good board placed in two I/O slots of the same computer, (4) a circuit diagram and component layout, (5) a copy of the Motorola Data Sheet for the MC68488, and (6) a three part article "Get Your Pet" on the IEEE 488 Bus reprinted with permission from Kilobaud Microcomputing.

The driver routines give sample programs in 6800 assembly language and are sufficiently well documented to be used as is or translated to a higher level language such as PL9. (The author is a relative novice at assembly language programming yet had no trouble in using them.) The back-to-back test routines are also of value in programming the board to talk to the bus, for several advanced routines not often used are not included in the 6800 routines above. The articles reprinted from Kilobaud will also give the novice some help in interfacing.

After studying it a great deal, the Motorola Data Sheet will also become indispensable. Have you read Ron Anderson's comments on the 6840? This chip comes with 13 active registers (one of which is not used in this implementation) at 7 different addresses. Each bit, of course, has its own meaning. The non-existent register, REGIW, has been used to send the control commands, REN, ATN, IFC and EOI. The unused register has been decoded to read SRQ. So you are back to 14 programmable registers. You may correctly gather that without John Moore's programs, I would have been lost! In fact without these programs the board would be effectively unuseable, or at least extremely difficult to begin to apply.

SOFTWARE PROBLEMS:

Now that I have praised John Moore's software, I must mention one problem that I had. I don't believe that it is a true bug, but it certainly bugged me for several weeks off and on as I found time to work on the review. (Being a chemistry professor is only a sidelight!)

Before I go into the problem with the software, some particulars about the IEEE-488 Standard should be given. The IEEE-488 Standard specifies a byte-serial bit-parallel intercommunication system between instruments designed to its specifications. This byte-serial bit-parallel arrangement necessitates a "handshake" between the listening device and the talking device. (The computer will either be listening or talking even when it is the controller.) Once a byte has been transmitted or received the other device must acknowledge that the byte has been accepted. In the MC68488, the data is transmitted or received through one register while the handshake is tested for completeness by accessing another. It seems reasonable to assume that the test for the handshake could be made either before or after the transmission. (If one tests before the next transmission then the 6809 during the intervening period could process information necessary to output the next bit while the MC68488 is testing the handshake of the previous transmission. This in theory could save time by having the 6809 not wait on the handshake.) Using the original software with the test for handshake completeness before the next transmission led to discovery of the problem. The problem is quite possibly hardware dependent so a description is necessary.

The University of Maine at Farmington has a Kelthley 192 Programmable DMM interfaced to a GIMIX 6809 Microcomputer through the Windrush board and a 1923 interface for the Kelthley. The voltmeter is a rather intelligent device (with a MC6808 at its heart and a MC68488 for an interface), and at first it looked like we might have a intermittent hardware problem with the 192 as the programs would crash in as little as a few seconds up to hours. It wasn't until Kelthley loaned me a second 192 that I knew the problem was with the software.

The Kelthley was programmed to take data at repeated intervals using the I/O test routines BYTOUT and BYTIN in the manual. The program would function for a time and then crash. The problem was usually found to be a forever loop involving the handshake of BYTOUT and would occur whether the computer was running at 2 MHz with the slow I/O stretch or at 1 MHz (although failures seemed to occur somewhat more often with the 6809 running at 2 MHz.) By outputting to the terminal the time from the internal clock and all bits sent to and received from the Kelthley, I was thus able to determine exactly how long the program ran before it crashed. Much to my amazement, both meters functioned essentially identically. I got quite used to having my computer run all night (or any other time I didn't need it) to test for another failure. It was even more amazing that the failure rate seemed to depend on what feature of the Windrush board was being tested or what kind of programming was being used on the Kelthley. (I should point out that I was simultaneously learning the capabilities of the Kelthley.) A given program would often run for hours with no failures and do so consistently, and another would crash in four minutes or even a few seconds. NOTE: Simply reversing the order of the test for the handshake cured the problem! And as written above, the problem may very well be hardware dependent! You may either be able to use John Moore's technique or may need to use mine. If your computer runs at 2 Mhz, you may also want to add NOPs (GEN $12 in PL9) after transmission of the universal command signals in order that a given instrument be able to execute the command before receiving any more information.

ORIGINAL OUTPUT ROUTINE: (by John Moore)

```
BYTOUT  PSHA
L7      LDAA    REG0
        ANDA    #$01000000  Previous Byte Gone?
        BEQ     L7
        PULA
        STAA    REG7
        RTS
```

MODIFIED OUTPUT: (In PL9)

```
PROCEDURE BYTOUT(BYTE OUTBYTE):BYTE INDEX;
     REG7 = OUTBYTE;
     GEN $12; GEN $12; /* USED ONLY AT
                          2 MHz for slight delay */
     REPEAT UNTIL REG0 AND $40 <> 0;
                  /* PREVIOUS BYTE GONE? */
     CHAR = OUTBYTE;
ENDPROC;
```

One other very minor point concerning the back-to-back test routines. The error checking routines will determine as many as 66 different errors. The only output is ERROR # . To determine where the error occurred you must read through the program listing-line by line. Then you must decipher what the program was doing to determine the cause of the

failure. This is a minor point for most of us will only have one board and thus couldn't use the routines.

CONCLUSION:

I would highly recommend this board to anyone who has an instrument with IEEE-488 capabilities and desires to have the instrument be interfaced to a computer with an S-30 or S-30C bus. I can also predict that you will have some learning to do, particularly if you are not already familiar with the IEEE-488 bus.

ACKNOWLEDGEMENT:

I would like to publicly thank Keithley Instruments for loan of a second 192 for comparison purposes. (Incidently for an American company which stands behind its products, you can do worse than Keithley. Their customer service personnel are fantastic and the hardware superb.) Without the 192 loaner, I might still be chasing non-existent hardware bugs.

VDISK REVIEW

Product Review of
VIRTUAL DISK FOR OS/9 LEVEL 1

by Bud Pass

INTRODUCTION

Many OS/9 users have hardware which will support extended addressing but would be quite expensive to upgrade to support OS/9 Level 2. This includes many SWTPC systems using the MP-09 CPU board and many GIMIX systems using several models of CPU boards; the primary requirement is that the CPU board have a OAT facility, and that the memory boards be capable of extended addressing.

This product allows such users to access the extended address space as if it were a single disk drive. In testing, virtual disk has been determined to be several times as fast as a hard disk, although the overall performance improvement is very dependent upon specific applications and usage.

HARDWARE CONFIGURATION

In order to use virtual disk, the system must have at least 8K bytes of extended memory beyond the 56K bytes of memory used by OS/9 Level 1. ALL memory boards in the computer must have extended addressing enabled. The processor board must also have a SWTPC or GIMIX OAT facility installed and enabled. The I/O board may have the extended addresses decoded if it is desired to use the top 8K on all except one extended page; otherwise, the I/O board decoding of extended addressing is not required.

If the memory boards do not support extended addressing, it may be possible to modify the board to use the VMA* line on the SS-50 bus, in addition to the four extended address lines on the SS-50C bus, as inputs to a 74LS138 decoder to cause the memory board to respond only to memory addresses within a specific 64K extended address range. This set of modifications was discussed in my article in the September 1982 issue of '68' Micro Journal and, in somewhat less detail, in an article in the September 1983 issue of '68' Micro Journal.

The addressing of the 56K bytes of memory used by OS/9 must be on extended page zero. The addressing of the memory for the virtual disk can be on any other page and in any order. This means that if the system does not decode the I/O board for extended addresses (as is the case in SWTPC mother boards), the maximum size of the virtual disk is 840K bytes. If the I/O board is capable of decoding extended addresses, it must be set to decode extended page $F. In this case, the maximum size of the virtual disk is 960K. For comparison, a standard 5.25" single-sided, single-density, 35-track, disk drive has an 87.5K byte capacity under OS/9.

SOFTWARE CONFIGURATION

Once the memory is properly configured, the memory map must be determined for the device descriptor. Memory is mapped in 4K segments which corresponds to the OAT mapping. All memory used must be in complete 4K byte segments.

There are two slightly-different OAT configurations that are supported by the program. The first is the SWTPC OAT, which has the high nybble equal to the extended page and the low nybble equal to the complement of the number of the 4K byte segment on the page. The other is the GIMIX OAT which has a high nybble the same as just described and the low nybble equal to the number of the 4K byte segment on the page. The driver determines the type of OAT from the device descriptor, which is assembled with an indicator of the type of OAT and a map of the extended address pages present on the system.

This extended address map is a list of the 4K byte segments which comprise the virtual disk space. Each entry in the list is one byte composed of the high-order eight bits of each 20-bit extended address representing the start of each 4K byte segment. The order of list entries is arbitrary, but, once established, should be consistent. Page zero of the system memory must not be included in the map, nor may the high 8K bytes of each 64K block if I/O extended addresses are not decoded.

Some examples of extended address map entries follow.

```
for a 32 K board addressed $10000 - $17FFF
 fcb $10,$11,$12,$13,$14,$15,$16,$17 pg 1 0-7 32K
for a 32 K board addressed $20000 - $27FFF
 fcb $20,$21,$22,$23,$24,$25,$26,$27 pg 2 0-7 32K
and 1 24K board addressed  $18000 - $1DFFF.
 fcb $18,$19,$1A,$1B,$1C,$1D          pg 1 8-E 24K
```

Once the device descriptor has been assembled and the device driver has been assembled (if necessary), they may be loaded. Their default names are "/v" and "Vdisk", respectively. Whenever the virtual disk is to be accessed, they must be present in memory. If desired, they may be loaded automatically with the OS/9 startup shell script or they may be included in the OS/9 boot file itself.

The standard OS/9 FORMAT program is used to initialize the virtual disk. If all of the system's extended memory is battery backed-up, the FORMAT program need only be used once unless the virtual disk contents are corrupted, the battery fails, or the user desires to clear it. However, if some or all of the system's extended memory is not battery backed-up, the FORMAT program must be used after each power-up. Since OS/9 Level 1 does not itself disturb the contents of the virtual disk, the FORMAT program is not required to be used after each re-boot. However, the virtual disk program has no facilities for determining if the contents of virtual disk has been corrupted by software,

hardware, or liveware error, so critical files should be reloaded to virtual disk after a re-boot, or the FORMAT program could be used on each boot to ensure the integrity of the virtual disk.

MODIFICATION FOR LARGE CONTIGUOUS MEMORY CARDS

There is a simple modification for 64K byte boards to "fold" the memory into two 32K byte pages. That is to switch address line A15 with A16. This will make the card's A16 extended address line determine whether the 32K byte pages will be decoded in the top or bottom half of the 64K byte physical addresses and the A15 extended address line will logically move the second 32K byte block to the next extended page. For 256K byte memory cards, line A15 may be switched with line A18. Whatever the memory board arrangement, the A15 line should be switched with the lowest address line that is switch or jumper selectable.

However, there are two limitations in the use of this technique. The first is that the first 64K of memory (extended page zero) must be logically and physically contiguous, since OS/9 does not intelligently manipulate the DAT to logically relocate blocks of memory, as FLEX and UNIFLEX do. The other is that this will limit the amount of memory which may be placed into the system to about 512K bytes.

EVALUATION OF THE VIRTUAL DISK FACILITY

Once the instructions for the establishment of the virtual disk have been followed, the virtual disk facility works as described earlier. If the user is accustomed to the speed of 5.25" disk operations, the speed of virtual disk will make the system seem vastly faster. It is also faster than 8" floppy or hard disk operations, although not by so large a margin.

The current cost per byte of one-megahertz virtual disk (about $200 per 64K bytes) is very comparable to the cost of 5.25" disk storage, and is dropping faster than the cost of disk drives. The virtual disk facility suggests a small system configuration of two 5.25" disk drives and 64K to 960K bytes of virtual disk used in place of a hard disk drive.

The current implementation has several shortcomings, which hopefully will be corrected in the future or are not important to the application.

The most obvious current problem concerns the necessity for manually constructing the extended memory map and reassembling the device descriptor whenever the extended address configuration is modified. Ideally, the device driver would automatically construct the extended memory map whenever the virtual disk is formatted. The only option selection required by the user then be the selection of the proper device descriptor for the type of DAT, and that should be required once per virtual disk installation. This is not a great problem for a system which does not change often.

Another current problem, already discussed earlier in this review, concerns the lack of virtual disk integrity assurance. There is no automatic warning that that the virtual disk has not been formatted or has become corrupted since formatting, due to hardware, software, or liveware error. A method such as a global checksum stored in virtual memory could be used to solve this problem. OS/9 provides a system call for generating a three-byte checksum which could be used to implement this integrity assurance. Without it, the contents of virtual memory should be considered suspect after a re-boot and critical files should be reloaded.

A related problem concerns the uncertainty surrounding the necessity of when to format the virtual disk. The definition of a conditional virtual disk formatter would solve this problem. It would format the virtual disk if the checksum were not correct.

CONCLUSION

The VDISK facility for OS/9 level I was reviewed here. I am using it and, other than the inconviences just discussed, it seems to function properly, at least under OS/9 version 1.2 on a SWTPC with 512K of memory. It is very fast and convenient, especially for DSAVE and other temporary files. If the limitations are not important in your application, it may be very useful to you also.

VDISK is available from South East Media. The object-only version is $79.95 and the version with source is $149.95. Their ad appears each month in '68' Micro Journal.

DISASSEMBLER

OVERVIEW

This disassembler is designed to work with TSC 6809 binary files. It reads the file and produces a source listing (a la ASMB) and a source file (a la .TXT).

The source file created is compatible with both the TSC editor and the TSC assembler. It contains only tags, opcodes, and operands.

The source listing is in the TSC 6809 FLEX assembler format. It contains the line number, address, object code, tags, opcode, and operand. In addition, it contains the NAM, ORG, FCB, FCC, SETDP, and END pseudo-ops. If direct paging is used, this fact is flagged, and a comment is printed on every line which uses it. In addition, the line(s) which loads the DP register is also commented.

After the source is printed, a sorted cross reference of all tags is printed, although this may be suppressed. It shows the line where the tag is defined, the tag, and all lines where it is referenced.

Since much of the information on the listing is NOT put to the source file, a hard copy is highly recommended.

DIRECT ADDRESSING

Because the 6809 allows the direct page to be anywhere in memory, it would be impossible for a disassembler to determine where the direct page is. Consequently, we simply flag the fact that it is used, and leave it up to an intelligent being to figure out what page it is on.

MULTIPLE FILES

Because of the APPEND utility, or a "backwards ORG" statement, the code on disc may not use

continually higher memory addresses. If this is the case, the disassembler will create multiple files & listings. These will be distinguished by the last letter of the extension, which is incremented for each section (.DSA, .DSB, etc.).

NOTE: the line numbers will start over at 1; the page numbers will continue to increment.

HIDDEN JUMPS

The entry point of any code which is not explicitly referenced will be flagged on the printed output by an asterisk in the tag column. This code may be unused. It may even be data. On the other hand, it may be accessed by another program or a

Jump table.

RUNNING THE THING

Copy the disassembler to your system disc. NEWDISK a disc and copy the binary file you want to disassemble to it. (Unfragmented discs run faster.)

The printed output is sensitive to the TTYSET parameters. Suggested parameters are:

TTYSET,EJ=0,WD=80

The depth should be whatever your CRT will handle. If the output is not going to the CRT, the programs assumes a 55 line page in the printer.

Adjust the paper so the printer is at the last line of the page. FLEX does a PCRLF before printing, so the first "printed" line will be line 1.

The command line for the disassembler is:

[P,] DISSEM,<file spec>[,<file spec>][+C]

(Use the "P" if you have a printer!!!)

The input file defaults to the working drive, .BIN extension. The output file defaults to the input file drive and name, but the extension is always .DSA (for DISAssembled). As noted above, the last letter (A) will increment if multiple files are created. The +C suppresses the cross reference printout.

For each section of the code, you will be asked the beginning and ending addresses of known data areas. (The ending address means the last byte of data, not the first byte of code which follows. Make as few or as many entries as you know or need.

PHILOSOPHY

The overriding concern in writing this disassembler was to have the computer do as much of the work as possible. Run time was a consideration, but not an overriding concern.

Some disassemblers start at the beginning of the code and try to disassemble it. When they run across a byte that isn't a legitimate opcode they print an FCB and continue.

The first problem is staying in sync. A BRA may be a space, and everything will be off until the disassembler hits the "right" sequence of invalid opcodes.

The second problem is finding data printed vertically for two pages.

These two problems can be avoided in different ways. One way is to have the operator (you) input address boundaries for data areas, which is great if you know where they are. Another is to have the disassembler distinguish data from code. A third would be a combination.

This disassembler is a combination. In simple cases it will distinguish data from code. However, in ambiguous situations, it will opt for code. If it makes a mistake, you can run it again, entering the newly discovered data areas. (Since Motorola chose 00 as a valid direct-page opcode, many direct page warnings indicate data!)

Disc input (vs. code in memory) was chosen to free up as much memory as possible for the tag tables. Any code in memory can be put to disc with the SAVE utility.

REQUIREMENTS

Since the disassembler runs under TSC's 6809 FLEX, you obviously need 6809 FLEX to run it. Before running, you will want to list the first couple pages and check the equates against the "Advanced Programmer's Guide". (If your system is in C000 - DFFF you're probably ok.)

The program is about 7k bytes long, and uses the memory from the end of the program to (MEMEND) for the various tag tables. As a guide, the disassembly of FLEX used an additional 6.5k, to 35DC, for the tags, and ran for one hour and two minutes (with a 60 LPM printer; output to 9600 baud CRT was much faster). If memory is too short for the cross reference, the cross reference is dropped. If memory is too short for the tag table, the disassembly is aborted (with an appropriate message).

HOW IT WORKS

The disassembler runs in four (+/-) passes.

The first pass creates all possible tags. It starts at the beginning of the file and creates temporary tags for all direct, indirect, extended and relative addressing modes. If the "code" terminates with a transfer of control (BRA, JMP, etc.), these tags are sorted into the permanent tag table. When the "code" terminates (absolute transfer, bad opcode, transfer to known data, end of file, or "backward" ORG), the process is repeated, starting one byte farther into the file. When this starting location is beyond the end of the file, the first pass terminates. We now have a table with every possible tag.

The second pass deletes all coding tags whose code contains either bad opcodes or transfers to "bad" tags. It is reexecuted until no more bad tags are found.

The third pass uses the refined tag table to decide whether the bytes are coding or data, and flags all tags that are referenced and/or in the tag column. This allows the fourth pass to print equates at the beginning (which allows the assembler to address direct addresses directly).

The fourth pass outputs the source to printer and disc. After the fourth pass, the cross reference prints. (In the case of multiple files, the whole process starts over with the next section.)

SUMMARY

This disassembler does most of the aggravating work involved in cracking object code.

The direct address warnings alert you to this possible problem (i.e., the tags are in page 00, but the DP register is somewhere else).

The sorted cross reference allows easy checking of tag usage. It is printed on separate pages for side-by-side use.

The use of FCC makes the listing shorter and easier to read. Any printable character, upper or lower case, will be FCC'd, except quotes (the delimiter).

Multiple files, if they are created, alert you to the possibility that earlier code may be overlaid by later code.

The listing contains both the input and output names in the heading so you know what it's a listing of.

And, best of all, the only price you pay is a slightly longer run time. In other words, the 6809 runs for twenty more minutes and saves you eight hours work, which is why you bought it!

William Stock
1125 Lois Dr.
Cincinnati, Ohio 45237
513-641-0181 after 6 pm Eastern time.

\- - -

```
       NAM   DISASSEMBLER FOR TSC 6809 FILES
* DISASSEMBLES 6809 CODE FROM
* TSC .BIN DISC FILE
       OPT

*
*
*  Call line:
*    (P) DISSEM <file-spec)[,<file-spec)][,+C]
*
*****
* Version 4 added user defined data areas.
*
* Version 5 corrected some minor bugs in 4:
*    added error checking of user data area
*       input;
*    put an * in column before tag to
*       indicate beginning of unreferenced
*       code (possible jump table);
*    checks for coding tags in user defined
*       areas before putting tag in temp
*       tag table (use less memory & run
*       faster).
*
* Version 6 doesn't examine known data areas
*     for possible code.
*
* Version 7 forces 66 lines per page if
*     printer is used (value at PDEPTH);
*   makes printing of cross-reference
*     optional (+C supresses print).
*
* Version 8 fixes the "first tag" problem
*     (is it data or is it code?).
*
* Version 9 fixes an "entry tag" problem.
*    Sometimes the entry tag will be non-
*    executable code (doesn't end with
*    absolute transfer).  If it's data,
*    it's disassembled as data.  You sort
*    it out.
*
*    Also automatically deletes old .DSx
*    files.
*

*
* MNEMONIC TABLE LAYOUT:
*  IIIIIBF WHERE IIIII = MNEMONIC
*          B = INFO BYTE AS FOLLOWS:
*  Tnnnbbb WHERE nnn = MODE
*                  1 = IS TRANSFER OF CONTROL
*                  bbb = # BYTES IN COMMAND
*                  T = T tag
*  nnn IS FURTHER DEFINED:
*     0 = INHERENT
*     1 = INDEXED
*     2 = DIRECT
*     3 = EXTENDED
*     4 = IMMEDIATE
*     5 = RELATIVE
*
*   F = FORMAT BYTE as follows:
*   bit on = print hex
*   bit off = print space.
*
*
* BIT DEFINITION OF 1ST BYTE OF TAG:
*  (D tags are data, T tags are code.  All
*     tags print as T(addr).)
*
*
*   TAG NOT IN    TAG IN
*   tag column  tag column
* D 0100 0100   1100 0100   tag not referenced
* T 0101 0100   1101 0100      don't EQUate
*
* d 0110 0100   1110 0100   tag
* t 0111 0100   1111 0100     referenced
*     EQUate    don't EQUate
*
* OPERAND LAYOUT in print line before space
*    supression:
*
*  0123456789ABCDE
*  (@-Tnnnn,--I++)
*     $       Y
*             U etc.
*
* FLEX SYSTEM EQUATES
C000 FLEX    EQU   $C000
C002 TTYEOL  EQU   FLEX+$0C02 *TTY END OF LINE CHAR
C003 DEPTH   EQU   FLEX+$0C03 *PAGE LENGTH
CC14 LNBFPT  EQU   FLEX+$0C14 *BUFFER POINTER
CC1A CURLN   EQU   FLEX+$0C1A *CURRENT LINE NUMBER
CC22 OUTSW   EQU   FLEX+$0C22 *CRT OR PRINTER
CC2B MEMEND  EQU   FLEX+$0C2B *END OF MEMORY
CD03 WARMS   EQU   FLEX+$0003 *FLEX REENTRY
CD0F OUTCH   EQU   FLEX+$000F *PRINTER OR CRT
CD12 OUTCH2  EQU   FLEX+$0D12 *CRT ONLY
CD15 GETCHR  EQU   FLEX+$0015 *GET CHAR FROM KBD
CD18 PUTCHR  EQU   FLEX+$001B *PUT CHAR TO WHATEVER
CD1B INBUFF  EQU   FLEX+$001B *GET INPUT BUFFER
CD1E PSTRNG  EQU   FLEX+$001E *PRINTS STRING
CD24 PCRLF   EQU   FLEX+$0024 *PRINTS CR/LF
CD27 NXTCH   EQU   FLEX+$0027 *NEXT CHR FROM BUFFER
CD2D GETFIL  EQU   FLEX+$002D *GETS FILE SPEC
CD33 SETEXT  EQU   FLEX+$0033 *SETS EXTENSION
CD3C OUTHEX  EQU   FLEX+$003C *PRINT 1 HEX BYTE
CD3F RPTERR  EQU   FLEX+$003F *REPORT FLEX ERROR
CD42 GETHEX  EQU   FLEX+$0042 *GET ADDR FROM BUFFER
CD45 OUTADR  EQU   FLEX+$0045 *PRINT 2 HEX BYTES
D403 FMSCLS  EQU   FLEX+$1403 *CLOSE ALL FILES
D406 FMS     EQU   FLEX+$1406 *CALL FMS
*
* FCB BYTE OFFSETS
0000 FUNC    EQU   0          *FUNCTION CODE
0001 STAT    EQU   1          *FILE STATUS
0003 DRV     EQU   3          *DRIVE NUMBER
0011 STRSEC  EQU   17         *START TRK & SECTOR
001E NEXT    EQU   30         *CURRENT POSITION
0022 DIDX    EQU   34         *DATA INDEX
0023 RIDX    EQU   35         *RANDOM DATA INDEX
0038 COMP    EQU   59         *COMPRESSION FLAG
0040 DATA    EQU   64         *256 BYTES OF SECTOR
*
* DATA AREAS
0000                  ORG   0
0000 42          PDEPTH FCB   66        *LINES / PRINTED PAGE
0001 0009              FDB   9          *VERSION
0003 1C77        DATST FDB   PGMEND     *HIGHEST DATA POINTER
0005 1C77        DATEND FDB  PGMEND
0007 1C77        TAGST FDB   PGMEND     *TAG TABLE POINTER
0009 1C7A        TAGEND FDB  PGMEND+3
000B 1C7D        ITGST FDB   PGMEND+6
000D 1C7D        ITGEND FDB  PGMEND+6
000F FFFF        XFRADD FDB  $FFFF      *TRANSFER ADDRESS
0011 00          CNTR  FCB   0          *BLOCK COUNTER
0012 00          DPFLAG FCB  0          *REAL D.P. FLAG
0013 0001        LINENO FDB  1          *CODING LINE #
0015 00          FCCCNT FCB  0          *CHAR CNT FOR "FCC"
0016 008A        FCCLOC FDB  BUFND+1    *CUR POSN OF "FCC"
0018 04          RRIDX FCB   4          *REWIND RIDX
0019 0000        HITOP FDB   0          *HIGH ADDRESS
001B 00          DEBUG FCB   0          *DUMP TAG FLAG
001C 00          ABORTF FCB  0          *ABORT FLAG
001D 00          DPFLAG FCB  0          *0 = PRINT DREF
001E             BIND  RMB   2          *BINARY SEARCH OFFSET
0020             ZAPFLG RMB  1          *NON-ZERO = ZAPPED TAG
0021             CODFLG RMB  1          *0 = DATA; NON-ZERO = CODE
0022             NONCTG RMB  1          *NON-CONTIGUOUS CODE
0023             BADOP RMB   1          *BAD OPCODE
0024             OFER  RMB   1          *BRA, JMP, ETC.
0025             LASTQ RMB   2          *LAST QUESTIONABLE ADDR
0027             LSTBLK RMB  2
0029             SAVCMD RMB  5          *COMMAND WE'RE WORKING ON
002E             CMDCNT RMB  1          * # BYTES IN COMMAND
002F             BYTCNT RMB  1          *BYTE COUNT
0030             MODE  RMB   1          *ADDRESSING MODE
0031             HEXFMT RMB  1          *PRINT FORMAT FOR OBJECT
```

```
0032                    TAGBLD  RMB     3       *TAG BUILD AREA
0035                    GOADDR  RMB     2       *LAST GOOD ADDRESS
0037                    GDTRK   RMB     2       *LAST GOOD TRACK/SECTOR
0039                    GDRIDX  RMB     1       *WHERE RIDX WERE
003A                    GDCNTR  RMB     1       *WHAT THE CNTR WERE
003B                    CURTAG  RMB     2       *CURRENT TAG ADDRESS
003D                    P2TAG   RMB     2       *MAYBE CURTAG
003F                    MSEC    RMB     2       *SECTOR WHERE 02 STARTS
0041                    MRIDX   RMB     1       *RIDX OF 02
0042                    IRST    RMB     2       *IREF START
0044                    IREND   RMB     2       *IREF END
0046                    IRTAG   RMB     3       *IREF WORK AREA
0049                    TTYDEP  RMB     1       *ORIGINAL DEPTH COUNT
004A                    PLINE   RMB     79      *PRINT LINE
                004A    PLNB    EQU     PLINE
                0050    PADDR   EQU     PLINE+6
                0055    PHEX    EQU     PLINE+11
                0063    PTAG    EQU     PLINE+25
                0069    POPCD   EQU     PLINE+31
                006F    POPND   EQU     PLINE+37
                0088    PASCII  EQU     PLINE+62
                008E    PCOM    EQU     PLINE+68
0099 04                 HEAD4   FCB     4
009A                    DLINE   RMB     56
                009A    DADDR   EQU     DLINE   *"FCC" PRINT LINE
                009F    DHEX    EQU     DLINE+5
                00AD    DTAG    EQU     DLINE+19
                00B3    DOPCD   EQU     DLINE+25
                00B7    DOPND   EQU     DLINE+31
0092 01                 FCB     1
0093 44 69 73 61 HEAD   FCC     "Disassembly of "
0097 73 73 65 6D
009B 62 6C 79 20
009F 6F 66 20
00E2 32 20 20 20 HFILE  FCC     "2                "
00E6 20 20 20 20
00EA 20 20 20 20
00EE 20 20 20 20
00F2 20
00F3 6F 75 74 70        FCC     "output = "
00F7 75 74 20 3D
00FB 20
00FC 32 20 20 20 HOFILE FCC     "2                "
0100 20 20 20 20
0104 20 20 20 20
0108 20 20 20 20
010C 20
0108 50 41 47 45        FCC     "PAGE "
0111 20
0112                    HPAGE   RMB     1
0116 04                 FCB     4
0117 0000               HPGND   FDB     0
0119                    FCBIN   RMB     320
0259                    FCBOUT  RMB     320

0399 4F                 START   CLRA            *SET DIRECT PAGE
039A 1F  8B                     TFR     A,DP
039C 17  1808                   LBSR    PRELIM  *DO DISC SETUP
039F 1025 0091                  LBCS    FINI    *ERRORS! ABORT
03A3 9E  07             START1  LDX     TAGST   *START DUMMY TAG
03A5 86  44                     LDA     #'D
03A7 A7  80                     STA     0,X+
03A9 CC  FFFF                   LDD     #$FFFF
03AC ED  84                     STD     0,X
03AE DD  0F                     STD     XFRADD  *ASSUME NO XFER
03B0 17  17FF                   LBSR    GETDAT  *GET KNOWN DATA AREAS
03B3 86  54                     LDA     #'T     *START DUMMY TAG
03B5 A7  9F 0007                STA     [TAGST]
03B9 17  087E                   LBSR    PASS1   *1ST PASS MAKES TAGS
03BC 25  48                     BCS     CLOSE
03BE 9E  00                     LDX     TTGST   * "END START" TAG
03C0 86  54                     LDA     #'T
03C2 A7  80                     STA     0,X+
03C4 DC  0F                     LDD     XFRADD
03C6 1083 FFFF                  CMPD    #$FFFF  *PRELOADED, NOT VALID
03CA 27  07                     BEQ     CONT4
03CC ED  81                     STD     0,X++
03CE 9F  00                     STX     TTGEND
03D0 17  0AF4                   LBSR    RIPPLE
03D3 0D  18             CONT4   TST     DEBUG
03D5 27  03                     BEQ     CONT1
03D7 17  1722                   LBSR    DUMP
03DA 17  0106           CONT1   LBSR    PASS2   *2ND PASS ELIMINATES
03DD 25  27                     BCS     CLOSE   *'T' TAGS WHICH
03DF 0D  20                     TST     ZAPFLG  *WON'T EXECUTE
03E1 27  F0                     BEQ     CONT4
03E3 DC  0F                     LDD     XFRADD
03E5 1083 FFFF                  CMPD    #$FFFF
03E9 27  09                     BEQ     CONT2   *NO ENTRY TAG
03ED 17  0892                   LBSR    FINDTG  *ENTRY TAG IS REFERENCED
03EE A6  84                     LDA     0,X
03F0 8A  20                     ORA     #%00100000
03F2 A7  84                     STA     0,X
03F4 17  01A6           CONT2   LBSR    PASS3   *DOCTOR THE TAGS
03F7 25  00                     BCS     CLOSE   *  TO REFLECT ACTUAL USE
03F9 17  176E                   LBSR    TAGCLN
03FC 0D  18                     TST     DEBUG
03FE 27  03                     BEQ     CONT3
0400 17  16F9                   LBSR    DUMP
0403 17  0215           CONT3   LBSR    PASS4   *4TH PASS OUTPUTS CODE
0406 BD  CD24           CLOSE   JSR     PCRLF
0409 8E  0259                   LDX     #FCBOUT
040C A6  04                     LDA     #4
040E A7  84                     STA     FUNC,X
0410 BD  0406                   JSR     FMS
0413 8E  0119                   LDX     #FCBIN

0416 DC  3F                     LDD     MSEC
0418 27  08                     BEQ     FINIA   *END OF FILE
041A 0D  1C                     TST     ABORTF
041C 26  07                     BNE     FINIA   *TAG TABLE OVERFLOW
041E 17  0505                   LBSR    REPASS  *BACKWARD ORG
0421 1027 FF7E                  LBEQ    START1  *DO NEXT SECTION
0425 8E  0119           FINIA   LDX     #FCBIN  *CLOSE FILES
0428 86  04                     LDA     #4
042A A7  84                     STA     FUNC,X
042C BD  0406                   JSR     FMS
042F 96  49                     LDA     TTYDEP
0431 87  CC03                   STA     DEPTH   *RESET DEPTH
0434 BD  0403           FINI    JSR     FMSCLS  *CLOSE ALL FILES
0437 7E  CD03                   JMP     WARMS   *EXIT TO FLEX

                        *
                        *PASS 1 BUILDS ALL POSSIBLE TAGS.
                        *IT FIRST BUILDS TEMPORARY TAGS FOR ONE
                        *SECTION OF CODE.  IF THE SECTION TURNS
                        *OUT GOOD, IT SAVES THE TEMP TAGS.
                        *IF THE SECTION IS BAD, IT DUMPS THEM.
                        *
                        *IT CONTINUES DOING THIS UNTIL ALL
                        *POSSIBLE TAGS HAVE BEEN BUILT.
                        *
043A 17  0AD9           PASS1   LBSR    REWIND
043D DC  19                     LDD     MXTOP
043F DD  25                     STD     LASTQ   *START HERE AND
0441 109E 0B                    LDY     TTGST
0444 ED  21                     STD     1,Y
0446 86  54                     LDA     #'T     *CODE, HOPEFULLY
0448 A7  A4                     STA     0,Y
044A 31  23                     LEAY    3,Y
044C 109F 00                    STY     TTGEND
044F EC  88 1E          PIO     LDD     NEXT,X  *QUICK ACCESS TO LAST
0452 DD  37                     STD     GDTRK   * ORGENT CODE
0454 A6  88 23                  LDA     RIDX,X
0457 97  39                     STA     GDRIDX
0459 96  1E                     LDA     CNTR
045B 97  3A                     STA     GDCNTR
045D DC  19             PIB     LDD     MXTOP
045F BD  64                     BSR     P1I
0461 0D  23                     TST     BADOP
0463 26  23                     BNE     PIC
0465 17  0530                   LBSR    DCDCMD  *DECODE OPCODE
0468 0D  1C                     TST     ABORTF  *CHECK TTAG OVFLO
046A 26  56                     BNE     P1H
046C 8E  0119                   LDX     #FCBIN
046F A6  01                     LDA     STAT,X
0471 27  06                     BEQ     P1F     *CONTINUE W/ NEXT CMD
0473 81  08                     CMPA    #8
0475 26  48                     BNE     P1E     *ERROR
0477 20  0F                     BRA     P1C     *EOF: BACK UP
0479 0D  22             P1F     TST     NONCTG
047B 26  08                     BNE     P1C     *NON-CONTIG ABURTS TAGS
047D 0D  23                     TST     BADOP
047F 26  07                     BNE     P1C     *BAD OPCODE ABURTS TAGS
0481 0D  24                     TST     XFER
0483 27  D8                     BEQ     P1B     *NEXT COMMAND
0485 17  0A3F                   LBSR    RIPPLE  *ABSOLUTE TRANSFER
0488 86  54             P1C     LDA     #'T
048A A7  9F 0008                STA     [TTGST]
048E DC  25                     LDD     LASTQ
0490 C3  0001                   ADDD    #1
0493 8D  30                     BSR     P1I
0495 DD  25                     STD     LASTQ   *BUMP LAST QUESTIONABLE
0497 17  08AA                   LBSR    FIND
049A 25  15                     BCS     P1A
049C 6A  88 23                  DEC     RIDX,X
049F 0C  11                     INC     CNTR
04A1 DD  25                     STD     LASTQ
04A3 DD  35                     STD     GOADDR
04A5 109E 08                    LDY     TTGST
04A8 31  21                     LEAY    1,Y
04AA ED  A1                     STD     0,Y++
04AC 109F 09                    STY     TTGEND
04AF 20  9E                     BRA     P1O     *FOUND IT; GO PROCESS
                        *
04B1 8E  0119           P1A     LDX     #FCBIN
04B4 A6  01                     LDA     STAT,X
04B6 27  04                     BEQ     P1G
04B8 81  08                     CMPA    #8      *IS STATUS EOF?
04BA 26  03                     BNE     P1E     *NO
04BC 1C  FE             P1G     CLC             *YES
04BE 39                         RTS
04BF BD  CD0F           P1E     JSR     RPTERR
04C2 1A  01             P1H     SEC
04C4 39                         RTS
04C5 9E  03             P1I     LDX     DATST   *SEE IF DATA AREA
04C7 0F  23                     CLR     BADOP
04C9 9C  05             P1J     CMPX    DATEND
04CB 27  11                     BEQ     P1L     *NO MORE KNOWN DATA
04CD 10A3 84                    CMPD    0,X
04D0 25  00                     BCS     P1K     *NOT THIS GROUP
04D2 10A3 02                    CMPD    2,X
04D5 22  08                     BHI     P1K     *NOT THIS GROUP
04D7 EC  02                     LDD     2,X     *DATA
04D9 C3  0001                   ADDD    #1
04DC 0C  23                     INC     BADOP
04DE 39                 P1L     RTS
04DF 30  04             P1K     LEAX    4,X
04E1 20  E6                     BRA     P1J

04E3 17  0A30           PASS2   LBSR    REWIND  *ELIMINATES ENTRY "T" TAGS
04E6 1025 009F                  LBCS    P2A     *CONTAINING BAD XFERS
04EA 86  FF                     LDA     #$FF    *(CONDITIONAL OR NOT)
04EC 97  20                     STA     ZAPFLG
```

```
04EE 17  0A75  P28    LBSR  NEXTT    *GET NEXT CODING TAG
04F1 1025 0094        LBCS  P2A
04F5 17   0846        LBSR  FIND     *FIND CODE TO MATCH
04F8 1025 0080        LBCS  P2A      *CHECK FOR END; TAGS
04FC 26   F0          BNE   P2B      * NOT CODED ARE GOOD
04FE 0C   11          INC   CNTR
0500 6A   88 23       DEC   RIDX,X
0503 DC   19          LDD   NXTOP    *SAVE LOCN FOR NEXT
0505 DD   35          STD   GOADDR
0507 EC   88 1E       LDD   NEXT,X
050A DD   37          STD   GOTRK
050C A6   88 23       LDA   RIDX,X
050F 97   39          STA   GDRIDX
0511 96   11          LDA   CNTR
0513 97   3A          STA   GDCNTR
0515 DC   38          LDD   CURTAG
0517 DD   3B          STD   P2TAG    *DYNAMIC CURTAG
0519 17   047C  P2C   LBSR  DISCMD
051C 8E   0119        LDX   #FCBIN
051F A6   01          LDA   STAT,X
0521 27   06          BEQ   P2D
0523 81   08          CMPA  #8
0525 27   28          BEQ   P2E
0527 20   60          BRA   P2A
0529 00   23  P2D     TST   BADOP
052B 26   25          BNE   P2E      *ABORT TAGS
052D 00   22          TST   NOMCTG
052F 26   21          BNE   P2E      *ABORT TAGS
0531 96   32          LDA   TAGBLD
0533 81   54          CMPA  #'T
0535 26   28          BNE   P2F
0537 96   30          LDA   MODE
0539 81   10          CMPA  #$10     *INDIRECT ALWAYS VALID
053B 27   25          BEQ   P2F
053D DC   33          LDD   TAGBLD+1
053F 1083 FFFF        CMPD  #$FFFF   *PRE-SET VALUE
0543 27   10          BEQ   P2F
0545 17   0A3B        LBSR  FINDTG   *FIND TAG
0548 25   08          BMS   P2E      *NOT FOUND
054A A6   84          LDA   0,X
054C 84   54          ANDA  #'T
054E 81   54          CMPA  #'T
0550 27   10          BEQ   P2F      *GOT GOOD TAG
0552 9E   38  P2E     LDX   CURTAG
0554 86   44          LDA   #'D      *REASSURE DATA
0556 A7   84  P2E1    STA   0,X      *FOR THIS TAG
0558 30   03          LEAX  3,X
055A 9C   3D          CMPX  P2TAG    *AND ANY SUBSEQUENT 'T'S
055C 23   FB          BLS   P2E1     *IN THIS GROUP
055E 0F   20          CLR   ZAPFLG   *SHOW WE DID SOMETHING
0560 20   BC          BRA   P2B      *AND TRY THIS GUY
0562 9E   3D  P2F     LDX   P2TAG
0564 00   24          TST   XFER
0566 27   04          BEQ   P2F1     *NOT END
0568 9F   38          STX   CURTAG   *INTERMEDIATE TAGS
056A 20   82          BRA   P2B      * ALREADY VALIDATED'
                      *
056C 30   03  P2F1    LEAX  3,X
056E 9C   09          CMPX  TAGEND
0570 27   A7          BEQ   P2C      *NO MORE
0572 EC   01          LDD   1,X
0574 1093 19          CMPD  NXTOP    *NEXT TAG = NEXT ADDR?
0577 25   06          BCS   P2F2     *NO. BUT OK IF DATA
0579 26   9E          BNE   P2C      *NO. QUIT
057B 9F   3D          STX   P2TAG    *YES: VERIFY THIS GUY TOO
057D 20   9A          BRA   P2C
057F A6   84  P2F2    LDA   0,X
0581 84   54          ANDA  #'T
0583 81   44          CMPA  #'D
0585 27   E5          BEQ   P2F1     *DATA TAG: OK
0587 20   90          BRA   P2C      *CODE TAG: END TEMP VERIFY
0589 8E   0119  P2A   LDX   #FCBIN
058C A6   01          LDA   STAT,X
058E 27   0A          BEQ   P2G
0590 81   08          CMPA  #8
0592 27   06          BEQ   P2G
0594 8D   CD3F        JSR   RPTERR
0597 1A   01          SEC
0599 39               RTS
059A 1C   FE  P2G     CLC
059C 39               RTS

                      *
                      *PASS 3 DISASSEMBLES WITHOUT OUTPUT,
                      *FLAGGING ALL TAGS WHICH ARE REFERENCED,
                      *IN TAG COLUMN, OR FIRST CODING TAG
                      *IN TAG COLUMN AFTER ABSOLUTE TRANSFER
                      *WHETHER REFERENCED OR NOT (SO PASS4
                      *WILL KNOW WHETHER TO DO CODE OR DATA).
                      *
059E 17   0976  PASS3  LBSR  REWIND   *REWIND
05A1 25   68          BCS   P3A      *ABORT
05A3 0F   12          CLR   OPFLAG
05A5 0F   21          CLR   CODFLG   *DEFAULT IST TIME
05A7 DC   08  P3C     LDD   TTGST    *AVOID BOMB
05A9 DD   0B          STD   TTGEND
05AB 17   07D0        LBSR  READ
05AE 26   58          BNE   P3A
05B0 DC   19          LDD   NXTOP
05B2 17   09CC        LBSR  FINDTG   *FIND THIS TAG
05B5 25   14          BCS   P3D      *NOT FOUND
05B7 A6   84          LDA   0,X
05B9 8A   80          ORA   #$80     *TAG IN TAG COLUMN
05BB A7   84          STA   0,X
05BD 84   54          ANDA  #'T
05BF 81   54          CMPA  #'T
05C1 26   08          BNE   P3D
05C3 00   21          TST   CODFLG
05C5 26   04          BNE   P3D
05C7 6C   84          INC   0,X      *FLAG NO-DELETE
05C9 97   21          STA   CODFLG   *NOT-0 = DO CODE
05CB 00   21  P3D     TST   CODFLG
05CD 26   09          BNE   P3E      *DO CODE
05CF DC   19          LDD   NXTOP
05D1 C3   0001        ADDD  #1
05D4 DD   19          STD   NXTOP
05D6 20   CF          BRA   P3C      *END DATA
05D8 0C   11  P3E     INC   CNTR
05DA 8E   0119        LDX   #FCBIN
05DD 6A   88 23       DEC   RIDX,X
05E0 17   03B6        LBSR  DCBCMD
05E3 00   24          TST   XFER
05E5 27   02          BEQ   P3H
05E7 0F   21          CLR   CODFLG
05E9 8E   0119  P3H   LDX   #FCBIN
05EC 60   01          TST   STAT,X
05EE 26   18          BNE   P3A      *ABORT OR END
05F0 DC   33          LDD   TAGBLD+1
05F2 1083 FFFF        CMPD  #$FFFF
05F6 27   08          BEQ   P3F      *TAG NOT BUILT
05F8 17   0986        LBSR  FINDTG
05FB 25   06          BCS   P3F      *SHOULD BE DISASTER
05FD A6   84          LDA   0,X
05FF 8A   20          ORA   #$20     *TAG REFERENCED
0601 A7   84          STA   0,X
0603 00   24  P3F     TST   XFER
0605 27   A0          BEQ   P3C
0607 0F   21          CLR   CODFLG
0609 20   9C          BRA   P3C
060B 8E   0119  P3A   LDX   #FCBIN
060E A6   01          LDA   STAT,X
0610 27   09          BEQ   P3G
0612 81   08          CMPA  #8
0614 27   05          BEQ   P3G
0616 8D   CD3F        JSR   RPTERR
0619 1A   01          SEC
061B 39               RTS
061C 26   08  P3G     BNE   P3D
061E 00   21          TST   CODFLG
0620 26   04          BNE   P3D
0622 6C   84          INC   0,X      *FLAG NO-DELETE
0624 97   21          STA   CODFLG   *NOT-0 = DO CODE
0626 00   21  P3D     TST   CODFLG
0628 26   09          BNE   P3E      *DO CODE
062A DC   19          LDD   NXTOP
062C C3   0001        ADDD  #1
062F DD   19          STD   NXTOP
0631 20   CF          BRA   P3C      *END DATA
0633 0C   11  P3E     INC   CNTR
0635 8E   0119        LDX   #FCBIN
0638 6A   88 23       DEC   RIDX,X
063B 17   03B6        LBSR  DCBCMD
063E 00   24          TST   XFER
0640 27   02          BEQ   P3H
0642 0F   21          CLR   CODFLG
0644 8E   0119  P3H   LDX   #FCBIN
0647 60   01          TST   STAT,X
0649 26   18          BNE   P3A      *ABORT OR END
064B DC   33          LDD   TAGBLD+1
064D 1083 FFFF        CMPD  #$FFFF
0651 27   08          BEQ   P3F      *TAG NOT BUILT
0653 17   0986        LBSR  FINDTG
0656 25   06          BCS   P3F      *SHOULD BE DISASTER
0658 A6   84          LDA   0,X
065A 8A   20          ORA   #$20     *TAG REFERENCED
065C A7   84          STA   0,X
065E 00   24  P3F     TST   XFER
0660 27   A0          BEQ   P3C
0662 0F   21          CLR   CODFLG
0664 20   9C          BRA   P3C
0666 8E   0119  P3A   LDX   #FCBIN
0669 A6   01          LDA   STAT,X
066B 27   09          BEQ   P3G
066D 81   08          CMPA  #8
066F 27   05          BEQ   P3G
0671 8D   CD3F        JSR   RPTERR
0674 1A   01          SEC
0676 39               RTS
0677 1C   FE  P3G     CLC
0679 39               RTS

                      *PASS 4 OUTPUTS THE .OSA FILE AND
                      *PRINTS THE LISTING.
                      *AT THE END IT PRINTS THE CROSS
                      *REFERENCE.
                      *
067A 17   09E7  PASS4  LBSR  CLRPLN
067D 8E   1250        LDX   #NAMS
0680 108E 0069        LDY   #PGPCD
0684 C6   03          LDB   #3
0686 A6   80  P4A     LDA   0,X+
0688 A7   A0          STA   0,Y+
068A 5A               DECB
068B 26   F9          BNE   P4A      *PUT NAM &
068D 31   23          LEAY  3,Y
068F 8E   0259        LDX   #FCBOUT  *FILENAME
0692 30   04          LEAX  DRV+1,X
0694 C6   08          LDB   #11
0696 A6   80  P4B     LDA   0,X+
0698 27   02          BEQ   P4C
069A A7   A0          STA   0,Y+
069C 5A               DECB
069D 26   F7  P4C     BNE   P4B
069F 17   09CF        LBSR  PRINT    *PRINT PLINE
06A2 1026 0299        LBNE  WRTERR
06A6 17   0988        LBSR  CLRPLN
06A9 8E   1270        LDX   #OPTS    *ADD OPT PAG
06AC 108E 0069        LDY   #PGPCD
06B0 C6   09          LDB   #9
06B2 A6   80  P4C1    LDA   0,X+
06B4 A7   A0          STA   0,Y+
06B6 5A               DECB
06B7 26   F9          BNE   P4C1
06B9 17   09B5        LBSR  PRINT
06BC 1026 027F        LBNE  WRTERR
06C0 0D   12          TST   OPFLAG
06C2 27   34          BEQ   P4F
06C4 17   0990        LBSR  CLRPLN   *DIRECT PAGE WARNING
06C7 8E   1230        LDX   #OPMSG2
06CA 108E 0063        LDY   #PTAG
06CE C6   26          LDB   #38
06D0 A6   80  P4D     LDA   0,X+
06D2 A7   A0          STA   0,Y+
06D4 5A               DECB
06D5 26   F9          BNE   P4D
06D7 17   0997        LBSR  PRINT
06DA 1026 0261        LBNE  WRTERR
06DE 17   0983        LBSR  CLRPLN
06E1 8E   1256        LDX   #OPMSG3
06E4 108E 0069        LDY   #PGPCD
06E8 C6   07          LDB   #7
06EA A6   80  P4E     LDA   0,X+
06EC A7   A0          STA   0,Y+     *FAKE SETUP 0
06EE 5A               DECB
06EF 26   F9          BNE   P4E
06F1 17   0970        LBSR  PRINT
06F4 1026 0247        LBNE  WRTERR
06F8 17   0969  P4F   LBSR  CLRPLN   *PRINT TAG EQUATES
06FB 9E   07          LDX   TAGST
06FD 9C   09  P4G     CMPX  TAGEND
0700 27   4F          BEQ   P4J      *DONE WITH EQUATES
0702 A6   80          LDA   0,X+
0704 2B   47          BMI   P4I      *TAG IN COLUMN
0706 85   20          BITA  #%00100000
0708 27   43          BEQ   P4I      *TAG NOT REFERENCED
```

```
06AA 86 54          LDA  #'T      *ALL TAGS PRINT "T"
06AC 97 63          STA  PTAG
06AE 8A 80          ORA  #$80     *XREF TAG
0680 97 46          STA  XRTAG
0682 86 24          LDA  #'$
0684 97 6F          STA  POPND
0686 A6 84          LDA  0,X
0688 97 47          STA  XRTAG+1
068A 17 079E        LBSR HEXASC
068D DD 64          STD  PTAG+1
068F DD 70          STD  POPND+1
06C1 A6 01          LDA  1,X
06C3 97 48          STA  XRTAG+2
06C5 17 0793        LBSR HEXASC
06C8 DD 66          STD  PTAG+3
06CA DD 72          STD  POPND+3
06CC 34 10          PSHS X
06CE 17 09C0        LBSR XRSORT   *SORT INTO XREF
06D1 8E 1267        LDX  #EQUS
06D4 108E 0069      LDY  #OPCD
06D8 C6 03          LDB  #3
06DA A6 80    P4H   LDA  0,X+
06DC A7 A0          STA  0,Y+
06DE 5A             DECB
06DF 26 F9          BNE  P4H
06E1 17 092E        LBSR PRINT
06E4 1026 01F8      LBNE WRTERR
06E8 17 091A        LBSR CLRPLN
06EB 35 10          PULS X
06ED 30 02    P41   LEAX 2,X
06EF 20 AD          BRA  P4G
06F1 17 0822  P4J   LBSR REWIND   *REWIND FILE
06F4 1025 01DD      LBCS P4END    *ERROR
06F8 17 0887        LBSR DOORG    *ASSUME DATA
06FB 0F 21          CLR  CODFLG
06FD 17 145F        LBSR CLRDLN

                    *
                    * THIS IS THE MAIN LOOP
                    *
0700 DC 08    P4K   LDD  TTGST
0702 DD 00          STD  TTGEND
0704 0F 22          CLR  MONCTG
0706 17 08FC        LBSR CLRPLN
0709 17 0671        LBSR READ
070C 1026 01C5      LBNE P4END
0710 97 29          STA  SAVCHR
0712 00 22          TST  MONCTG
0714 27 DA          BEQ  P4K1
0716 17 141F        LBSR FCCPRT   *PRINT "FCC" BEFORE "ORG"
0719 1026 01C9      LBNE WRTERR
071C 17 0892        LBSR DOORG    *MONCTG = NEW ORG STATEMENT
071F DC 19    P4K1  LDD  NXTOP
0722 17 085B        LBSR FINDTG   *A TAG IN COLUMN?
0725 25 46          BCS  P4M      *NO TAG
0727 A6 84          LDA  0,X
0729 84 7F          ANDA #$7F
072B 81 44          CMPA #'D
072D 27 3E          BEQ  P4M      *DATA TAG NOT REF'D
072F 34 02          PSHS A
0731 17 1404        LBSR FCCPRT   *IF TAG, FLUSH
0734 35 02          PULS A
0736 1026 01A6      LBNE WRTERR   *DISC ERROR
073A 81 64          CMPA #'d
073C 27 0C          BEQ  P4L
073E 97 21          STA  CODFLG
0740 81 54          CMPA #'T
0742 26 06          BNE  P4L      *REF'D CODING TAG
0744 86 2A          LDA  #'*
0746 97 62          STA  PTAG-1
0748 20 23          BRA  P4M

074A B6 54    P4L   LDA  #'T
074C 97 63          STA  PTAG
074E 97 60          STA  DTAG
0750 8A 80          ORA  #$80
0752 97 46          STA  XRTAG    *XREF TAG
0754 96 19          LDA  NXTOP
0756 97 47          STA  XRTAG+1
0758 17 0700        LBSR HEXASC
075B DD 64          STD  PTAG+1   *STORE TAG IN PLINE
075D DD AE          STD  DTAG+1   *  AND DLINE
075F 96 1A          LDA  NXTOP+1
0761 97 48          STA  XRTAG+2
0763 17 06F5        LBSR HEXASC
0766 DD 66          STD  PTAG+3
0768 DD B0          STD  DTAG+3
076A 17 0931        LBSR XRSORT
076D 00 21    P4M   TST  CODFLG
076F 1026 00AM      LBNE P4P      *DO CODE
0773 96 15    P4N1  LDA  FCCCNT
0775 81 14          CMPA #20
0777 26 07          BNE  P4N2
0779 17 138C        LBSR FCCPRT
077C 1026 0160      LBNE WRTERR
0780 96 29    P4N2  LDA  SAVCHR
0782 2B 54          BMI  P4N      *NOT PRINTABLE
0784 81 20          CMPA #$20
0786 25 50          BCS  P4N      *NOT PRINTABLE
0788 81 22          CMPA #$22
078A 27 4C          BEQ  P4N      *DELIMITER
078C 81 5B          CMPA #$5B
078E 25 08          BCS  P4N6     *OK
0790 81 61          CMPA #$61
0792 25 44          BCS  P4N      *WEIRD ASCII: NO PRINT
0794 81 7B          CMPA #$7B
0796 24 40          BCC  P4N      *WEIRD ASCII: NO PRINT

0798 9E 16    P4N6  LDX  FCCLOC   *PUT CHAR IN FCC LINE
079A A7 80          STA  0,X+
079C 9F 16          STX  FCCLOC
079E 00 15          TST  FCCCNT
07A0 26 1E          BNE  P4N5
07A2 96 19          LDA  NXTOP
07A4 17 0684        LBSR HEXASC
07A7 DD 9A          STD  DADDR
07A9 96 1A          LDA  NXTOP+1
07AB 17 06AD        LBSR HEXASC
07AE DD 9C          STD  DADDR+2
07B0 8E 1274        LDX  #FCCS
07B3 108E 0083      LDY  #$0083
07B7 C6 07          LDB  #7
07B9 A6 80    P4N4  LDA  0,X+
07BB A7 A0          STA  0,Y+
07BD 5A             DECB
07BE 26 F9          BNE  P4N4
07C0 D6 15    P4N5  LDB  FCCCNT   *FIRST 4 HEX
07C2 C1 04          CMPB #4
07C4 24 0E          BCC  P4N3
07C6 8E 009F        LDX  #HEX
07C9 86 03          LDA  #3
07CB 30             MUL
07CC 3A             ABX
07CD 96 29          LDA  SAVCHR
07CF 17 0689        LBSR HEXASC
07D2 ED 84          STD  0,X
07D4 0C 15    P4N3  INC  FCCCNT
07D6 20 35          BRA  P401
07D8 17 1350  P4N   LBSR FCCPRT
07DB 1026 0101      LBNE WRTERR
07DF 96 29          LDA  SAVCHR
07E1 17 0677        LBSR HEXASC
07E4 DD 70          STD  POPND+1  *STASH HEX
07E6 DD 55          STD  PHEX
07E8 96 19          LDA  NXTOP
07EA 17 066E        LBSR HEXASC
07ED DD 50          STD  PADDR
07EF 96 1A          LDA  NXTOP+1
07F1 17 0667        LBSR HEXASC
07F4 DD 52          STD  PADDR+2
07F6 8E 126A        LDX  #FCB5
07F9 108E 0069      LDY  #OPCD
07FD C6 07          LDB  #7
07FF A6 80    P40   LDA  0,X+     *STASH "FCB"
0801 A7 A0          STA  0,Y+
0803 5A             DECB
0804 26 F9          BNE  P40
0806 17 0809        LBSR PRINT    *PRINT THE LINE
0809 1026 0003      LBNE WRTERR
080D DC 19    P401  LDD  NXTOP
080F C3 0001        ADDD #1       *BUMP NEXT ADDRESS
0812 DD 19          STD  NXTOP
0814 16 FEE9        LBRA P4K      *BACK FOR MORE
0817 17 131E  P4P   LBSR FCCPRT   *DO CODE
081A 1026 00C2      LBNE WRTERR
081E 17 133E        LBSR CLRDLN
0821 96 19          LDA  NXTOP
0823 17 0635        LBSR HEXASC
0826 DD 50          STD  PADDR
0828 96 1A          LDA  NXTOP+1
082A 17 062E        LBSR HEXASC
082D DD 52          STD  PADDR+2
082F 0C 11          INC  CNTR
0831 8E 0119        LDX  #FCBIN
0834 6A 88 23       DEC  [0,X]
0837 17 015E        LBSR GETCMD   *STASHES OPCD & OPND
083A 8E 0119        LDX  #FCBIN
083D 6D 01          TST  STAT,X
083F 1026 0092      LBNE P4END
0843 DC 33          LDD  TAGBLD+1 *HAVE TO DO TAG
0845 1083 FFFF      CMPD #$FFFF   *INVALID VALUE, HENCE
0849 27 20          BEQ  P40      *NO TAG BUILT
084B 17 0732        LBSR FINDTG
084E A6 84          LDA  0,X
0850 86 54          LDA  #'T
0852 97 72          STA  POPND+3
0854 97 46          STA  XRTAG    *XREF
0856 96 33          LDA  TAGBLD+1
0858 97 47          STA  XRTAG+1
085A 17 05FE        LBSR HEXASC
085D DD 73          STD  POPND+4  *STORE TAG
085F 96 34          LDA  TAGBLD+2
0861 97 48          STA  XRTAG+2
0863 17 05F5        LBSR HEXASC
0866 DD 75          STD  POPND+6
0868 17 0833        LBSR XRSORT
086B 8E 006F  P40   LDX  #OPOPND  *SQUEEZE SPACES
086E 1F 12          TFR  X,Y      *FROM OPERAND
0870 C6 19          LDB  #25
0872 A6 80    P4R   LDA  0,X+
0874 81 A0          CMPA #$A0
0876 27 02          BEQ  P4S      *ACH! A SPACE!
0878 A7 A0          STA  0,Y+
087A 5A       P4S   DECB
087B 26 F5          BNE  P4R
087D B6 A0          LDA  $A0
087F 10BC 0088 P4T  CMPY #OPOPND+25
0883 27 04          BEQ  P4U
0885 A7 A0          STA  0,Y+
0887 20 F6          BRA  P4T
0889 8E 0054  P4U   LDX  #PHEX-1  *STASH THE OBJECT
088C 108E 0029      LDY  #SAVCHR
0890 D6 31          LDB  NEXFNT
0892 30 01    P4V   LEAX 1,X
```

```
0894 58          P4W    LSLB
0895 27 0F              BEQ    P4W1      *ALMOST DONE
0897 24 F9              BCC    P4V
0899 A6 A0              LDA    0,Y+
089B 34 04              PSHS   B
089D 17 0588            LBSR   HEXASC
08A0 ED 81              STD    0,X++     *STASH DECODED BYTE
08A2 35 04              PULS   B
08A4 20 EE              BRA    P4W
08A6 A6 A4     P4W1     LDA    0,Y
08A8 17 0580            LBSR   HEXASC
08AB ED 84              STD    0,X
08AD 8E 0029    P4I     LDX    #SAVCMD   *PUT ASCII IN LINE
08B0 108E 0088          LDY    #PASCII
08B4 D6 2E              LDB    #NOCNT
08B6 A6 80      P4Y     LDA    0,X+
08B8 2B 04              BMI    P4Z       *NOT ASCII
08BA 81 20              CMPA   #$20
08BC 24 02              BCC    P4AA      *GOOD
08BE 86 20      P4Z     LDA    #$20      *NOT PRINTABLE
08C0 A7 A0      P4AA    STA    0,Y+
08C2 5A                 DECB
08C3 26 F1              BNE    P4Y
08C5 17 074A            LBSR   PRINT     *LINE IS FINISHED
08C8 26 16              BNE    WRTERR
08CA 0D 24              TST    IFER
08CC 1027 FE30          LBEQ   P4K
08D0 0F 21              CLR    CODFLG
08D2 16 FE2B            LBRA   P4K
                *  END OF MAIN LOOP

08D5 8E 0119    P4END   LDX    #FCBIN    *TEST EOF OR ERROR
08D8 A6 01              LDA    STAT,X
08DA 27 08              BEQ    P4END1
08DC 81 08              CMPA   #8
08DE 27 04              BEQ    P4END1    *NORMAL
08E0 BD C03F    WRTERR  JSR    RPTERR    *EITHER FILE!
08E3 39                 RTS
                *
08E4 17 1251    P4END1  LBSR   FCCPRT    *PURGE FCC IF ANY
08E7 17 0718            LBSR   CLRPLN    *OUTPUT "END START"
08EA 8E 1273            LDX    #ENDS
08ED 108E 0069          LDY    #POPCD
08F1 C6 03              LDB    #3
08F3 A6 80      P4END2  LDA    0,X+
08F5 A7 A0              STA    0,Y+
08F7 5A                 DECB
08F8 26 F9              BNE    P4END2
08FA DC 0F              LDD    IFRADD
08FC 1083 FFFF          CMPD   #$FFFF
0900 27 17              BEQ    P4END3    *NO TRANSFER ADDRESS
0902 DD 47              STD    IRTAG+1
0904 17 0554            LBSR   HEXASC
0907 DD 70              STD    POPND+1
0909 96 10              LDA    IFRADD+1
090B 17 0540            LBSR   HEX2SC
090E DD 72              STD    POPND+3
0910 86 54              LDA    #'T
0912 97 6F              STA    POPND
0914 97 46              STA    IRTAG
0916 17 0785            LBSR   IRSORT
0919 17 06F6    P4END3  LBSR   PRINT     *PRINT IT
091C 26 C2              BNE    WRTERR
091E 0D 1D              TST    IRFLAG
0920 26 03              BNE    P4END4    *NO IREF DESIRED
0922 17 0801            LBSR   IRPRNT    *PRINT IREF
0925 39         P4END4  RTS

                *REPASS RESETS THE INPUT FILE TO THE
                *NEXT SECTION.  THIS IS USED ONLY
                *ON MULTIPLE FILES, SINCE OVERLAYING
                *CODE COULD SCREW UP THE TAGS.
                *
0926 ED 88 11   REPASS  STD    SIRSEC,X  *FAKE REWIND SECTOR
0929 96 41              LDA    MRIDI     *CHANGE RRIDI
092B 97 18              STA    RRIDI     * TO NEXT RIDI
092D 17 05E6            LBSR   REWIND
0930 25 65              BCS    REPAS1    *ERROR
0932 8E 0259            LDX    #FCBOUT
0935 6C 0E              INC    DRV+11,X  *EXT = xxA to xxZ
0937 8E 0259            LDX    #FCBOUT
093A 86 02      REPAS4  LDA    #2
093C A7 84              STA    FUNC,X
093E BD D406            JSR    FMS
0941 27 1D              BEQ    REPAS6    *OK
0943 A6 01              LDA    STAT,X
0945 81 03              CMPA   #3        *FILE ALREADY EXISTS
0947 26 4E              BNE    REPAS1    *ERROR
0949 A6 04              LDA    DRV+1,X
094B A7 9F 0007         STA    [TADST]
094F 86 0C              LDA    #12
0951 A7 84              STA    FUNC,X
0953 BD D406            JSR    FMS       *DELETE
0956 26 3F              BNE    REPAS1
0958 A6 9F 0007         LDA    [TADST]
095C A7 04              STA    DRV+1,X   *RESTORE NAME
095E 20 DA              BRA    REPAS4
0960 6F 88 38   REPAS6  CLR    COMP,X
0963 8E 0250            LDX    #FCBOUT+DRV+1
0966 108E 00FC          LDY    #MOFILE
096A C6 08              LDB    #8
096C 17 1204            LBSR   MOVENM    *MOVE NAME TO HEADING
096F B6 CC03            LDA    DEPTH
0972 27 03              BEQ    REPAS2
0974 17 0704            LBSR   PAGE
0977 9E 07      REPAS2  LDX    TADST     *RESTORE FLAGS, ETC.
0979 30 03              LEAX   3,X
```

```
097B 9F 09              STX    TAGEND
097D 30 03              LEAX   3,X
097F 9F 06              STX    TTGST
0981 9F 00              STX    TTGEND
0983 CC FFFF            LDD    #$FFFF
0986 DD 0F              STD    IFRADD
0988 CC 008A            LDD    #BOPND+1
098B DD 16              STD    FCCLOC
098D 0F 11              CLR    CNTR
098F 0F 13              CLR    LINEND
0991 0F 14              CLR    LINEND+1
0993 0C 14              INC    LINEND+1
0995 0F 15              CLR    FCCCNT    *ALSO SETS EQUAL FLAG
0997 39         REPAS1  RTS
                *

                * LIB 1,DECODE.TXT
                        PAG
                *DECODES THE BINARY INTO SOURCE.
                *THE HARDEST TO CODE AND UNDERSTAND
                *OF ANY ROUTINE IN THE PROGRAM.
                *
0998 0F 23      DCDCMD  CLR    BADOP
099A 0F 22              CLR    MONCTG
099C 0F 2E              CLR    CMDCNT    *0 BYTES IN COMMAND
099E CC FFFF            LDD    #$FFFF
09A1 DD 33              STD    TAGBLD+1
09A3 17 03D7            LBSR   READ      *GET OPCODE
09A6 27 03              BEQ    DCD1
09A8 0C 23              INC    BADOP
09AA 39                 RTS
09AB 108E 0029  DCD1    LDY    #SAVCMD   *SAVE AS YOU GO
09AF A7 A0              STA    0,Y+
09B1 0C 2E              INC    CMDCNT
09B3 DC 19              LDD    NXTOP
09B5 C3 0001            ADDD   #1
09B8 DD 19              STD    NXTOP     *BUMP NEXT ADDRESS
09BA E6 3F              LDB    -1,Y
09BC 8E 1283            LDX    #CMDTBL   *POINT TO OPCODE TABLE
09BF B6 07              LDA    #7
09C1 3D                 MUL
09C2 30 8B              LEAX   D,X       *POINT TO MNEMONIC
09C4 A6 84              LDA    0,X
09C6 2A 03              BPL    DCD2      *GOOD MNEMONIC
09C8 0C 23              INC    BADOP
09CA 39                 RTS
09CB 81 10      DCD2    CMPA   #$10      *SPECIAL CMD PREFIX
09CD 26 04              BNE    DCD2A
09CF 8D 7C              BSR    DD10
09D1 20 07              BRA    DCD2B
09D3 81 11      DCD2A   CMPA   #$11      *ANOTHER SPECIAL
09D5 26 03              BNE    DCD2B
09D7 17 00A2            LBSR   DD11
09DA 0D 23      DCD2B   TST    BADOP
09DC 27 01              BEQ    DCD3
09DE 39                 RTS
                        PAG
09DF 34 36      DCD3    PSHS   X,Y,D
09E1 108E 0069          LDY    #POPCD
09E5 C6 05              LDB    #5
09E7 A6 80      DCD3D   LDA    0,X+      *MOVE MNEMONIC
09E9 A7 A0              STA    0,Y+
09EB 5A                 DECB
09EC 26 F9              BNE    DCD3D
09EE 35 36              PULS   X,Y,D
09F0 A6 06              LDA    6,X       *HEX PRINT FORMAT
09F2 97 31              STA    HEXFMT
09F4 A6 05              LDA    5,X       *INFO BYTE
09F6 1F 89              TFR    A,B
09F8 84 07              ANDA   #7
09FA 97 2F              STA    BYTCNT    *# BYTES IN COMMAND
09FC 1F 98              TFR    B,A
09FE 84 08              ANDA   #8
0A00 97 24              STA    IFER      *TRANSFER OF COMMAND
0A02 86 44              LDA    #'D
0A04 97 32              STA    TAGBLD
0A06 C4 F0              ANDB   #$F0
0A08 2A 06              BPL    DCD3C
0A0A B6 54              LDA    #'T
0A0C 97 32              STA    TAGBLD
0A0E C4 70              ANDB   #$70
0A10 D7 30      DCD3C   STB    MODE
0A12 DA 2F      DCD3A   DEC    BYTCNT    *GET REST OF BYTES
0A14 27 12              BEQ    DCD3B     *THAT'S ALL
0A16 17 0364            LBSR   READ
0A19 26 31              BNE    DCD3RT
0A1B A7 A0              STA    0,Y+
0A1D 0C 2E              INC    CMDCNT
0A1F DC 19              LDD    NXTOP
0A21 C3 0001            ADDD   #1
0A24 DD 19              STD    NXTOP
0A26 20 EA              BRA    DCD3A
0A28 D6 30      DCD3B   LDB    MODE
0A2A 1027 0207          LBEQ   INH       *INHERENT: TEST FOR PC
0A2E C1 10              CMPB   #$10
0A30 27 79              BEQ    IDX       *TAG IF PCR
0A32 C1 20              CMPB   #$20
0A34 1027 01BB          LBEQ   DIR       *TAG DIRECT
0A38 C1 30              CMPB   #$30
0A3A 1027 0102          LBEQ   EXT       *TAG EXTENDED
0A3E C1 40              CMPB   #$40
0A40 1027 0205          LBEQ   IMM       *IMMEDIATE: NEED OPERAND
0A44 C1 50              CMPB   #$50
0A46 1027 01CE          LBEQ   REL       *TAG RELATIVE
0A4A 0C 23              INC    BADOP     *TABLE MISTAKE!
0A4C 39         DCD3RT  RTS
```

```
                        PAG
0A40 17 0320   DO10     LBSR    READ        *GET IO CODE
0A50 26 27              BNE     DO10A
0A52 A7 A0              STA     0,Y+
0A54 0C 2E              INC     CNDCNT
0A56 DC 19              LDD     MITOP
0A58 C3 0001            ADDD    #1
0A5B DD 19              STD     MITOP
0A5D E6 3F              LDB     -1,Y
0A5F BE 12BA            LDX     #LOOK10     *FIND 2ND BYTE
0A62 BC 12AA   DO10C    CMPX    #LKIOE
0A65 27 12              BEQ     DO10A       *NOT THERE
0A67 E1 80              CMPB    0,X+
0A69 26 F7              BNE     DO10C
0A6B 1F 10              TFR     X,D
0A6D 83 1285            SUBD    #LOOK10+1
0A70 86 07              LDA     #7
0A72 3D                 MUL
0A73 8E 1983            LDX     #TENCMD
0A76 30 8B              LEAX    D,X
0A78 39                 RTS
0A79 0C 23     DO10A    INC     BADOP
0A7B 39                 RTS
                        *
0A7C 17 02FE   DO11     LBSR    READ
0A7F 26 27              BNE     DO11A
0A81 A7 A0              STA     0,Y+
0A83 0C 2E              INC     CNDCNT
0A85 DC 19              LDD     MITOP
0A87 C3 0001            ADDD    #1
0A8A DD 19              STD     MITOP
0A8C E6 3F              LDB     -1,Y
0A8E BE 12AA            LDX     #LOOK11     *SAME LOGIC AS IO
0A91 BC 12B3   DO11C    CMPX    #LK11E
0A94 27 12              BEQ     DO11A
0A96 E1 80              CMPB    0,X+
0A98 26 F7              BNE     DO11C
0A9A 1F 10              TFR     X,D
0A9C 83 12AB            SUBD    #LOOK11+1
0A9F 86 07              LDA     #7
0AA1 3D                 MUL
0AA2 8E 1A80            LDX     #EQVCMD

0AA5 30 8B              LEAX    D,X
0AA7 39                 RTS
0AA8 0C 23     DO11A    INC     BADOP
0AAA 39                 RTS
                        PAG
0AAB A6 3F     IDX      LDA     -1,Y        *POST BYTE
0AAD 2B 1D              BMI     IDX2
0AAF 17 D124            LBSR    IDXREG      * STASH REG IN PLINE
0AB2 84 1F              ANDA    #$1F        *VALID, BUT TSC DOESN'T
0AB4 1027 0092          LBEQ    IDX9        * USE FOR 0 OFFSET*
0AB8 85 1D              BITA    #$10
0ABA 27 05              BEQ     IDX1        *POSITIVE OFFSET
0ABC 40                 NEGA
0ABD C6 2D              LDB     #'-
0ABF D7 71              STB     POPND+2
0AC1 84 0F     IDX1     ANDA    #$F
0AC3 8B 00              ADDA    #0
0AC5 19                 DAA
0AC6 17 0292            LBSR    HEXASC
0AC9 DD 73              STD     POPND+4
0ACB 39                 RTS
0ACC 85 10     IDX2     BITA    #$10
0ACE 27 0E              BEQ     IDX3
0AD0 C6 5B              LDB     #'[
0AD2 D7 6F              STB     POPND
0AD4 C6 5D              LDB     #']
0AD6 07 7D              STB     POPND+14
0AD8 81 9F              CMPA    #%10011111
0ADA 1027 00CC          LBEQ    IDX16       *INDIRECT ADDR
0ADE 84 0D     IDX3     ANDA    #$0D
0AE0 81 0D              CMPA    #$0D
0AE2 27 04              BEQ     IDX4        *PCR
0AE4 81 0C              CMPA    #$0C
0AE6 26 10              BNE     IDX5
0AE8 86 2C     IDX4     LDA     #',         *PCR
0AEA C6 50              LDB     #'P
0AEC DD 77              STD     POPND+8
0AEE 86 43              LDA     #'C
0AF0 C6 52              LDB     #'R
0AF2 DD 79              STD     POPND+10
0AF4 A6 3F              LDA     -1,Y
0AF6 20 05              BRA     IDX6
0AF8 A6 3F     IDX5     LDA     -1,Y        *GET POSTBYTE AGAIN
0AFA 17 0009            LBSR    IDXREG      *STASH REG IN PLINE
0AFD 84 9F     IDX6     ANDA    #%10011111
0AFF 81 9F              CMPA    #%10011111
0B01 1027 00A5          LBEQ    IDX16       *INDIRECT
0B05 84 9F              ANDA    #%10011111  *MASK REGISTER
0B07 81 87              CMPA    #$87        *ILLEGAL POST-IDX BYTE
0B09 27 3F              BEQ     IDX9
0B0B 81 8A              CMPA    #$8A
0B0D 27 3B              BEQ     IDX9
0B0F 81 8E              CMPA    #$8E
0B11 27 37              BEQ     IDX9
0B13 81 8F              CMPA    #$8F
0B15 27 33              BEQ     IDX9
0B17 81 90              CMPA    #$90
0B19 27 2F              BEQ     IDX9
0B1B 81 92              CMPA    #$92
0B1D 27 2B              BEQ     IDX9
0B1F 81 97              CMPA    #$97
0B21 27 27              BEQ     IDX9
0B23 81 9A              CMPA    #$9A
0B25 27 23              BEQ     IDX9
0B27 81 9E              CMPA    #$9E
0B29 27 1F              BEQ     IDX9
0B2B 84 0F              ANDA    #$F
0B2D 85 08              BITA    #8
0B2F 27 1C              BEQ     IDX10
0B31 81 08              CMPA    #8
0B33 27 42              BEQ     IDX19
0B35 81 09              CMPA    #9
0B37 27 48              BEQ     IDX20
0B39 81 0B              CMPA    #$B
0B3B 26 05              BNE     IDX8
0B3D C6 44              LDB     #'D
0B3F D7 73              STB     POPND+4
0B41 39                 RTS
0B42 81 0C     IDX8     CMPA    #$C
0B44 27 71              BEQ     IDX17       *1 BYTE,PCR
0B46 81 0D              CMPA    #$D
0B48 27 70              BEQ     IDX18       *2 BYTE,PCR
0B4A 0C 23     IDX9     INC     BADOP
0B4C 39        IDXR     RTS
0B4D C6 30     IDX10    LDB     #'0
0B4F D7 73              STB     POPND+4
0B51 C6 2B              LDB     #'+
0B53 4D                 TSTA
0B54 27 19              BEQ     IDX12       * ,R+
0B56 4A                 DECA
0B57 27 14              BEQ     IDX11       * ,R++
0B59 C6 2D              LDB     #'-
0B5B 4A                 DECA
0B5C 27 16              BEQ     IDX14       * ,-R
0B5E 4A                 DECA
0B5F 27 11              BEQ     IDX13       * ,--R
0B61 4A                 DECA
0B62 27 E8              BEQ     IDXR
0B64 C6 42              LDB     #'B
0B66 4A                 DECA
0B67 27 01              BEQ     IDX15
0B69 5A                 DECB                * = 'A
0B6A D7 73     IDX15    STB     POPND+4
0B6C 39                 RTS
0B6D D7 7C     IDX11    STB     POPND+13
0B6F D7 78     IDX12    STB     POPND+12
0B71 39                 RTS
0B72 D7 79     IDX13    STB     POPND+10
0B74 D7 78     IDX14    STB     POPND+9
0B76 39                 RTS
0B77 17 D2BA   IDX19    LBSR    GET1
0B7A 26 DD              BNE     IDXR
0B7C E6 3F              LDB     -1,Y
0B7E 1D                 SEX
0B7F 20 07              BRA     IDX20A
0B81 17 02A0   IDX20    LBSR    GET2
0B84 26 C6              BNE     IDXR
0B86 EC 3E              LDD     -2,Y
0B88 2A 00     IDX20A   BPL     IDX20B
0B8A 34 04              PSHS    B
0B8C C6 2D              LDB     #'-
0B8E D7 71              STB     POPND+2
0B90 35 04              PULS    B
0B92 43                 COMA
0B93 53                 COMB
0B94 C3 0001            ADDD    #1
0B97 34 04     IDX20B   PSHS    B
0B99 C6 24              LDB     #'$
0B9B D7 72              STB     POPND+3
0B9D 17 02B8            LBSR    HEXASC
0BA0 DD 73              STD     POPND+4
0BA2 35 02              PULS    A
0BA4 17 02B4            LBSR    HEXASC
0BA7 DD 75              STD     POPND+6
0BA9 39                 RTS
                        *
0BAA 17 0277   IDX16    LBSR    GET2        *READ 2 BYTES
0BAD 26 98              BNE     IDXR
0BAF EC 3E              LDD     -2,Y        *GET INFO
0BB1 DD 33              STD     TAGBLD+1    *STASH IN TAG
0BB3 17 02BC            LBSR    TRIPPL      *RIPPLE INTO TEMP
0BB6 39                 RTS
                        *
0BB7 17 0274   IDX17    LBSR    GET1        *READ 1 BYTE
0BBA 26 90              BNE     IDXR
0BBC E6 3F              LDB     -1,Y
0BBE 1D                 SEX
0BBF D3 19              ADDD    MITOP       *ADD OFFSET
0BC1 DD 33              STD     TAGBLD+1    *STASH
0BC3 17 02AC            LBSR    TRIPPL
0BC6 39                 RTS
                        *
0BC7 17 025A   IDX18    LBSR    GET2        *READ 2 BYTES
0BCA 26 80              BNE     IDXR
0BCC EC 3E              LDD     -2,Y
0BCE D3 19              ADDD    MITOP
0BD0 DD 33              STD     TAGBLD+1    *STASH
0BD2 17 0290            LBSR    TRIPPL
0BD5 39                 RTS
                        *
0BD6 34 06     IDXREG   PSHS    D
0BD8 C6 58              LDB     #'X
0BDA 84 60              ANDA    #$60
0BDC 27 00              BEQ     IDXRE1
0BDE 5C                 INCB
0BDF 81 20              CMPA    #$20
0BE1 27 08              BEQ     IDXRE1      * Y
0BE3 C6 55              LDB     #'U
0BE5 81 40              CMPA    #$40
0BE7 27 02              BEQ     IDXRE1      * U
```

```
08E9 C6 53          LDB   #'S
08EB D7 7A   IDIRE1 STB   POPND+11   *STASH REGISTER
08ED 86 2C          LDA   #',
08EF 97 77          STA   POPND+8
08F1 35 86          PULS  D,PC
                    PAG
08F3 E6 3F   DIR    LDB   -1,Y        *DIRECT ADDR BYTE
08F5 4F             CLRA
08F6 DD 33          STD   TAGBLD+1
08F8 86 01          LDA   #1
08FA 97 12          STA   OPFLAG      *FLAG OP USED
08FC 17 0273        LBSR  TRIPPL
08FF 8E 1225        LDX   #OPMSG1
0C02 108E 008E      LDY   #POOR
0C06 C6 08          LDB   #11
0C08 A6 80   DIR1   LDA   0,X+        *PRINT DIRECT WARNING
0C0A A7 A0          STA   0,Y+
0C0C 5A             DECB
0C0D 26 F9          BNE   DIR1
0C0F 39             RTS
                    *
0C10 EC 3E   EXT    LDD   -2,Y
0C12 DD 33          STD   TAGBLD+1
0C14 17 025B        LBSR  TRIPPL
0C17 39             RTS
                    *
0C18 E6 3F   REL    LDB   -1,Y
0C1A 96 29          LDA   SAVCMD
0C1C 81 10          CMPA  #$10
0C1E 27 08          BEQ   REL1
0C20 81 16          CMPA  #$16        *LONG BRANCHES
0C22 27 07          BEQ   REL1
0C24 81 17          CMPA  #$17
0C26 27 03          BEQ   REL1
0C28 1B             SEX
0C29 20 02          BRA   REL2
0C2B A6 3E   REL1   LDA   -2,Y
0C2D D3 19   REL2   ADDD  NXTOP
0C2F DD 33          STD   TAGBLD+1
0C31 17 023E        LBSR  TRIPPL
0C34 39             RTS
                    PAG
0C35 DC 29   IMM    LDD   SAVCMD
0C37 81 34          CMPA  #$34        *PSHS
>0C39 1027 006C     LBEQ  IMM16
0C3D 81 35          CMPA  #$35        *PULS
0C3F 27 5C          BEQ   IMM1
0C41 81 36          CMPA  #$36        *PSHU
>0C43 1027 0062     LBEQ  IMM16
0C47 81 37          CMPA  #$37        *PULU
0C49 27 52          BEQ   IMM1
0C4B 81 1F          CMPA  #$1F        *TFR
0C4D 27 04          BEQ   IMM
0C4F 81 1E          CMPA  #$1E        *EXG
0C51 26 47          BNE   IMM2
0C53 34 04   IMM    PSHS  B           *PUT OPND IN PRINT
0C55 17 0096        LBSR  IMM7
0C58 DD 75          STD   POPND+6
0C5A E6 E4          LDB   0,S
0C5C C4 0F          ANDB  #$F
0C5E C1 08          CMPB  #$8
0C60 26 03          BNE   IMM4B
0C62 17 01E2        LBSR  OPWARN      *WARN ABOUT OP
0C65 E6 E4   IMM4B  LDB   0,S
0C67 58             ASLB
0C68 58             ASLB
0C69 58             ASLB
0C6A 58             ASLB
0C6B 17 0080        LBSR  IMM7
0C6E DD 78          STD   POPND+9
0C70 C6 2C          LDB   #',
0C72 D7 77          STB   POPND+8
0C74 35 04          PULS  B
0C76 1F 98          TFR   B,A         *CHECK FOR INVALID
0C78 48                               *TRANSFER BYTE
0C79 48             ASLA
0C7A 48             ASLA
0C7B 48             ASLA
0C7C 2A 0E          BPL   IMM6        *BOTH MUST BE POS
0C7E C4 F0          ANDB  #$F0        *OR
0C80 2A 14          BPL   IMM5        *BOTH MUST BE NEG
0C82 34 04          PSHS  B           *BOTH NEG
0C84 AA E0          ORA   0,S+
0C86 85 40          BITA  #$40
0C88 26 0C          BNE   IMM5        *INVALID
0C8A 20 0E          BRA   IMM2        *OF NO CONCERN
0C8C C1 60   IMM6   CMPB  #$60        *BOTH POSITIVE
0C8E 24 06          BCC   IMM5        *INVALID
0C90 81 50          CMPA  #$50
0C92 27 04          BEQ   IMM3        *PC
0C94 20 04          BRA   IMM2        *OF NO CONCERN
0C96 DC 23   IMM5   INC   BADOP
0C98 D7 24   IMM3   STB   IFER
0C9A 1A 01   IMM2   SEC
0C9C 39             RTS
0C9D 5D      IMM1   TSTB
0C9E 2A 02          BPL   IMM14
0CA0 D7 24          STB   IFER        *PC LOADED
0CA2 C5 08   IMM14  BITB  #8
0CA4 27 03          BEQ   IMM16
0CA6 17 019E        LBSR  OPWARN      *OP WARNING
0CA9 34 30   IMM16  PSHS  X,Y
0CAB BE 006F        LDX   #POPND
0CAE 108E 120C      LDY   #PSPL
0CB2 54      IMM17  LSRB
0CB3 24 0C          BCC   IMM18
0CB5 34 06          PSHS  D
0CB7 EC A4          LDD   0,Y
0CB9 ED 81          STD   0,X++
0CBB C6 2C          LDB   #',
0CBD E7 80          STB   0,X+
0CBF 35 06          PULS  D
0CC1 27 04   IMM18  BEQ   IMM19
0CC3 31 22          LEAY  2,Y
0CC5 20 EB          BRA   IMM17
0CC7 C6 A0   IMM19  LDB   #$A0
0CC9 E7 1F          STB   -1,X        *ZAP EXTRA COMMA
0CCB 96 29          LDA   SAVCMD
0CCD E6 82   IMM20  LDB   0,-X        *SEARCH SU
0CCF C1 55          CMPB  #'U
0CD1 27 09          BEQ   IMM21
0CD3 BC 006F        CMPX  #POPND
0CD6 26 F5          BNE   IMM20
0CD8 1A 01          SEC
0CDA 35 80          PULS  X,Y,PC
0CDC C6 A0   IMM21  LDB   #$A0
0CDE B5 02          BITA  #2
0CE0 27 06          BEQ   IMM22       *SYSTEM STACK
0CE2 E7 84          STB   0,X
0CE4 1A 01          SEC
0CE6 35 80          PULS  X,Y,PC
0CE8 E7 1F   IMM22  STB   -1,X        *USER STACK
0CEA 1A 01          SEC
0CEC 35 80          PULS  X,Y,PC
                    *
0CEE 57      IMM7   ASRB
0CEF 57             ASRB
0CF0 57             ASRB
0CF1 57             ASRB
0CF2 2B 12          BMI   IMM9        *A,B,CC,DP
0CF4 34 20          PSHS  Y           *O,X,Y,U,S,PC
0CF6 108E 1FF8      LDY   #TFRG
0CFA 58             TSTB
0CFB 26 04   IMM10  BNE   IMM11
0CFD EC A4          LDD   0,Y
0CFF 35 A0          PULS  Y,PC
0001 31 22   IMM11  LEAY  2,Y
0003 5A             DECB
0004 20 F5          BRA   IMM10
0006 34 20   IMM9   PSHS  Y           *A,B,CC,DP
0008 108E 1204      LDY   #TFRG+12
000C C4 07          ANDB  #7
000E 26 04   IMM12  BNE   IMM13
0010 EC A4          LDD   0,Y
0012 35 A0          PULS  Y,PC
0014 31 22   IMM13  LEAY  2,Y
0016 5A             DECB
0017 20 F5          BRA   IMM12
                    *
                    *
0019 C6 23   IMM    LDB   #'#         *PUT IMM VALUE IN PLINE
001B D7 70          STB   POPND+1
001D C6 24          LDB   #'$
001F D7 72          STB   POPND+3
0021 96 2E          LDA   CNDCNT
0023 81 02          CMPA  #2
0025 27 0F          BEQ   IMM2
0027 EC 3E          LDD   -2,Y
0029 DD 33          STD   TAGBLD+1
002B 17 0120        LBSR  HEXASC
002E DD 73          STD   POPND+4
0030 8D 04          BSR   IMM2
0032 17 0130        LBSR  TRIPPL
0035 39             RTS
0036 A6 3F   IMM2   LDA   -1,Y        *1 BYTE IMM
0038 17 0120        LBSR  HEXASC
003B DD 75          STD   POPND+6
003D 39             RTS
* END OF DECODE LOGIC
* LIB 1,SUPPORT.TXT
                    PAG
                    *FIND LOCATES THE BYTE OF CODE
                    *WHICH IS AT A SPECIFIC ADDRESS
                    *(NXTOP).
                    *
003E DC 08   FIND   LDD   TTOST       *ABORT TEMP TAGS
0040 DD 00          STD   TTGEND
0042 8E 0119        LDX   #FCBIN
0045 6F 01          CLR   STAT,X
0047 DC 37          LDD   GOTRK
0049 10A3 88 1E     CMPD  NEXT,X
004D 27 0C          BEQ   FIND1
004F ED 88 1E       STD   NETT,X
0052 86 09          LDA   #9
0054 A7 84          STA   FUNC,X
0056 BD 0406        JSR   FMS
0059 26 1F          BNE   FIND3
005B 96 39   FIND1  LDA   GDRIDX
005D A7 88 23       STA   RIBX,X      *PSEUDO REWIND
0060 96 3A          LDA   GDRGTR
0062 97 11          STA   CNTR
0064 DC 35          LDD   GDRGCP
0066 83 0001        SUBD  #1
0069 C3 0001 FIND2  ADDD  #1
006C DD 19          STD   NXTOP
006E 8D 00          BSR   READ        *LOOP TILL FOUND
0070 26 08          BNE   FIND3
0072 DC 19          LDD   NXTOP       *ADDR WE'RE AT
0074 1093 25        CMPD  LASTB       *ADDR WE WANT
0077 25 F0          BCS   FIND2       *NOT YET
0079 39             RTS
007A 1A 01   FIND3  SEC
007C 39             RTS
```

```
                        PAG
                    *READ SIMPLY READS THE NEXT BYTE
                    *IN THE FILE.
                    *IT IS ENTERED WITH THE EXPECTED
                    *ADDRESS IN NXTOP.
                    *IT EXITS WITH THE NEXT BYTE, AND
                    *THE REAL ADDRESS IN NXTOP.
                    *IF THEY DIDN'T MATCH, NONETG IS
                    *SET.
                    *
007D  6  30   READ    PSHS  X,Y
007F FE  CC28         LDU   #ERMSG
0082 8E  0119         LDX   #FCBIN      *POINT TO FCB
0085 00  11           TST   CNTR        *READY FOR NEXT BYTE?
0087 26  53           BNE   READ3       *YES
0089 EC  88 1E        LDD   NEXT,X      *SAVE CUR LOCATION
008C DD  3F           STD   NSEC
008E A6  88 23        LDA   RIDX,X
0091 97  41           STA   NRIDX
0093 BD  63     READ6 BSR   READR       *02 OR 16
0095 26  4D           BNE   READ4
0097 4D               TSTA
0098 36  02           PSHU  A
009A 27  19           BEQ   READ9       *REST OF BLOCK NULL
009C BD  5A           BSR   READR
009E 26  44           BNE   READ4
00A0 36  02           PSHU  A
00A2 BD  54           BSR   READR
00A4 26  3E           BNE   READ4
00A6 1F  89           TFR   A,B
00A8 37  02           PULU  A
00AA 1F  02           TFR   D,Y
00AC 37  02           PULU  A
00AE 81  16           CMPA  #$16        *START ADDR
00B0 26  0E           BNE   READ1
00B2 109F 6F          STY   CFRADD
00B5 EC  88 40  READ9 LDD   DATA,X      *NEXT SECTOR =
00B8 DD  3F           STD   NSEC        * NEXT LOCATION
00BA 86  04           LDA   #4
00BC 97  41           STA   NRIDX
00BE 20  24           BRA   READ4
00C0 81  02     READ1 CMPA  #2          *STX STARTS NEXT BLOCK
00C2 26  2C           BNE   READ7       *ERROR
00C4 109F 27          STY   LSTBLK
00C7 109C 19          CMPY  NXTOP       *NEXT ADDRESS?
00CA 27  04           BEQ   READ2       *YES
00CC 25  16           BCS   READ4       *ADDR DON'T GO BACKWARDS
00CE 0C  22           INC   NONETG      *FLAG NON-CONTIGUOUS
00D0 109F 19    READ2 STY   NXTOP       *ADDR WHERE WE ARE
00D3 BD  2            BSR   READR
00D5 26  00           BNE   READ4
00D7 4D               TSTA
00D8 27  0A           BEQ   READ4       *NO NULL LENGTH
00DA 97  11           STA   CNTR        *NEW BLOCK COUNTER
00DC 0A  11     READ3 DEC   CNTR
00DE BD  18           BSR   READR
00E0 26  02           BNE   READ4
00E2 35  B0           PULS  X,Y,PC      *RESTORE & RETURN
                    *
00E4 6D  01     READ4 TST   STAT,X      *CHECK STATUS
00E6 26  06           BNE   READ5
00E8 86  08           LDA   #8          *FORCE EOF
00EA A7  01           STA   STAT,X
00EC 1C  FB     READ8 ANDCC #$FB        *TURN OFF EQUAL
00EE 35  B0     READ5 PULS  X,Y,PC      *RESTORE & RETURN
00F0 8E  11E4   READ7 LDX   #MSG4       *FORMAT ERROR
00F3 BD  CD1E         JSR   PSTRNG
00F6 20  F4           BRA   READ8
                        PAG
                    * READ RANDOM FETCHES WHATEVER
                    *BYTE IS AT OFFSET RIDX IN THE
                    *BUFFER.
                    *IF RIDX = 00, THE NEXT SECTOR
                    *IS READ.
                    *
00F8 E6  88 23  READR LDB   RIDX,X      *RANDOM READ
00F8 26  16           BNE   READR1
00FD EC  88 40        LDD   DATA,X
0100 27  1D           BEQ   READR2      *EOF
0102 ED  88 1E        STD   NEXT,X
0105 86  09           LDA   #9          *READ NEXT SECTOR
0107 A7  84           STA   FUNC,X
0109 BD  D406         JSR   FMS
010C 26  0E           BNE   READR3      *ERROR
010E C6  04           LDB   #4          *FIRST DATA OFFSET
0110 E7  88 23        STB   RIDX,X
0113 4F         READR1 CLRA
0114 31  88 40        LEAY  DATA,X
0117 A6  A8           LDA   D,Y
0119 6C  88 23        INC   RIDX,X
011C 6D  01     READR3 TST   STAT,X
011E 39               RTS
011F C6  08     READR2 LDB   #8
0121 E7  01           STB   STAT,X
0123 39               RTS
                        PAG
                    *GET2 AND GET1 GET THE NEXT
                    *1 OR 2 BYTES IN THE FILE.
                    *THEY ARE USED WHEN THE IDX
                    *MODE REQUIRES 1 OR 2 MORE
                    *BYTES TO COMPLETE THE COMMAND.
                    *
0124 BD  08     GET2  BSR   GET1
0126 26  1E           BNE   GETRET
0128 86  01           LDA   #1          *EXTRA BYTES MUST PRINT
012A 9A  31           ORA   IEXFMT
012C 97  31           STA   IEXFMT
012E 17  FF4C   GET1  LBSR  READ
0131 26  13           BNE   GETRET
0133 A7  A0           STA   0,Y+
0135 0C  2E           INC   CMDCNT
0137 86  02           LDA   #2          *ADDITIONAL BYTE TO PRINT
0139 9A  31           ORA   IEXFMT
013B 97  31           STA   IEXFMT
013D DC  19           LDD   NXTOP
013F C3  0001         ADDD  #1
0142 DD  19           STD   NXTOP
0144 85  00           BITA  #0          *EQUAL RETURN
0146 39         GETRET RTS
                    *
                    *
                    *
                    *OPWARN PRINTS THE DIRECT PAGE
                    *LOADED WARNING.
                    *(USED AFTER A TFR OR EXG).
                    *
0147  4  36   OPWARN PSHS  X,Y,D
0149 8E  121C         LDX   #OPMSG      *OP WARNING
014C 10BE 00BE        LDY   #PCOM
0150 C6  09           LDB   #9
0152 A6  80     INM15 LDA   0,X+
0154 A7  A0           STA   0,Y+
0156 5A               DECB
0157 26  F9           BNE   INM15
0159 35  B6           PULS  X,Y,D,PC
                    *
                    *CHANGES HEX IN A REG TO ASCII
                    *IN D REG
                    *
015B 1F  89     HEXASC TFR  A,B         *A UNPACKED TO D
015D BD  04           BSR   HEXAS1      *MSD
015F 1E  89           EXG   A,B
0161 20  04           BRA   HEXAS2
                    *
0163 54         HEXAS1 LSRB
0164 54                LSRB
0165 54                LSRB
0166 54                LSRB
0167 C4  0F     HEXAS2 ANDB  #$F
0169 CA  30            ORB   #$30
016B C1  3A            CMPB  #$3A
016D 25  02            BCS   HEXAS3
016F CB  07            ADDB  #7
0171 39         HEXAS3 RTS
                        PAG
                    *APPENDS THE JUST-BUILT TAG
                    *(TAGBLD) TO THE TEMPORARY
                    *TAG TABLE.
                    *
0172 34  10     TRIPPL PSHS  X
0174 BD  23           BSR   ZAPT
0176 0D  32           TST   TAGBLD
0178 27  1D           BEQ   TRIP4       *NO TAG TO APPEND
017A 9E  00           LDX   TTGEND
017C 30  03           LEAX  3,X
017E BC  CC28         CMPX  #ERMEND
0181 24  0C           BCC   TRIP5
0183 9F  00           STX   TTGEND
0185 DC  33           LDD   TAGBLD+1
0187 ED  83           STD   0,--X
0189 96  32           LDA   TAGBLD
018B A7  82           STA   0,-X
018D 20  08           BRA   TRIP4
018F 8E  11C7   TRIP5 LDX   #MSG2
0192 BD  CD1E         JSR   PSTRNG
0195 0C  1C           INC   ABORTF      *ABORT
0197 35  90     TRIP4 PULS  X,PC
                        PAG
                    * CHECK CODING TAG IN DATA AREAS
                    *
0199  4  10   ZAPT   PSHS  X
019B 0C  19           LDD   NXTOP       *CHECK IF STILL CODE
019D 83  0001         SUBD  #1
01A0 BD  0C           BSR   ZAPT4
01A2 96  32           LDA   TAGBLD
01A4 81  54           CMPA  #'T
01A6 26  04           BNE   ZAPT2       *NOT T DON'T CARE
01A8 0C  33           LDD   TAGBLD+1
01AA BD  02           BSR   ZAPT4
01AC 35  90     ZAPT2 PULS  X,PC
                    *
01AE 9E  03     ZAPT4 LDX   DATST       *TAG IN DATA AREA?
01B0 9C  05     ZAPT3 CMPX  DATEND
01B2 27  0E           BEQ   ZAPT5       *TAG OK
01B4 10A3 84          CMPD  0,X
01B7 25  0A           BCS   ZAPT6       *< START: NEXT AREA
01B9 10A3 02          CMPD  2,X
01BC 22  05           BHI   ZAPT6       *> END: NEXT AREA
01BE 0F  32           CLR   TAGBLD      *ZAP TAG
01C0 0C  23           INC   BADOP       *NOT CODE
01C2 39         ZAPT5 RTS
01C3 30  04     ZAPT6 LEAX  4,X
01C5 20  E9           BRA   ZAPT3
                        PAG
                    *RIPPLES THE TEMP TAG TABLE INTO
                    *THE REAL TAG TABLE.  ANY "DATA
                    *TAGS WHICH MAY ALSO BE CODING
                    *TAGS ARE CHANGED TO "T".
                    *
01C7  4  30   RIPPLE PSHS  X,Y          *RIPPLE ENTIRE TEMP
01C9 109E 08          LDY   TTGST       *TAG TABLE TO REAL TAG
01CC 109C 00    RIP1  CMPY  TTGEND      *TABLE
01CF 27  08           BEQ   RIP8
```

```
0ED1 31  21          LEAY  1,Y      *SKIP O/T
0ED3 EC  A1          LDD   0,Y++    *GET ADDR
0ED5 8D  0F          BSR   RIP3
0ED7 20  F3          BRA   RIP1
0ED9 1A  01    RIP8  SEC
0EDB DC  09          LDD   TAGEND
0EDD C3  0003        ADDD  #3
0EE0 DD  08          STD   TTGST
0EE2 DD  00          STD   TTGEND   *ZAP TEMP TAGS
0EE4 35  80          PULS  X,Y,PC
                *
0EE6 17  0097  RIP3  LBSR  FINDTG   *TAG ALREADY THERE?
0EE9 25  07          BCS   RIP5     *NO
0EEB A6  30          LDA   -3,Y     *FOUND MATCH
0EED AA  84          ORA   0,X      *POSSIBLE D TO T
0EEF A7  84          STA   0,X
0EF1 39              RTS
                *
0EF2 9E  09    RIP5  LDX   TAGEND   *PUT IT IN
0EF4 30  03          LEAX  3,X
0EF6 9F  09          STX   TAGEND
0EF8 30  1D    RIP6  LEAX  -3,X
0EFA 9C  07          CMPX  TAGS1
0EFC 27  0F          BEQ   RIP9     *PUT IT HERE
0EFE EC  1E          LDD   -2,X
0F00 10A3 3E         CMPD  -2,Y
0F04 25  08          BCS   RIP9     *PUT IT HERE
0F06 ED  01          STD   1,X      *EVERYBODY SHIFT DOWN
0F08 A6  1D          LDA   -3,X
0F0A A7  84          STA   0,X
0F0C 20  EB          BRA   RIP6
0F0E EC  3E    RIP9  LDD   -2,Y     *HERE IT IS
0F10 ED  01          STD   1,X
0F12 A6  3D          LDA   -3,Y
0F14 A7  84          STA   0,X
0F16 39              RTS
                PAG
                *SETS UP 1ST ADDR &
                * RESTORES FILE FOR 1ST READ
0F17 8E  0119  REWIND LDX   #FCB1N   *REWIND FILE
0F1A EC  88 11       LDD   STRSEC,X
0F1D 10A3 88 1E      CMPD  NEXT,X
0F21 27  0C          BEQ   REWIN2   *NO READ NEEDED
0F23 ED  88 1E       STD   NEXT,X
0F26 86  09          LDA   #9
0F28 A7  84          STA   FUNC,X
0F2A BD  D406        JSR   FMS
0F2D 26  35          BNE   REWIN1   *ERROR
0F2F 6F  01    REWIN2 CLR  STAT,X
0F31 0F  11          CLR   CNTR
0F33 0F  19          CLR   NXTOP    *AVOID BOMB
0F35 0F  1A          CLR   NXTOP+1
0F37 96  18          LDA   ARIDX
0F39 A7  88 23       STA   RIDX,X
0F3C 17  FE36        LBSR  READ
0F3F 26  23          BNE   REWIN1   *ERROR
0F41 DC  19          LDD   NXTOP
0F43 DD  35          STD   GDACDR
0F45 EC  88 1E       LDD   NEXT,X
0F48 DD  37          STD   GDTRK
0F4A A6  88 23       LDA   RIDX,X
0F4D 4A              DECA
0F4E A7  88 23       STA   RIDX,X
0F51 97  39          STA   GDRIDX
0F53 96  11          LDA   CNTR
0F55 4C              INCA
0F56 97  11          STA   CNTR
0F58 97  3A          STA   GDCNTR   *FILE AT BEGINNING
0F5A DC  07          LDD   TAGST
0F5C 83  0003        SUBD  #3
0F5F DD  38          STD   CURTAG   *PRELOAD TAG
0F61 1C  FE          CLC
0F63 39              RTS
0F64 1A  01    REWIN1 SEC
0F66 39              RTS
                PAG
                *FINDS NEXT CODING (T) TAG.
                *
0F67 9E  38    NEXTT LDX   CURTAG
0F69 86  54          LDA   #'T
0F6B 30  03    NEXTT1 LEAX  3,X
0F6D 9C  09          CMPX  TAGEND
0F6F 27  0D          BEQ   NEXTT2
0F71 A1  84          CMPA  0,X
0F73 26  F6          BNE   NEXTT1
0F75 9F  38          STX   CURTAG
0F77 EC  01          LDD   1,X
0F79 DD  25          STD   LASTQ
0F7B 1C  FE          CLC
0F7D 39              RTS
0F7E 1A  01    NEXTT2 SEC
0F80 39              RTS
```

TO BE CONTINUED

READING NON-FLEX BASED DISKS

CONTINUED FROM LAST MONTH

```
C3E2 35  30          PULS  X,Y      restore regs
C3E4 39              RTS            return

                *
                * Name     - READSS
                * Function - This routine reads in the track/sector of
                *            the disk in the drive specified by an FCB
                *            pointed to by X. Reg D specifies the
                *            track/sector to read in.
                *
                *            A BNE or BEQ should be used after calling
                *            this routine to check for possible read
                *            errors.
                *
                *            All registers except A are preserved.
                *
          C3E5  READSS EQU   *
C3E5 ED  88 1E       STD   FCBCP,X  set trk/sec
C3E8 86  09          LDA   #IRSS    get function code
C3EA A7  84          STA   FCBFC,X  set code in FCB
C3EC 7E  D406        JMP   FMSCAL   read and return

                *
                * Name     - MVC
                * Function - This routine moves the "from" string
                *            to the "to" string as follows:
                *
                *              X -> "from" field
                *              Y -> "to" field
                *              D = # characters to move
                *
                *            Regs A,B,X,Y are preserved
                *
          C3EF  MVC    EQU   *
C3EF 34  36          PSHS  A,B,X,Y  save regs

C3D0 30  88 40       LEAX  FCBSB,X  "from" field
C3D3 CC  0080        LDD   #SSIZE   length to move
C3D6 10BE 0836       LDY   #BUFFER  "to" field
C3DA 8D  13          BSR   MVC      perform move

C3DC EC  A4          LDD   DOSNXT,Y get next trk/sec to read
C3DE 84  3F          ANDA  #%00111111 keep low 6 bits of track
C3E0 C4  1F          ANDB  #%00011111 keep low 5 bits of sector

          C3F1  MOVE   EQU   *
C3F1 34  06          PSHS  D        save len to move
C3F3 E6  80          LDB   0,X+     get a "from" char
C3F5 E7  A0          STB   0,Y+     trans to "to" field
C3F7 35  06          PULS  D        get len
C3F9 83  0001        SUBD  #1       decr by 1
C3FC 1083 0000       CMPD  #0       done yet?
C400 26  EF          BNE   MOVE     branch if not

C402 35  36          PULS  A,B,X,Y  restore regs
C404 39              RTS            and return

                *
                * Name     - CLC
                * Function - This routine compares two strings
                *            as follows:
                *
                *              X -> string1
                *              Y -> string2
                *              A = length of strings to compare
                *
                *            String1 is compared to String2, and the
                *            appropriate condition code is set.
                *
                *            Regs A,B,X,Y are preseved.
                *
          C405  CLC    EQU   *
C405 34  36          PSHS  A,B,X,Y  save regs

          C407  COMPR  EQU   *
C407 E6  80          LDB   0,X+     get a string1 char
```

```
C409 E1  A0          CMPB  0,Y+      compare to string2 char
C40B 26  03          BNE   RPROC     if new then done

C40D 4A              DECA            all through strings?
C40E 26  F7          BNE   COMPR     branch if not

             C410 RPROC  EQU   *
C410 35  36          PULS  A,B,X,Y   restore regs
C412 39              RTS             and return


                     *
                     * Name     - PDATA
                     * Function - This routine prints a string to an
                     *            output device. On entry, X points
                     *            to the string to print. As with the
                     *            PSTRNG routine in FLEX, an EOT
                     *            delimiter must appear after the
                     *            string to print.
                     *            Regs A and B will remain as is, and
                     *            X will be pointing to the EOT
                     *            delimiter char.
                     *
             C413 PDATA  EQU   *
C413 34  06          PSHS  A,B       save regs

             C415 PDATA1 EQU   *
C415 A6  80          LDA   0,X+      get char from string
C417 81  04          CMPA  #EOT      end of text?
C419 27  05          BEQ   PDATA2    branch if yes
C41B BD  CD1B        JSR   PUTCHR    print char
C41E 20  F5          BRA   PDATA1    and continue

             C420 PDATA2 EQU   *
C420 35  06          PULS  A,B       restore regs
C422 39              RTS             and return


                     *
                     * Name     - ROUTE
                     * Function - This routine determines where the user
                     *            wants his output to be routed. The user
                     *            has three options:
                     *
                     *            RO=TERMINAL: In this case, no control
                     *               info need be set, other than ensuring
                     *               the output switch (DSWTCH) is non
                     *               zero so FLEX will use the OUTCH2
                     *               routine to print characters.
                     *
                     *            RO=PRINTER: In this case, the printer
                     *               module routine must be loaded into
                     *               memory if it is not already there,
                     *               and the output switch must be cleared
                     *               so FLEX will use OUTCH to process
                     *               the characters
                     *
                     *            RO=FILE: In this case, the user is
                     *               prompted for a file to which all
                     *               output will be routed. FLEX's PUTCHR
                     *               routine automatically routes outputted
                     *               chars to the file's FCB if told
                     *               to do so.
                     *
                     *       All registers are preserved
                     *
             C423 ROUTE  EQU   *
C423 34  36          PSHS  A,B,X,Y   save regs

C425 8E  0000        LDX   #0        clear file output address
C428 BF  CC24        STX   FOA       in FLEX

C42B 8E  0843        LDX   #OUTFIL   prompt user for route
C42E BD  CD1E        JSR   PSTRNG
C431 BD  CD09        JSR   INCH      get response
C434 84  5F          ANDA  #$5F      lowercase->uppercase

C436 81  50          CMPA  #'P       printer?
C438 27  6F          BEQ   R2        branch if so

C43A 81  54          CMPA  #'T       terminal?
C43C 27  72          BEQ   R3        branch if so

C43E 81  0D          CMPA  #CR       CR? (same as terminal)
C440 27  6E          BEQ   R3        if so branch

C442 81  46          CMPA  #'F       file?
C444 26  71          BNE   R6        leave if not

                     *
                     * output to a file
                     *
C446 8E  0877        LDX   #TOFILE   ask for "to" filename
C449 BD  CD1E        JSR   PSTRNG
C44C BD  CD1B        JSR   INBUF     get response
C44F 8E  09F3        LDX   #FILFCB   "to" FCB
C452 BD  CD20        JSR   GETFIL    validate filename
C455 24  08          BCC   R4        check if file exists

C457 8E  08A1        LDX   #INVSPC   invalid filename entered
C45A BD  CD1E        JSR   PSTRNG
C45D 20  58          BRA   R6        return

             C45F R4     EQU   *
C45F 86  0B          LDA   #OUT      set extension
C461 BD  CD33        JSR   SETEXT
                     *
                     * Open the file
                     *
C464 86  02          LDA   #IOWRIT   open for write
C466 A7  84          STA   FCBFC,X   save in FCB
C468 BD  D406        JSR   FMSCAL    call FMS
C46B 27  08          BEQ   R7        continue if file not there
                     *
                     * File already exists -> say so and return
                     *
C46D 8E  08BD        LDX   #FXIST
C470 BD  CD1E        JSR   PSTRNG
C473 20  42          BRA   R6        return

             C475 R7     EQU   *
C475 8E  08DB        LDX   #FTYPE    ask for file type
C478 BD  CD1E        JSR   PSTRNG
C47B BD  CD09        JSR   INCH      get response
C47E 84  5F          ANDA  #$5F      lowercase->uppercase

C480 5F              CLRB
C481 81  54          CMPA  #'T       text?
C483 27  11          BEQ   R8        branch if so

C485 81  0D          CMPA  #CR       CR? (same as text)
C487 27  0D          BEQ   R8        branch if so

C489 81  42          CMPA  #'B       binary?
C48B 26  2A          BNE   R6        bad leave if not

C48D C6  A0          LDB   #OPN+BINRY set open-bin bits
C48F 86  FF          LDA   #$CFNSC   type=binary
C491 B7  0A2E        STA   FILFCB+FCBSCF set compress. flag
C494 20  02          BRA   R1

             C496 R8     EQU   *
C496 C6  C0          LDB   #OPN+TEX  set open-text bits
             C498 R1     EQU   *
C498 8E  09F3        LDX   #FILFCB   set all o/p to file

C49B BF  CC24        STX   FOA
C49E 7F  CC22        CLR   DSWTCH    o/p to the file
C4A1 F7  0834        STB   OPNFLG    save flag bits
C4A4 7F  CC04        CLR   WIDTH     zero TTY width value
C4A7 20  0A          BRA   R9        return

             C4A9 R2     EQU   *
                     *
                     * perform setup for output to printer
                     *
C4A9 7F  CC22        CLR   DSWTCH    o/p to aux. device
C4AC BD  0E          JSR   PRTSET    fetch printer module
C4AE 20  03          BRA   R9        and return

             C4B0 R3     EQU   *
                     *
                     * perform setup for output to terminal
```

```
                    *
C4B0 7C  CC22       INC   OSWTCH
                    *
         C4B3 R9    EQU   *
C4B3 1C  FE         CLC             set good RC
C4B5 20  02         BRA   R10       and leave
                    *
         C4B7 R6    EQU   *
C4B7 1A  01         SEC             set bad RC
         C4B9 R10   EQU   *
C4B9 35  36         PUL.S A,B,I,Y   restore regs
C4BB 39             RTS             and return


                    *
                    * Name     - PRTSET
                    * Function - This routine is called to load the
                    *           printer module if necessary,
                    *           and init the PIA, so that output can
                    *           be routed to the printer.
                    *
                    *      No registers are preserved
                    *
         C4BC PRTSET EQU   *
C4BC 7F  CC09       CLR   PAU       disable pause feature
C4BF B6  CCE4       LDA   POUT      get 1st byte of space
C4C2 81  39         CMPA  #$39      is it "RTS"?
C4C4 26  29         BNE   P1S       if not the loaded

                    *
                    * Load printer routine
                    *
C4C6 8E  0914       LDX   #PSYS     move in print rtn name
C4C9 10BE CB44      LDY   >SYSFCB+FCBNAM into system FCB
C4CD CC  0008       LDD   #11
C4D0 17  FF1C       LBSR  MVC
C4D3 7F  0843       CLR   SYSFCB+FCBDN check drive 0

C4D6 8E  CB40       LDX   #SYSFCB   point to system FCB
C4D9 86  01         LDA   #1CREAD   open for read
C4DB A7  84         STA   FCBFC,X
C4DD BD  D406       JSR   FMSCAL    call FMS
C4E0 27  05         BEQ   P1        branch if open ok

C4E2 BD  CD3F       JSR   RPTERR    report error
C4E5 20  11         BRA   P2        and return

         C4E7 P1    EQU   *

C4E7 86  FF         LDA   #SCFMSC   set for binary read
C4E9 A7  88 38      STA   FCBSCF,X
C4EC BD  CD00       JSR   LOAD      load module

         C4EF P1S   EQU   *
C4EF BD  CCC0       JSR   PRINIT    go init port
C4F2 8E  CCE4       LDX   #POUT     set o/p address
C4F5 BF  CD10       STX   OUTCH+1   stuff in FLEX
         C4F8 P2    EQU   *
C4F8 39             RTS             return


                    *
                    * Name     - RESET
                    * Function - This routine is called to reset the
                    *           FLEX output switch and close any file
                    *           that might be open through the FILFCB
                    *           FCB.
                    *
         C4F9 RESET EQU   *
C4F9 86  01         LDA   #1
C4FB B7  CC22       STA   OSWTCH    reset output switch
C4FE B7  CC09       STA   PAU       enable pause
C501 7D  0834       TST   OPNFLG    is a file open?
C504 27  15         BEQ   RS1       return if not
                    *
                    * Output was going to file, close it
                    *
C506 B6  0835       LDA   TTYWID    restore TTY width
C509 B7  CD04       STA   WIDTH

C50C BE  09F3       LDX   #FILFCB   set FCB

C50F B6  04         LDA   #1CLOSE   close code
C511 A7  84         STA   FCBFC,X
C513 BD  D406       JSR   FMSCAL    call FMS
C516 27  03         BEQ   RS1       if ok return

C518 BD  CD3F       JSR   RPTERR    report error
         C51B RS1   EQU   *
C51B 7F  0834       CLR   OPNFLG    indicate no file open
C51E 39             RTS             and return

SYMBOL TABLE:

ADDBX  CD36  ASREAC 0001  ASWRIT 0002  BAC    0008  BADFIL 088C
BADIN  09B3  BAK    0005  BAS    0003  BELL   0007  BIN    0000
BINRY  0020  BS     CC00  BSE    CC07  BSIZ5  0080  BUFFER 0B36
BUFPNT CC14  CLASS  C021  CLC    C405  CLN    CC1A  CLOCK  F700
CMD    0002  CMDFLG CC28  COC    CC29  COLDS  C000  COMPR  C407
COPYIT C251  CR     000D  CR1F   0004  CURC   CC18  DAT    0007
DDRV   DE00  DEL    CC01  DEPTH  CC03  DIR    0009  DIRBEG 0001
DIREND 00FE  DIRMSG 076C  DIRTS  0005  DOCMND CD4B  DOS    CC00
DOSDIR 0000  DOSDSS 0004  DOSFSL 0000  DOSLEN 0008  DOSNXT 0000
DOSPRV 0002  DOSUNU 0005  DP1    C28E  DP2    C2A2  DP4    C2F4
DRV    0B33  DRVPNT 07F2  EJECT  CC08  ENBCOP 07C4  ENDIR  0788
ENTADR 0000  ENV    CC2D  EOF    0000  EOL    CC02  EOT    0004
ERR    C195  ERR02  C17F  ESC    CC0A  ESCRR  CC16  FACP   0040
FADP   0040  FARP   0020  FAWP   0080  FCBAS  0002  FCBASE D409
FCBCDA 002F  FCBCP  001E  FCBCRN 0020  FCBCUR D408  FCBDI  0022
FCBDN  0003  FCBEDA 0013  FCBESB 0001  FCBFA  000F  FCBFC  0B0D
FCBFCD 0019  FCBFDD 0032  FCBFS  0015  FCBFSR 0017  FCBLEN 0149
FCBLP  001C  FCBNAM 0004  FCBNAB 0024  FCBRI  0023  FCBRSI 0010
FCBRS2 0018  FCBSB  0040  FCBSCF 0038  FCBSCR 0035  FCBSDA 0011
FCBVER 0435  FCDDAY 001A  FCDMTH 0019  FCDYR  001B  FIA    CC26
FIB    0000  FIBEFS 0017  FIBEXT 0006  FIBFN  0000  FIBFS  000A
FIBFSI 0008  FIBFT  0009  FIBLEN 0018  FIBLSU 000D  FIBRSV 0011
FIBSUC 000F  FICF   CC2F  FILFCB 09F3  FLEX   CD00  FMS    D400
FMSCAL D406  FMSCLS D403  FMSERR CC20  FMSINT D400  FOA    CC24
FORMD  C354  FSLBOT 0011  FSLDC  0001  FSLDIF 0009  FSLDT  000A
FSLFF  0000  FSLLEN 0018  FSLLST 000D  FSLNIT 0008  FSLSC  000F
FSLSPD 0013  FSLUNU 0015  FSMRAN 0002  FSMSEQ 0000  FTYPE  0B0B
FXIST  0B8D  GET1   C2B8  GET2   C2CA  GET3   C2CC  GET4   C2C6
GETBLK C3C5  GETCHR CD15  GETDRV C2AA  GETFIL CD2D  GETHEX CD42
GETO   C27D  GT1    C2E9  GT2    C2FD  GT3    C306  GTBAD  C30A
GTGOOD C30C  GTSPEC C2CF  HEAD   0100  INBUF  CD1B  INCH   CD09
INCH2  CD0C  INDEC  CD48  INDRV  0B2B  INFILE 0B14  INTRB  091F
INVSPC 08A1  IOFLG  CC21  ISWTCH CC23  JT     C13B  LAD    CC1B
LCL    0000  LF     000A  LNEBUF CD80  LNKOUT C237  LOAD   CD30
LPC    C35F  LS1    0004  LS3    0009  LS5    0002  LSTRM  CC11
M0     C143  M1     C157  M2     C161  M3     C16C  MAP    CC00
MEMEND CC2B  MEM2   C11C  MENU   CJ19  MOVE   C3F1  MVC    C3EF
MX     C16F  NULL   CC05  NXTCH  CD27  OPN    0080  OPNFLG 0B34
OSWTCH CC22  OUT    0008  OUTADR CD45  OUTCH  CD0F  OUTCH2 CD12
OUTDEC CD39  OUTDRV 0787  OUTFIL 0B43  OUTHEX CD3C  P1     C4E7
P15    C4EF  P2     C4F8  PAU    CC09  PB1    C367  PB2    C38C
PB3    C3C2  PB4    C3BC  PB5    C3A0  PB6    C3AC  PB7    C3B4
PCRLF  CD24  PD1    C1D3  PD2    C1D1  PD3    C1C0  PDA    C21D
PDB    C1E3  PD6    C1FA  PDATA  C413  PDATA1 C415  PDATA2 C420
PDIR   C188  PDXIT  C22F  POUT   CCE4  PROMK  CCDB  PREVC  CC19
PRINIT CCC0  PROM   C125  PRT    000A  PRTBLK C35F  PRTSET C4BC
PSTRNG C01E  PSYS   0914  PUTCHR CD1B  R1     C498  R10    C4B9
R2     C4A9  R3     C4B0  R4     C45F  R6     C487  R7     C475
R8     C496  R9     C483  RDERR  0900  READSS C3E5  REKTER CD06
RESET  C4F9  RET    C358  ROUTE  C423  RPMOC  C410  RPTERR CD3F

RSTRIO CD2A  RSI    C518  SBDATA 0044  SBLINK 0040  SBRSI  0042
SCFMSC 00FF  SCFSC  0000  SCR    0006  SETEXT CD33  SETPAU C177
SFA    C980  SIRCRE 0023  SIRDAY 0024  SIRFSB 001D  SIRFSE 001F
SIRFSS 0021  SIRLEN 0028  SIRMTH 0023  SIRRTS 0026  SIRWAR 0010
SIRTS  0003  SIRVOL 001B  SIRYR  0025  SP     0020  SP4    B80F
SPS    C700  SRC1   C318  SRC2   C32C  SRC3   C32A  SRC4   C341
SRCHM  C30F  SRCXIT C350  SSIZ5  0080  START  C100  START1 C103
STAT   CD4E  STKA   C000  SVEFIL 0000  SYDR   CC0E  SYDRV  CC0B
SYS    0004  SYSCON CC4E  SYSCR1 CC0B  SYSCR2 CC2A  SYSCR3 CC30
SYSCR4 CCF8  SYSFCB 0840  TAB    C806  TABCOM 0009  TEMP   0002
TEX    0040  TMP4   0000  TOFILE 0677  TRADDR CC1E  TRFLG  CC1D
TTYWID 0B35  TXT    0001  UCA    C100  UCTA   CC12  URAM   0000
WARMS  CD03  WELCO2 C36A  WELCOM 0124  WIDTH  CC04  WKDRV  BC0C
XBOR   0016  XCLOSE 0004  XDELET 000C  XFWD   0014  XGIR   0007
XGRB   0011  XMSS   000F  XODIR  0006  XOREAD 0001  XOSIR  0010
XOUPUT 0003  XOWRIT 0002  XPIR   0008  XPOSN  0015  XPRB   0012
XRENAM 0008  XRESI  0008  XRES2  B00E  XRES3  0013  XREWND 0005
XRSS   0009  XRWNB  0000  XWSS   000A
```

```
*****************************************************
*                                                   *
*     TRS-80 to FLEX Utility                        *
*                                                   *
*        Written by: Scott R. Fraser                *
*                    547 Sharron Bay                *
*                    Winnipeg, Manitoba, Canada     *
*                    R2G 0M8                        *
*                    Ph. (204) 338-7641             *
*                                                   *
*****************************************************

*
* Introductory Information on TRS-80 NEWDOS-80 V.2
*
* We assume the NEWDOS/80 diskette's directory
* consists of only 2 granules, and is set up for
* 10 sectors/track, 2 granules/lump and 5 sectors/
* granule operations ( 5 sectors per granule is
* standard for NEWDOS/80).
*
* NEWDOS/80 makes all File Directory Entries (FDEs)
* of a diskette, except those for BOOT/SYS and
* DIR/SYS, available for use: thus, a 2 granule
* directory on a newly formatted data diskette has
* 62 FDEs available. NEWDOS/80 allows the directory
* to be allocated with up to 6 granules during
* diskette formatting, thereby providing for a
* maximum of 222 available FDEs.
*
* A diskette's directory always starts on a lump
* boundary and contains the GAT sector followed
* by the HIT sector followed by 8, 13, 18, 23
* or 28 FDE sectors, depending upon the number
* of 5 sector granules allocated to the directory.
* The starting lump number of the directory is
* ALWAYS contained as a hexadecimal value in the
* third byte of each diskette's 1st sector (that
* is, track 00, sector 00).
*
*  NOTE: Program tested for single sided,
*        single density disks only
*****************************************************
*
* Define DSECT for FDE
*
8000                    ORG     $8000
0000 0000  FDE          EQU     *
0000 0000  FDEF1F       RMB     1       free/inuse flag
0010 0010  FIFREE       EQU     %00010000 bit 4=0 means FDE free
0080 0080  FIFPDE       EQU     %10000000 bit 7=0 means FDE is an
                        *                 FPDE (otherwise an FDE
0040 0040  FIFSYS       EQU     %01000000 bit 6=1 if a system file
0008 0008  FIFINV       EQU     %00001000 bit 3=1 if file has invis. attr
           *
           * Bits 2-0 = access level code
           *
0001 0001  FDEIGN       RMB     1       ignore
0002 0002  FDERSV       RMB     1       reserved
0003 0003  FDEDFL       RMB     1       low order EOF byte
0004 0004  FDELRL       RMB     1       logical record length (0=256)
0005 0005  FDEFNM       RMB     8       file name
000D 000D  FDEEXT       RMB     3       extension name
0010 0010  FDEUPW       RMB     2       encoded update password
0012 0012  FDEAPW       RMB     2       encoded access password
0014 0014  FDEOFM       RMB     1       middle order EOF byte
0015 0015  FDEOFH       RMB     1       high order EOF byte
0016 0016  FDEXEL       RMB     8       start of 4, 2 byte extend elmt pairs
           *
           * 1st byte of pair:
           *
00FF 00FF  XELEND       EQU     $FF     end of extent elements
00FE 00FE  XELDEC       EQU     $FE     nxt byte contains DEC for
           *                               1st or nxt FDE
           *
           * otherwise is starting lump (track) of file
           *
           *
```

```
           * 2nd byte of pair (if 1st byte < $FE)
           *
           *  bits 7-5: number of granules from start
           *            of lump to start of area
           *            (ie., area offset within track)
           *
           *  bits 4-0: number less one of contiguous
           *            granules assigned to this area
           *
001E 001E  FDEND        RMB     2       extent elmt whose 1st byte is FF or FE
0020 0020  FDELEN       EQU     *       length of the FDE

0100                    ORG     $0100
           *
           ***** DEFINE PROMPTS USED BY ALL ROUTINES
           *


           * PDIR routine
           *
0100 0100  HEAD         EQU     *
0100 0D0A               FDB     CRLF
0102 4E 41 4D 45        FCC     /NAME    TYPE EXTENT-LIST/
011A 20 20 20 20        FCC     /          SIZE/
0128 0D0A               FDB     CRLF
012A 04                 FCB     EOT

0128 0128  WELCOM       EQU     *
0128 0D0A               FDB     CRLF
012D 20 59 4F 55        FCC     / YOUR MOTHER WEARS ARMY BOOTS!/
0148 04                 FCB     EOT

014C 014C  DIRMSG       EQU     *
014C 0D0A               FDB     CRLF
014E 20 20 20 20        FCC     /      Directory of Drive /
0167 00    0167 0167    OUTDRV  FCB     0       put drive# here
0168 0D0A               FDB     CRLF
016A 04                 FCB     EOT

0168 0168  ENDIR        EQU     *
0168 20 20 20 45        FCC     /  End of Directory Listing. /
0188 48 69 74 20        FCC     /Hit [Return] to Continue.../
01A3 04                 FCB     EOT

01A4 01A4  ENDCOP       EQU     *
01A4 20 20 20 45        FCC     /  End of Copy. Enter /
01B9 5B 52 65 74        FCC     /[Return] to Continue.../
01D1 04                 FCB     EOT

01D2 01D2  DRVPMT       EQU     *
01D2 20 20 20 45        FCC     '  Enter drive of TRS disk: '
01EE 04                 FCB     EOT

           *
           * BLANOUT routine
           *
01EF 20 20 20 20  SP4   FCC     /   /   4 spaces
01F3 04                 FCB     EOT

           *
           * COPYIT routine
           *
01F4 01F4  INFILE       EQU     *
01F4 20 20 20 45        FCC     /  Enter "from" file: /
020A 04                 FCB     EOT

0208 0208  INDRV        EQU     *
0208 20 20 20 45        FCC     /  Enter "from" drive: /
0222 04                 FCB     EOT

0223 0223  OUTFIL       EQU     *
0223 20 20 20 4F        FCC     /  Output to File(F), Printer(P), /
0245 6F 72 20 54        FCC     /or Terminal(Te)? /
0256 04                 FCB     EOT

0257 0257  TOFILE       EQU     *
0257 20 20 20 45        FCC     /  Enter "to" file: /
0268 04                 FCB     EOT
026C 026C  BADFIL       EQU     *
026C 46 49 4C 45        FCC     /FILE DOES NOT EXIST!/
0280 04                 FCB     EOT
```

```
                0281  INVSPC  EQU   *
0281 49 4E 56 41        FCC   /INVALID FILE SPECIFICATION!/
029C 04                 FCB   EOT

                0290  FXIST   EQU   *
029D 53 4F 52 52        FCC   /SORRY, FILE ALREADY EXISTS/
02B7 04                 FCB   EOT

                02B8  FTYPE   EQU   *
02B8 20 20 20 49        FCC   /   Is file type Binary(B) /
02D2 6F 72 20 54        FCC   /or Text(T)? /
02DF 04                 FCB   EOT

                      *
                      * Misc
                      *
02E0 3C 3C 3C 20  RDERR FCC   /<<< READ ERROR >>>!/
02F3 04                 FCB   EOT

02F4 50 52 49 4E  PSYS  FCC   /PRINT/
02F9 00 00 00           FCB   0,0,0
02FC 53 59 53           FCC   /SYS/

                02FF  INTRO   EQU   *
02FF 0D0A 0D0A          FDB   CRLF,CRLF
0303 20 20 20 46        FCC   "   FLEX/TRS-80 UTILITY (C) 1980"
0322 20 62 79 20        FCC   " by Scott Fraser"
0332 0D0A 0D0A          FDB   CRLF,CRLF
0336 20 20 20 20        FCC   /          0 - Directions/
034E 20 20 20 20        FCC   "          1 - TRS Directory"
0369 0D0A               FDB   CRLF
036B 20 20 20 20        FCC   /          2 - Copy File/
0382 20 20 20 20        FCC   /           3 - EXIT Program/
039D 0D0A 0D0A          FDB   CRLF,CRLF
03AE 20 20 20 45        FCC   /   ENTER SELECTION: /
03B5 04                 FCB   EOT

                0386  BADIN   EQU   *
0386 20 20 20 49        FCC   /   INVALID SELECTION, REENTER: /
03B5 04                 FCB   EOT

0306            FILFCB  RMB   FCBLEN    reserve FCB area

                      *
                      * Define some TRS constants
                      *
                0100  SSIZ5   EQU   256       #bytes per sector (5")
                0005  GRANUL  EQU   5         # sectors per granule
                0089  MAXSEC  EQU   $89       maximum possible sector on track
                0002  DIRS1Z  EQU   2         # granules for directory
                0000  DIRPTR  EQU   $0000     trk/sec of ptr byte to dir
0516 0000       DIRBEG  FDB   $0000     trk/sec of start of TRS dir
                0009  TABCHR  EQU   $09       universal tab char (Ctrl)I)

051B            DRV     RMB   1         holds current drive#
0519 00         OPNFLG  FCB   0         file open bits
                0080  OPN     EQU   %10000000 #File is open
                0040  TEX     EQU   %01000000 #file is text
                0020  BINRY   EQU   %00100000 #file is binary
051A            TTYWID  RMB   1         save TTY Width value here

                0500  BSIZ5   EQU   SSIZ5*GRANUL size of buffer (5")

051B            BUFFER  RMB   BSIZ5     alloc max buffer size




C100                    ORG   $CA
C100 20  01     START   BRA   START1    ***** Begin Program *****

C102 04                 FCB   4         <<<< VERSION # >>>>



                C103  START1  EQU   *
C103 10FE CC2B          LDS   #MEMEND   initialize system stack ptr

C107 B6  CC0C           LDA   >WRKDRV   default to work drive
C10A B7  051B           STA   DRV       save default

C10D 8E  C119           LDX   #MENU
C110 BF  CC16           STX   ESCRR     modify escape return register
C113 B6  #C04           LDA   WIDTH     set current TTY line width
C116 B7  051A           STA   TTYWID    and save it

                C119  MENU    EQU   *
C119 7F  CC1A           CLR   CLN       clear current line num ist
                C11C  MEN2    EQU   *
C11C 7C  CC22           INC   DSWTCH    set output switch
C11F 8E  02FF           LDX   #INTRO    print intro mss
C122 BD  CD1E           JSR   PSTRNG    ask for selection

                C125  PROM    EQU   *
C125 BD  CD09           JSR   INCH      get a response
C128 BD  CD21           JSR   CLASS     classify it
C129 25  36             BCS   MX        bad selection
C12D B1  33             CMPA  #'3       over 3?
C12F 22  32             BHI   MX        bad selection

C131 84  0F             ANDA  #%00001111 keep low 4 bits only
C133 48                 ASLA            selection=selection*2
C134 8E  C138           LDX   #JT       addr of jump table
C137 30  86             LEAX  A,X       addr proper selection
C139 6E  84             JMP   0,X       goto selection

C13B 20  06       JT    BRA   M0        selection=0

C13D 20  0C             BRA   M1        =1
C13F 20  14             BRA   M2        =2
C141 20  1D             BRA   M3        =3

                C143  M0      EQU   *     do selection 0
C143 8E  012B           LDX   #WELCOM   print instructions
C146 BD  CD1E           JSR   PSTRNG
C149 20  20             BRA   SETPAU

                C14B  M1      EQU   *
C14B BD  32             BSR   PDIR      print dir
C14D 8E  016B           LDX   #ENDDIR
C150 BD  CD1E           JSR   PSTRNG
C153 20  16             BRA   SETPAU

                C155  M2      EQU   *
C155 17  0170           LBSR  COPYIT    copy file
C158 8E  01A4           LDX   #ENDCOP
C15B BD  CD1E           JSR   PSTRNG
C15E 20  0B             BRA   SETPAU

                C160  M3      EQU   *
C160 7E  CD03           JMP   WARMS     return to FLEX

                C163  MX      EQU   *     bad selection
C163 8E  0386           LDX   #BADIN
C166 BD  CD1E           JSR   PSTRNG    print mss
C169 20  BA             BRA   PROM

                C16B  SETPAU  EQU   *
C16B B6  CD03           LDA   DEPTH     pause (if set)
C16E B7  CC1A           STA   CLN       so user can read stuff
C171 20  A9             BRA   MEN2

                C173  ERR02   EQU   *
C173 BD  CD3F           JSR   RPTERR    report error first
C176 8E  02E0           LDX   #RDERR    set mss

                C179  ERR     EQU   *
C179 BD  CD1E           JSR   PSTRNG    print mss
C17C 7E  CD03           JMP   WARMS     and return to FLEX

                      *
                      * Name      - PDIR
                      * Function  - This routine is called to print out
                      *             directory information on a TRS disk.
                      *
                      *
                      *             No parms are required
                      *             All registers are used and not restored
                      *
                C17F  PDIR    EQU   *
C17F 32  E9 FF89        LEAS  -LS1,S    alloc local storage
```

FOR THE ELEKTRA

- OS-9™ with Editor, Assembler, and Debugger **$250.00**
- STAR-DOS™ Level 1 (FLEX™ compatible but with up to 10 active drives: (i.e. 4 8" floppies, 4 5" floppies, and 2 Winchester drives) **$75.00**
- STAR-DOS™ Adaptation Guide **$50.00**

FOR OS-9™ by Epstein Associates

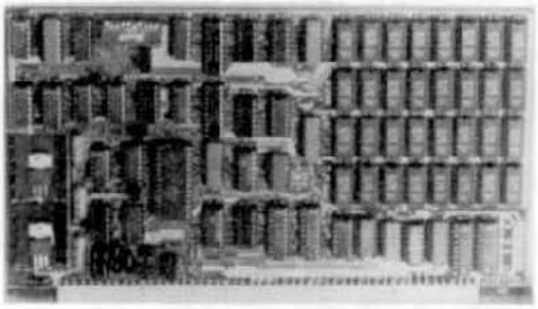
- Super Modem Program with autodial, configuration file, etc. (Available exclusively through AAA Chicago Computer Center) **$100.00**

THE ELEKTRA SUPER FLOPPY CONTROLLER

- Emulates the DC-1, DC-2, DC-3, DC-4 as well as the GIMIX #28, #38, #48, and #58 controllers **$295.00**

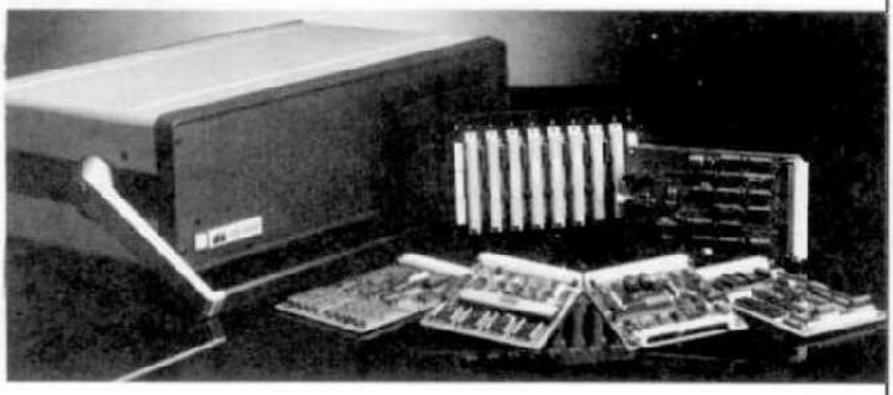
RACKMOUNT ELEKTRA COMPUTER CABINET

- 17" W x 21.5" D x 6.7" H
  Holds two half height 5¼" floppy drives **$250.00**



2MHz 256K Memory Board
with on board DAT by
COMPUTER EXCELLENCE, INC.
$749.00

Now Available
68010
and
VME Bus
by
MIZAR INC.
Phone for details



Phone:
AAA Chicago Computer Center
Technical Consultation available most weekdays from 4 p.m. to 6 p.m. CST
(312) 459-0450 120 Chestnut Lane Wheeling, IL 60090
See our catalog and ordering information on the next page.

ELEKTRA COMPUTER SYSTEM Includes chassis, dual port serial interface with two cables, CPU 8/9, 4K Humbug, 56K static RAM, super floppy controller with inboard ribbon cable, Star-Dos, dual 80 track DSDD floppy drives (other combinations available; phone) $2795.00

ELEKTRA COMPUTER CABINET THE LARGEST SS-50 COMPUTER CABINET AVAILABLE! Made of heavyweight 0.090" thick aluminum. Interior is 18-1/2" wide by 21-7/8" deep by 6-3/4" high. Heavy duty A.C. line cord. A.C. fuse holder. EMI filter. Fan with filter. Back panel has 10 cutouts for 'D' type data connectors. Front panel has key on/off power switch, 2 illuminated push button switches (Reset and NMI/Abort), and two cutouts for 5-1/4" disk drives. $250.00

RACKMOUNT ELEKTRA COMPUTER CABINET 17"w x 21.5"d x 6.7"h. Holds two half height 5-1/4" floppy drives. $250.00

Filler Plate for 5-1/4" drive opening: $10.00 Fan Filter: $10.00

POWER SUPPLY. Highest quality linear power supply CONSERVATIVELY rated at 15a @ 8v, 3a @ 16v, 3a @ -16v. Multi-tapped primary for line tuning $200.00

DISK REGULATOR BOARD WITH CABLES Standard version for 2 floppy drives $50.00
Heavy duty version for 1 Winchester drive and 1 floppy drive. $75.00

AUXILIARY POWER SUPPLY to power second Winchester drive $125.00

ELEKTRA UNIVERSAL SS-50/SS-50C MOTHERBOARD Heavyweight 0.125" thick, 18" long by 9" wide. 11 memory (50 pin) slots. 8 I/O (30 pin) slots. Complete address decoding and selection, as well as extended address capability for I/O slots. Choice of 4, 8, or 16 addresses per I/O slot. 1" spacing between all memory and I/O slots. On board baud rate generator with low and high ranges providing jumper selectable rates of 75 through 38,400 for each of the five baud rate lines. Slow device circuitry permitting 1 MHz 30 pin disk controllers to run with 2MHz 50 pin CPU boards.
Mounting hardware $5.00 Bareboard w/documentation $80.00
Kit w/gold connectors: $400.00 Assembled w/gold connectors $480.00
Kit w/pin connectors: $320.00 Assembled w/pin connectors: $380.00

ELEKTRA CHASSIS Includes cabinet, 110v power supply, power supply cables, standard disk regulator board with power cables, motherboard with gold square pin connectors, assembled and tested. $950.00

ELEKTRA CPU 8/9 Use either the 6502 or 6809 (to run 6800 software) or 6809 as provision for up to 32768 Eproms, 1K scratchpad, MC6840 triple timer, and an optional baud rate generator providing baud rates from 110 through 38,400 baud in two user selectable ranges. Run OS-9™, FLEX™, STAR-DOS.
Bareboard: $50.00 Kit: $225.00 Assembled: $275.00
Optional Baud Rate Generator. $25.00

ELEKTRA DPS DUAL PORT SERIAL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 or 16 addresses per port. RTS, CTS, DTR, DCD, IRQ, FIRQ, NMI, and baud rate can be appropriately implemented for each port.
Bareboard: $25.00 Kit: $75.00 Assembled: $95.00
Cable with jack socket assemblies (two needed per board) Each: $25.00

ELEKTRA DPP DUAL PORT PARALLEL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 or 16 addresses per I/O slot. The direction of the TTL buffers can be controlled by either on board jumper connectors or by a signal from the peripherals. The interrupt request line for each port may be individually jumpered to either the IRQ or FIRQ/NMI bus line.
Bareboard: $25.00 Kit: $60.00 Assembled: $80.00
Cable with jack socket assemblies (two needed per board) Each: $25.00

ELEKTRA 64K STATIC RAM/ROM MEMORY BOARDS with gold connectors (pin available) Assembled and tested. With 56K RAM $269.00 With 64K RAM $299.00
Kit With 56K RAM $219.00 With 64K RAM $249.00

ELEKTRA UNIVERSAL SUPER FLOPPY CONTROLLER THE BEST 30 PIN FLOPPY DISK CONTROLLER THAT YOU CAN BUY! Controls up to four 5-1/4" drives and four 8" drives for a total of eight system drives. (FLEX system limit is four drives.) Single density or double density. 1MHz or 2MHz, 6800 or 6809. (Double density 8" must be at 2MHz; all other combinations of performance are possible.) Analog phase locked loop data separators with separate adjustments for 5" and 8" drives. Analog write precompensation circuit with separate adjustments for 5" and 8" drives. Designed to meet the data hold requirements of Western Digital floppy controller IC. Assembled and tested. $295.00
Disk with drivers, setup, and formatting utilities. Specify FLEX 2.0, 6800 Gen FLEX, FLEX 9.0, FLEX 9.1, or 6809 Gen FLEX, 5" or 8" $30.00
Disk with drivers for OS-9 (Specify 5" or 8") $50.00
STAR-DOS (Specify ELEKTRA or OC-2, 5" or 8") $75.00 Adaptation guide $50.00

ELEKTRA WINCHESTER SYSTEMS THE BEST WINCHESTER SYSTEMS THAT YOU CAN BUY! Has automatic error detection and CORRECTION of up to 11 bit burst errors. SS-50 bus, extended addressing capabilities, DMA, on board sector buffer, drivers included for 6809 FLEX or OS-9. Specify whose version of FLEX that you are using. Drivers for FLEX2 (6800) are available for an additional $100.00. Price includes host interface controller, drive(s), and cables.

7 Megabyte single drive sys.	$1995.00	14 Megabyte dual drive sys.	$2995.00
12 Megabyte single drive sys.	$2295.00	24 Megabyte dual drive sys.	$3595.00
19 Megabyte single drive sys.	$2995.00	38 Megabyte dual drive sys.	$4695.00

(19 Megabyte drives are the largest that can be supported by FLEX)
Circuit boards, cables, software (No drives) 995.00
SS-50C DMA Bus interface board only 695.00

ELEKTRA HD-5 Cabinet for dual 5 1/4" floppy drives with power supply, line cord, fuse, power switch, and power cables to drives. 150.00

ELEKTRA H-5W as above but with EMI filter, fan, and heavy duty power supply. Powers 1 floppy and 1 Winchester. 199.00

5' ribbon cable for dual outboard 5-1/4" disk drives 40.00
2' ribbon cable for dual inboard 5-1/4" disk drives 35.00
Custom cables available Phone

ELEKTRA HD-8 Dual 8" drive cabinet, EMI filter, fan with filter, power supply and power supply cables. 350.00

8' ribbon cable for dual 8" disk drives 45.00

ELEKTRA 38 PIN PROTOTYPING BOARD 20.00
ELEKTRA 50 PIN PROTOTYPING BOARD 40.00

GOLD 10 PIN CONNECTORS (Specify male with square pins or female) 1.50
TIN 10 PIN CONNECTORS (Specify male with square pins or female) .50

ELEKTRA is a trademark of AAA Chicago Computer Center.
FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc.
HELIX is a trademark of Hazelwood Computer Systems.
OS-9 and BASIC09 are trademarks of Motorola Inc. and Microware Systems Corp.

AAA CHICAGO COMPUTER CENTER (312) 459-0460
120 CHESTNUT LANE @ WHEELING, IL 60090
Technical consultation available 4 PM to 6 PM most weekdays. Closed evenings and weekends.

TERMS Minimum order $20.00. Shipping and handling estimates within the Continental U.S., add 3% (MINIMUM $2.50). Illinois residents add 7% sales tax. We will refund your overestimated shipping and handling charges. Foreign shipping and handling, add 10% (MINIMUM $10.00). Foreign orders must be prepaid in U.S. dollars. Checks must be drawn on a U.S. bank. Heavy foreign items will be shipped air freight collect. Please phone between 4 PM and 6 PM weekdays if questions arise regarding shipping fees. Master Charge, Visa, and American Express honored.

Our apology: We are not staffed to answer technical inquiries through the mail. Please phone for technical help during the hours indicated above. The too frequent changing of our inventory and prices makes it uneconomical to publish a catalog. Our ads are intended to serve that purpose. Prices and inventory are subject to change without advance notice.

SUPER MODEM PROGRAM Single character commands. No interrupts required. Transmit manually or transmit disk files (text) of any length to distant computer. Receive and save disk files (text) on local disk system. X-on/X-off supported. Tested for full duplex at speeds up to 9600 baud. Half duplex option. Echo option. Replaces CR with CR/LF (user option). Slow disk file transmit option.
Please specify 6800 or 6809, SSB or FLEX™, 5" or 8".
Instruction Manual and disk with both source and object code $75.00

OS-9 Super Modem Program by Ep lein Associates with autodial configuration file, etc. 100.00

ALL IN ONE
Editor — Text Processor — Mailing Labels — Mailing Lists — Multiple Form Letters
Use any CRT terminal and printer — Best Package For The Money Anywhere!
Specify 6800 or 6809, SSB or FLEX™, 5" or 8" 75.00
Add $35.00 for printed source listing; add $100 for source on disk.
All-In-One, Write'n spell, and Spell'n Fix package 250.00

Software by Technical Systems Consultants, Inc.	FLEX™ Source (List)	FLEX™ Source (Disk)	FLEX™ Man. Only	FLEX™ Object w/Man.	UniFLEX™ Add. Man.	UniFLEX™ Man. Only	UniFLEX™ Object w/Man.
Gen FLEX w/Edit & ASMB	—	—	—	250	—	—	—
FLEX 9.1 (OC-2) w/Edit & ASMB	—	—	25	150	40	100	550
Advanced Programmers Guide	—	—	25	—	—	—	—
Editor	100	250	25	50	—	—	—
Assembler	150	250	25	50	—	—	—
Debug	175	250	25	75	—	—	—
Extended Basic	—	—	25	100	20	50	200
Basic Precompiler	—	—	25	50	10	25	150
Sort/Merge	—	—	25	75	20	35	150
Utilities	—	In	25	75	10	25	150
Diagnostics	—	—	25	75	—	—	—
ext Processor	150	250	25	75	20	35	150
68000 X-ASMB on 6809	—	—	25	250	20	35	300
Pascal	—	—	50	200	25	50	300
Rel ASMB/Linking Loader	—	—	25	150	20	35	175
6800 X-ASMB on 6809	—	—	—	100	—	—	—
Cobol	—	—	—	—	30	75	750
Fortran 77	—	—	—	—	35	65	450

Software by Microware Systems Corp. (Suggested List Prices, varies/mfg)	Run-Time Package	Source	Manual Only	Object w/Man.
OS-9™ Level 1 w/Edit, Asm, Debug		400.00	40.00	250.00
OS-9™ Level 2 w/Edit, Asm, Debug		400.00	40.00	500.00
OS-9™ Edit, Asm, Debug Pkg			25.00	125.00
Device Driver for Disk Controller (Specify model)		100.00		
Device Driver for ACIA and PIA		50.00		
Clock Driver for 6840 and 58167 clock chips		35.00		
Entertainment Pack I, or File Handler Toolbox, or NiteOwl			10.00	65.00
Print Spooler (Level 2 only)			15.00	95.00
Virtual Disk Driver (Level 2 only)			10.00	85.00
RMA Relocatable Macro Assembler			20.00	125.00
RMA/68000 Cross Assembler				400.00
BASIC09™ W/Run-Time		50.00	N/A	25.00	200.00
BASIC09 Tour Guide Book				18.95
"C" Compiler				25.00	250.00
C Programming Language (Kernighan & Ritchie)				19.95
CIS Cobol Compiler w/Forms2 Prog. Gen.		50.00	N/A	40.00	400.00
Pascal Compiler		50.00	N/A	25.00	250.00
Sage Application Generator		300.00	N/A	25.00	995.00
Microware yearly support service (All products)					150.00
Edition Update w/o manuals 25.00		Version Update w/manuals			75.00

Special Software

STAR-DOS Level I (Specify ELEKTRA or OC-2)	$75.00	Adaption guide	$50.00
2K MICROBUG	40.00	4K HUMBUG 75.00	Custom versions $85.00
Spell'n Fix by Peter Stark	178.58	Write'n Spell by Peter Stark	75.11
All-In-One, Spell'n Fix, and Write'n Spell package			250.00
SUPER SLEUTH Disassembler System ($101.00 for OS-9 version)			90.00

SO/DD DISK DRIVES	1 head Tandon	2 heads Tandon	2 heads CDC	1 head MPI	2 heads MPI
30 day guarantee					
5-1/4", 40 tracks	225.00	300.00	300.00	250.00	325.00
5-1/4", 80 tracks	300.00	375.00	375.00	325.00	400.00
MPI or CDC Service Manual (Specify 40 or 80 track)			25.00	Qume QT-8	550.00

OUTBOARD EPROM PROGRAMMERS BY OPTIMAL TECHNOLOGY

Model EP-2A-79 (Personality modules extra)	169.00
Optimal Technology, Inc. 30 pin parallel I/O board for EP-2A-79	37.00
FLEX™ Software package for EP-2A-79 (Specify 6800 or 6809)	30.00
OS-9 Software package for EP-2A-79	10.00
Model EP-2B-87 (RS-232/20 MA, Motorola Imt. 8K buffer, 1200/9600 baud)	575.00
Model EP-2B-88-4 (Copies 1 to 4 EPROMS)	550.00
Personality Copy Modules for 2708, 2716, 27C16, 2732, 27C32, 2732A, 2758, MCM68764, MCM68766, 2764, 27C64, 2764A, 27128, 27128A, 27256, 27C256, 2508, 2516, 2532, 2564, 25128, 2816, 2816A, R87C32, 8751, 8E70	$37 to $39

SPECIALS

* U.S. Robotics 1200 baud direct connect auto answer modem			299.00
* SSB BFO Floppy Disk Controller (Version 3) Run FLEX or SSB DOS			100.00
* SWTPC 4K Memory	$15.00	MP-MB (4K board)	9.95
* SWTPC MP-N (Kit)*	$54.95	S-32* RAM not included	124.50
* SWTPC MP-09 2MHz CPU*	$295.00	While supplies last	
* High speed tape reader	50.00	300 Baud acoustic modem	129.00
* TI810 Printer w/lower case and full vertical forms control			1200.00

SPECIAL BOARDS

Microtime II Calendar and Clock Board (Assembled)	60.00
Data Mart 16K EPROM bareboard (2708 chips)	30.00

Smoke Signal Broadcasting

DCB-4A Double Density Controller Board for 5" and 8" with DOS	549.00
SSB DOS (Specify 6800 or 6809, BFD or DCB-4A, 5" or 8")	75.00
SSB version of FLEX™ (While supply lasts)	150.00
LMB-1A Motherboard	399.00
SCB-69 6809 CPU Board	399.00
Chief 9524 64K Computer System with OS-DT-DD 5" FD	4325.00
Dynamic Memory Boards M-128-X 995.00 M-2 -X 1295.00 M-512-X	1695.00

HELIX

64K 6809 Computer	$2395.00	Other computer systems available	
DMA 5" and 8" Floppy Controller	495.00	6809 CPU Board	495.00
68008 board for SS-50	595.00	CP/M-68K	350.00

GIMIX CLEARANCE SALE	LIST	OUR PRICE		LIST	OUR PRICE
Cable (Par I/O)	24.95	20.00	6800 CPU board	2 4 0 3	100.00
Double disk reg. card	68.22	50.00	#28 controller w/Flex	328.28	270.00
64 X 16 Video Boards	198.71	100.00	Single prt ser, 1 cable	113.35	90.00
16K Mem Bds. w/cntrl reg.		145.00	Dual prt par, 2 cables	138.32	110.00
93L422 DAT chip	17.50	15.00	4K PPD PROM Bd. and burner		100.00

WARNING AAA Chicago Computer Center does not provide repair or diagnostic service for customer assembled kits. AAA Chicago Computer Center does warranty and maintain service for our assembled boards. The customer should carefully take into consideration the small differential separating out kit and assembled prices when making his choice of purchase.

```
C183 8E  0102          LDX   #DRVPMT  ask for a drive
C186 BD  CD1E          JSR   PSTRNG
C189 17  0189          LBSR  GETDRV   get drive #
C18C 1025 0095         LBCS  PDIIT    if bad then leave

C190 17  0381          LBSR  ROUTE    where to route output
C193 1025 008E         LBCS  PDIIT    leave if bad
C197 BD  CD24          JSR   PCRLF
C19A B6  0518          LDA   DRV      get drive
C19D B7  C843          STA   SYSFCB+FCBDN save in FCB
C1A0 8A  30            ORA   #'0      convert to ascii
C1A2 B7  0167          STA   OUTDRV   save to print

C1A5 8E  014C          LDX   #DIRMSG  print msg
C1A8 17  0359          LBSR  PDATA

C1AB 8E  0100          LDX   #HEAD    print dir header
C1AE 17  0353          LBSR  PDATA

C1B1 8D  7A            BSR   FINDIR   find beginning of directory
C1B3 B6  02            LDA   #DIRSIZ  get size of dir in granules
C1B5 A7  E9 0006       STA   TMP5,S   and save
C1B9 FC  0516          LDD   DIRBEG   get begin of directory

         C1BC  PD3     EQU   *
C1BC 17  0202          LBSR  GETGRN   get a granule of dir info
C1BF ED  E9 0002       STD   TEMP,S   save next trk/sec to read
                       *
                       * BUFFER is full of directory data.
                       *
                       * If this is the first granule of dir
                       * info, then skip over GAT and HIT data
                       *
C1C3 8E  0518          LDX   #BUFFER  position to buffer begin
C1C6 A6  E9 0006       LDA   TMP5,S   get current dir granule
C1CA B1  02            CMPA  #DIRSIZ  on first granule?
C1CC 26  04            BNE   PD1      branch if not
C1CE 30  89 0200       LEAX  SSIZ5*2,X skip 2 sectors worth
         C1D2  PD1     EQU   *
C1D2 AF  E9 0000       STX   ENTADR,S save entry addr

C1D6 A6  84            LDA   FDEFIF,X
C1D8 85  10            BITA  #FIFREE  valid file entry?
C1DA 27  33            BEQ   PD4      branch if not

                       *
                       * Now print the file name
                       *
C1DC 30  05            LEAX  FDEFN,X  field to move
C1DE C6  08            LDB   #11      length to move
         C1E0  PD5     EQU   *
C1E0 A6  80            LDA   0,X+     get a char
C1E2 26  02            BNE   *+2+2    if =0 then skip
C1E4 86  20            LDA   #SP
C1E6 BD  CD1B          JSR   PUTCHR   print the char
C1E9 5A                DECB
C1EA 27  08            BEQ   PD6      exit if all done
C1EC C1  03            CMPB  #3       just extension left?
C1EE 26  F0            BNE   PD5      branch if not
C1F0 86  2E            LDA   #'.      print a '.' first
C1F2 BD  CD1B          JSR   PUTCHR
C1F5 20  E9            BRA   PD5      and continue

         C1F7  PD6     EQU   *
C1F7 AE  E9 0000       LDX   ENTADR,S
C1FB 30  8B 16         LEAX  FDEIEL,X print extent element pairs
C1FE BD  47            BSR   PRTIEL

C200 BD  6C            BSR   GETSIZ   find file size in sectors
C202 FD  0004          STD   OUTSIZ   save size
C205 8E  0004          LDX   #OUTSIZ  and print value
C208 5F                CLRB
C209 BD  CD39          JSR   OUTBIC
C20C BD  CD24          JSR   PCRLF

         C20F  PD4     EQU   *
C20F AE  E9 0000       LDX   ENTADR,S
C213 30  8B 20         LEAX  FDELEN,X goto next file entry
C216 8C  0418          CMPX  #BUFFER+#SIZ5 finished?
C219 25  B7            BLO   PD1      branch if not
```

```
C21B EC  E9 0002       LDD   TEMP,S   read in next granule
C21F 6A  E9 0006       DEC   TMP5,S   all done?
C223 26  97            BNE   PD3      continue if not

         C225  PDIIT   EQU   *
C225 17  03C2          LBSR  RESET    clear things up first

C228 32  E9 0007       LEAS  LS1,S    release local storage
C22C 39                RTS           return

                       *
                       * Data area for PDIR routine
                       *
         C220  LPC     SET   *
0000                   ORG   $0000
         0000  LCL     SET   *
0000     ENTADR RMB    2
0002     TEMP   RMB    2
0004     OUTSIZ RMB    2
0006     TMP5   RMB    1       holds size of dir in granules
         0007  LS1     EQU   *-LCL   len of local area

C220                   ORG   LPC     restore PC

                       *
                       * Name      - FINDIR
                       *
                       * Function - This routine is called to find
                       *           the starting lsap of the directory
                       *
                       *           All registers are preserved, and
                       *           the TRS dir's starting trk/sec is
                       *           stored in variable DIRBEG
                       *
         C220  FINDIR  EQU   *
C220 34  12            PSHS  A,X      save regs

C222 8E  C840          LDX   #SYSFCB  point to an FCB
C225 CC  0000          LDD   #DIRPTR  sector to read
C228 17  029E          LBSR  READSS   read the sector
C22B 1026 FF37         LBNE  ERR02    leave if error
C22F 30  88 40         LEAX  FCBSB,X  point to data portion
C232 A6  02            LDA   2,X      get dir lump value
C234 B7  0516          STA   DIRBEG   save it
C237 35  12            PULS  A,X      restore regs
C239 39                RTS           and return


                       *
                       * Name      - PRTIEL
                       * Function - This routine will print the 4 extended
                       *           element pairs in an FDE entry.
                       *
                       *           On entry, X -> beginning of the extended
                       *           element pair list.
                       *(
                       *           All registers are preserved
                       *
         C23A  PRTIEL  EQU   *
C23A 34  16            PSHS  D,X      save regs

C23C 86  20            LDA   #SP      print a space first
C23E BD  CD1B          JSR   PUTCHR

C241 C6  04            LDB   #4       4 pairs to print
C243 86  02            LDA   #2       bytes per pair
C245 34  02            PSHS  A        save on stack
         C247  PI1     EQU   *
C247 BD  CD3C          JSR   OUTHEX
C24A 30  01            LEAX  1,X      go to next byte
C24C 6A  E4            DEC   0,S      done with this pair?
C24E 26  F7            BNE   PI1      branch if not

C250 86  20            LDA   #SP      print a space
C252 BD  CD1B          JSR   PUTCHR
C255 86  02            LDA   #2
C257 A7  E4            STA   0,S      save on stack
C259 5A                DECB           done with all pairs?
C25A 26  EB            BNE   PI1      continue if not

C25C 32  61            LEAS  1,S      pop stack
```

```
C26B 35 16          PULS  D,X
C26D 39             RTS             return


                    *
                    * Name      - GETSIZ
                    * Function  - This routine determines the size
                    *             of a TRS file
                    *
                    *             On entry the X reg should point to
                    *             the beginning of the extent element
                    *             list.
                    *
                    *             On exit, reg D contains the size of
                    *             the file in granules
                    *
                    *             All regs are preserved
                    *
          C26E      GETSIZ EQU    *
C26E 34 10          PSHS  X         save regs

C270 32 E9 FFFD     LEAS  -LS2,S    alloc local storage

C274 6F E9 0000     CLR   FSIZ,S    clear file size area
C278 6F E9 0001     CLR   FSIZ+1,S
C27C C6 04          LDB   #4        4 pairs to analyze
C27E E7 E9 0002     STB   CTR1,S    save pair ctr
          C282      GS    EQU     *
C282 8D 1F          BSR   GETXEP    get extent elmt pair info
C284 5D             TSTB            end of extent?
C285 27 11          BEQ   GS2       branch if so

C287 4F             CLRA            clear A for summing
C288 E3 E9 0000     ADDD  FSIZ,S    add to cumulative sum
C28C ED E9 0000     STD   FSIZ,S    and save back
C290 30 02          LEAX  2,X       goto next pair
C292 6A E9 0002     DEC   CTR1,S    done yet?
C296 26 EA          BNE   GS        continue if not

          C298      GS2   EQU     *
C298 EC E9 0000     LDD   FSIZ,S    get file size
C29C 32 E9 0003     LEAS  LS2,S     free storage
C2A0 35 10          PULS  X         restore regs
C2A2 39             RTS             and return
                    *
                    * Data area for GETSIZ
                    *
          C2A3      LPC   SET     *
0000                ORG   $0000
          0000      LCL   SET     0
0000                FSIZ  RMB     2
0002                CTR1  RMB     1
          0003      LS2   EQU     0-LCL   len of local storage
C2A3                ORG   LPC       restore PC


                    *
                    * Name      - GETXEP
                    * Function  - This routine analyses one pair of bytes
                    *             in the extent element list of the FDE.
                    *
                    *             On entry, the X reg points to an
                    *             extent element pair
                    *
                    *             On exit, reg B will contain the number
                    *             of granules in the area, and Y will
                    *             contain the starting trk/sec of the
                    *             area in question.
                    *             If the pair represents a non-existent
                    *             area, then B will be zero.
                    *
                    *             All other regs are preserved
                    *
          C2A3      GETXEP EQU    *
C2A3 34 12          PSHS  A,X       save regs

C2A5 A6 84          LDA   0,X       get 1st byte of pair
C2A7 81 FF          CMPA  #$EL.END  end of pair?
C2A9 27 19          BEQ   GXXIT     branch if so
C2AB 81 FE          CMPA  #$EL.DEC
```

```
C2AD 27 15          BEQ   GXXIT

                    *
                    * Find starting trk/sec of area
                    *
C2AF A6 01          LDA   1,X       get 2nd byte of pair
C2B1 44             LSRA            get bits 5-7
C2B2 44             LSRA
C2B3 44             LSRA
C2B4 44             LSRA
C2B5 44             LSRA
C2B6 C6 05          LDB   #EXPANL
C2B8 3D             MUL             calc start sec of area
C2B9 A6 84          LDA   0,X       get start trk of area
C2BB 1F 02          TFR   D,Y       and save in Y

                    *
                    * calculate size of area in sectors
                    *
C2BD E6 01          LDB   1,X       get 2nd byte of pair
C2BF C4 1F          ANDB  #%00011111 get # granules-1
C2C1 5C             INCB            # granules
C2C2 20 01          BRA   GXRETN    and leave
          C2C4      GXXIT EQU     *
C2C4 5F             CLRB            indicate no extent element
          C2C5      GXRETN EQU    *
C2C5 35 12          PULS  A,X       restore regs


                    *
                    * Name      - COPYIT
                    * Function  - This rtn is called to copy a TRS file
                    *             to a FLEX file. The user is prompted
                    *             for a "from" file (off the TRS disk)
                    *             and a "to" file (in FLEX). If the "to"
                    *             file is not specified, output goes to
                    *             the terminal.
                    *
                    *             All registers are used and not restored
                    *
          C2C8      COPYIT EQU    *
C2C8 32 E9 FFEC     LEAS  -LS3,S    alloc local storage

C2CC 8E 020B        LDX   #INDRV    get "from" drive
C2CF BD CD1E        JSR   PSTRNG
C2D2 8D 71          BSR   GETDRV    get drive#
C2D4 25 67          BCS   DP2       if bad then leave

C2D6 8E 01F4        LDX   #INFILE   prompt for "from" filename
C2D9 BD CD1E        JSR   PSTRNG
C2DC BD CD1B        JSR   INBUF     get "from" filename
C2DF 30 E9 0000     LEAX  SVFFIL,S  where to put filename
C2E3 8D 00          BSR   GTSPEC    get TRS filename
*** RELATIVE BRANCH TOO LONG
C2E5 25 56          BCS   DP2       if bad then leave

C2E7 17 00C0        LBSR  SRCHM     go do search
C2EA 24 08          BCC   GET0      if found, get "to" file
                    *
                    * File not found in TRS directory -> flag
                    * and reprompt user.
                    *
C2EC 8E 026C        LDX   #BADFIL   print msg
C2EF BD CD1E        JSR   PSTRNG
C2F2 20 49          BRA   DP2       return

                    *
          C2F4      GET0  EQU     *
                    *
                    * The TRS directory entry has been found.
                    *

C2F4 B6 0518        LDA   DRV       get drive #
C2F7 B7 C043        STA   SYSFCB+FCBDR save in FCB
C2FA 17 0217        LBSR  ROUTE     set route
C2FD 7D 0519        TST   OPNFLG    writing to a file?
C300 26 03          BNE   DP1       yes, then no lead CRLF

C302 BD CD24        JSR   PCRLF     print CRLF first
          C305      DP1   EQU     *
```

```
                    *
                    * Byte FDEOFM specifies what sector from the
                    * beginning of the file is the end of file,
                    * and byte FDEOFL specifies the offset within
                    * that EOF sector that the copy should continue
                    * up to. Get these values first and save
                    * them in EOFSEC and EOFOFF respectively so
                    * that the PRTBLK routine can transfer the
                    * proper bytes
                    *
C305 A6  88 14      LDA   FDEOFM,X  get EOF sector
C308 B7  C08D       STA   EOFSEC    and save
C30B A6  03         LDA   FDEOFL,X  get offset
C30D B7  C08E       STA   EOFOFF    and save

C310 30  88 16      LEAX  FDEXEL,X  pt to extent element list
C313 31  E9 000C    LEAY  TMPXEL,S  transfer to temp area
C317 CC  0008       LDD   #$0008    len to move
C31A 17  01C3       LBSR  MVC       move bytes
C31D 30  E9 000C    LEAX  TMPXEL,S  pt X at temp area

         C321 DP4   EQU   *
C321 17  FF7F       LBSR  GETXEP    find start of area and len
C324 5D             TSTB            gotten last granule?
C325 27  16         BEQ   DP2       branch if so
C327 E7  E9 0008    STB   TFSIZ,S   save area size
C32B 1F  20         TFR   Y,D       and save start of area to D
         C32D DP3   EQU   *
C32D 17  0161       LBSR  GETGRN    get a granule of data
C330 17  0000       LBSR  PRTBLK    and print it

C333 6A  E9 0008    DEC   TFSIZ,S   transferred all grans in this area?
C337 26  F4         BNE   DP3       continue if not
C339 30  02         LEAX  2,X       goto next extent pair
C33B 20  E4         BRA   DP4       and continue transfer

         C33D DP2   EQU   *
C33D 17  03AA       LBSR  RESET     clear things up first

C340 32  E9 0014    LEAS  LS3,S     release local storage
C344 39             RTS             return
                    *
                    * Data area for COPYIT routine
                    *
         C345 LPC   SET   *
0000                ORG   $0000
         0000 LCL   SET   *
0000     SVEFIL     RMB   11
0008     TFSIZ      RMB   1
000C     TMPXEL     RMB   8
         0014 LS3   EQU   *-LCL     len of local storage

C345                ORG   LPC       restore PC

                    *
                    * Name     - GETDRV
                    * Function - This routine gets a drive # from the
                    *            terminal. The user can either enter a
                    *            drive # in the range 0-3, or type null,
                    *            in which case the default drive# in
                    *            "DRV" will be taken.
                    *
                    *            The carry is clear if # ok, else it is
                    *            set. The valid drive number is returned
                    *            in ACC A, and is also saved in DRV.
                    *
                    *            All registers are preserved.
                    *
         C345 GETDRV EQU   *
C345 34  34         PSHS  B,X,Y     save regs

C347 BD  CD09       JSR   INCH      get response
C34A 81  0D         CMPA  #CR       use default?
C34C 26  05         BNE   GET1      branch if not

C34E B6  0518       LDA   DRV       get default
C351 20  0E         BRA   GET4      and save it

         C353 GET1  EQU   *
C353 BD  CD21       JSR   CLASS     classify it
C356 25  0D         BCS   GET2      bad drive

C358 81  33         CMPA  #'3       in range 0-3?
C35A 22  09         BHI   GET2      bad drive

C35C 84  0F         ANDA  #%00001111 keep low 4 bits
C35E B7  0518       STA   DRV       save in DRV storage
         C361 GET4  EQU   *
C361 1C  FE         CLC             set good return
C363 20  02         BRA   GET3      and return

         C365 GET2  EQU   *
C365 1A  01         SEC             set bad return
         C367 GET3  EQU   *
C367 35  34         PULS  B,X,Y     restore regs
C369 39             RTS             and return

                    *
                    * Name     - GTSPEC
                    * Function - This routine parses the system INBUF
                    *            buffer for a TRS file name. A valid
                    *            filename must contain up to 8 chars,
                    *            then an optional '.' and up to 3 char
                    *            file extension. The X reg points to
                    *            an area to place the 11 char file name
                    *            and any unused chars are padded with
                    *            spaces.
                    *
                    *            Carry is clear if file spec ok, else
                    *            it is set.
                    *
                    *            All regs are preserved.
                    *
         C36A GTSPBC EQU   *
C36A 34  36         PSHS  A,B,X,Y   save regs

                    *
                    * First blank out the "to" field
                    *
C36C 86  20         LDA   #SP       pad with spaces
C36E A7  84         STA   0,X
C370 31  01         LEAY  1,X       "to" field
C372 CC  000A       LDD   #11-1     length to move
C375 17  0168       LBSR  MVC       move in spaces

C378 108E C080      LDY   #LINEBUF  pt to system buffer
C37C B6  08         LDA   #11       # chars to move

C37E E6  A4         LDB   0,Y       user just hit return?
C380 C1  00         CMPB  #CR
C382 27  21         BEQ   GTBAD     yes, then bad return
         C384 GT1   EQU   *
C384 E6  A0         LDB   0,Y+      get a char

C386 C1  20         CMPB  #SP       space here?
C388 27  FA         BEQ   GT1       skip spaces

C38A C1  2E         CMPB  #'.       have an extension
C38C 26  0A         BNE   GT2       nope, then continue
                    *
                    * The rest of the filename from the system
                    * buffer is the extension. Adjust the
                    * "to" ptr so that any unused chars in the
                    * "to" name are spaces
                    *
C38E 1F  89         TFR   A,B       copy A
C390 C0  03         SUBB  #3
C392 2B  11         BMI   GTBAD     if neg then bad name
C394 30  85         LEAX  B,X       adjust "to" ptr
C396 20  EC         BRA   GT1       and continue

         C398 GT2   EQU   *
C398 C1  00         CMPB  #CR       end yet?
C39A 27  05         BEQ   GT3       done if so

C39C E7  80         STB   0,X+      save to "to" field
C39E 4A             DECA            moved all chars?
C39F 26  E3         BNE   GT1       branch if not

         C3A1 GT3   EQU   *
C3A1 1C  FE         CLC             good return
```

```
C3A3 20 02            BRA   GTGOOD
         C3A5 GTBAD   EQU   *
C3A5 1A 01            SEC             set bad return
C3A7 35 36    GTGOOD  PULS  A,B,X,Y   restore regs
C3A9 39               RTS             and return


                      *
                      * Name     - SRCHM
                      * Function - This routine searches the directory on
                      *            a TRS disk for the file name pointed to
                      *            by the X register. "SYSFCB" is the FCB
                      *            used, and is assumed to contain the
                      *            drive where the TRS disk lies
                      *
                      *            On exit, X->TRS file entry of found
                      *            file. If file is not found, the carry
                      *            is set, otherwise it is clear
                      *
                      *            All registers are preserved
                      *
         C3AA SRCHM   EQU   *
C3AA 34 26            PSHS  A,B,Y     save regs
C3AC 32 E9 FFFB       LEAS  -LSS,S    alloc local storage

C3B0 17 FE7A          LBSR  FINDIR    find begin of directory
C3B3 86 02            LDA   #DIRSIZ   size of dir (granules)
C3B5 A7 E9 0002       STA   TDSIZ,S   save it
C3B9 FC 0516          LDD   DIRBEG    get begin of dir

         C3BC SRC1    EQU   *
C3BC 17 0002          LBSR  GETGRN    get a granule of dir
C3BF ED E9 0003       STD   TNITS,S   save next trk/sec to read
C3C3 108E 051B        LDY   #BUFFER   start a buf beginning
C3C7 A6 E9 0002       LDA   TDSIZ,S   at 1st gran of dir?
C3CB 81 02            CMPA  #DIRSIZ
C3CD 26 04            BNE   SRC2      branch if not

C3CF 31 A9 0200       LEAY  SSIZ32,Y  skip over GAT and HIT
         C3D3 SRC2    EQU   *
C3D3 10AF E9 0000     STY   TMP4,S    save entry addr
C3D8 A6 A4            LDA   FDEF1F,Y  get free/inuse flag
C3DA 85 10            BITA  #FIFREE   valid file entry?
C3DC 27 09            BEQ   SRC4      skip it if not

C3DE 31 25            LEAY  FDEFN,Y   point to file name
C3E0 86 08            LDA   #11       8 bytes to compare
C3E2 17 0111          LBSR  CLC       compare names
C3E5 27 1C            BEQ   FOUND     if match then leave
         C3E7 SRC4    EQU   *

C3E7 10AE E9 0000     LDY   TMP4,S    set back entry addr
C3EC 31 A8 20         LEAY  FDELEN,Y  goto next FDE entry
C3EF 108C 0A1B        CMPY  #BUFFER+BSIZ5 finished?
C3F3 25 DE            BLO   SRC2      branch if not

C3F5 EC E9 0003       LDD   TNITS,S   get nxt trk/sec to read
C3F9 6A E9 0002       DEC   TDSIZ,S   gone through all dir grans?
C3FD 26 B0            BNE   SRC1      branch if not

         C3FF SRCIIT  EQU   *
C3FF 1A 01            SEC             file not found
C401 20 06            BRA   RET       return

         C403 FOUND   EQU   *
C403 AE E9 0000       LDX   TMP4,S    get back entry addr
C407 1C FE            CLC             file found
         C409 RET     EQU   *

C409 32 E9 0005       LEAS  LSS,S     release local storage
C40D 35 26            PULS  A,B,Y     restore regs
C40F 39               RTS             and return
                      *
                      * Data area for SRCHM routine
                      *
         C410 LPC     SET   *
0000                  ORG   $0000
         0000 LCL     SET   *
0000               TMP4  RMB   2
0002               TDSIZ RMB   1
0003               TNITS RMB   2
         0005 LSS     EQU   *-LCL     len of local storage

C41D                  ORG   LPC       restore PC
                      *
                      * Name     - PRTBLK
                      * Function - This routine dumps the contents of
                      *            BUFFER to the standard output.
                      *            The EOFSEC and EOFOFF bytes are used
                      *            to determine when we have reached the
                      *            end of file sector. Only bytes up to
                      *            and including the EOFSEC and EOFOFF
                      *            are transferred to the standard output
                      *
                      *            All registers are preserved
                      *
                      *
         C410 PRTBLK  EQU   *
C410 34 36            PSHS  A,B,X,Y   save regs

C412 B6 C48D          LDA   EOFSEC    get EOF sector
C415 81 05            CMPA  #GRANUL   is it in this granule?
C417 22 17            BHI   PB6       branch if not
                      *
                      * calc EOF addr within buffer
                      *
C419 8E 051B          LDX   #BUFFER   pt to begin of buffer
         C41C PB2     EQU   *
C41C 4A               DECA            incr buf ptr far enough?
C41D 27 06            BEQ   PB7       branch if yes
C41F 30 89 0100       LEAX  $SIZ5,X   incr buff ptr
C423 20 F7            BRA   PB2       and continue

         C425 PB7     EQU   *
C425 F6 C48E          LDB   EOFOFF    get offset within sector
C428 4F               CLRA
C429 30 8B            LEAX  D,X       and incr buf ptr
C42B BF C48F          STX   EOBUF     save this address
C42E 20 08            BRA   PB8

         C430 PB6     EQU   *
C430 80 05            SUBA  #GRANUL   reduce EOF sec value
C432 B7 C48D          STA   EOFSEC    and save it
C435 8E DA1B          LDX   #BUFFER+BSIZ5 calc end of buf addr
C438 BF C48F          STX   EOBUF     and save

         C43B PB8     EQU   *
                      *
                      * EOBUF has end of buffer address
                      *
C43B 8E 051B          LDX   #BUFFER   point to buffer area
C43E 5F               CLRB            start column counter at 0

         C43F PB1     EQU   *
C43F BC C48F          CMPX  EOBUF     end of buffer yet?
C442 27 46            BEQ   PB3       branch if so
C444 A6 80            LDA   0,X+      get a char

C446 34 04            PSHS  B         save col ctr
C448 F6 0519          LDB   OPNFLG    get open flag bits
C44B C1 A0            CMPB  #OPN+BINARY file open and binary?
C44D 35 04            PULS  B         restore col ctr
C44F 27 33            BEQ   PB4       branch if so

C451 81 0A            CMPA  #LF       line feed?
C453 27 EA            BEQ   PB1       skip if so

C455 81 0D            CMPA  #CR       a carriage return?
C457 26 08            BNE   PB9       branch if not

C459 5F               CLRB            clear column counter
C45A 7D 0519          TST   OPNFLG    writing to a file?
C45D 26 25            BNE   PB4       branch if so
C45F BD C024          JSR   PCRLF     print a CRLF
C462 20 DB            BRA   PB1
         C464 PB9     EQU   *
C464 81 09            CMPA  #TABCHR   a tab char?
C466 26 1C            BNE   PB4       nop, then output char as is
```

TO BE CONTINUED

BIT BUCKET

[illegible]
[illegible]
[illegible]

CONTACT:
Bob Gildenberg
(212) 420-0317
Peter Glenkoff
(212) 867-0818

IMMEDIATE RELEASE

Philon, Inc. Unveils PHILON FAST/HELP:
Innovative Customer Support System
Dramatically Increases Responsiveness to Users

NEW YORK, May 7 -- Citing an industry-wide need for highly responsive customer support, Philon, Inc. today announced the implementation of a proprietary, interactive data information customer support system. The system -- PHILON FAST/HELP -- complements Philon's recent introduction of fast-executing mainframe quality language compilers for the MC68000/UNIX operating environments. "We intend to provide a level of customer support not usually found in the microprocessor environment," said Bob Gildenberg, Philon vice president of marketing.

The PHILON FAST/HELP system is designed to capture all pertinent information concerning a question or problem. Each inquiry is immediately prioritized and tagged with a target resolution date. Call-backs from customers to provide additional information or request a status update on the question are routed by the PHILON FAST/HELP system to the appropriate Philon customer support representative. Solutions to previous questions can be easily referenced by the customer support representative to assist in answering future questions from customers.

The PHILON FAST/HELP system generates a series of daily and monthly reports which analyse the status of all customer support work. An automatic daily reminder feature insures quick transmittal of solutions to the customer. Monthly reports are used by Philon to analyze the responsiveness and performance of the PHILON FAST/HELP system and the various Philon groups that support it. Monthly status reports are prepared to summarize all known outstanding questions for each hardware/operating system environment supported by the PHILON FAST/Compilers. Workarounds, fixes, new documentation, and information on future release dates are presented. A copy of this report is also sent to each Philon customer who has a maintenance agreement.

According to Michael Parrella, Philon president, "PHILON FAST/Compilers are of particular interest to professional software developers who require the fast execution, excellent documentation and program development tools that Philon provides. They also demand the type of highly responsive support the PHILON FAST/HELP system has been designed to give. This results in faster turnaround for questions and problems, increased responsiveness, and accurate up-to-date information for our customers and ourselves."

Formed in 1980, Philon, Inc. has developed programming software to serve the computer industry's need for a new generation of fast-executing, portable language compilers. Headquartered in New York, Philon is under contract from a Fortune 100 corporation to develop compilers for the U.S. Government.

```
00001        ********************************************
00002        * LOWRES - A SUBROUTINE FOR BASIC09        *
00003        * BY RONALD D. VOIGTS                      *
00004        * CALLED BY: RUN LOWRES(LOC,VAR)           *
00005        * LOC IS AN INTEGER FROM 0 TO 511          *
00006        * VARIABLE CAN BE ARITHMETIC BUT NOT REAL  *
00007        * BEST IF A BYTE OR BYTE-ARRAY             *
00008        * OR BLANK-NOT CALLS LOC POINTS TO         *
00009        * NEXT SCREEN LOCATION IF NOT CHANGED      *
00010        * SUBROUTINE IS TERMINATED WHEN THE        *
00011        * VARIABLE IS PRINTED OUT ON SCREEN        *
00012        * END IS ATTEMPTED. LOC LARGER THAN        *
00013        * 511 ARE ASSUMED TO LOCATION 0.           *
00014        ********************************************
00015
00016                              NAM   LOWRES
00017                              TTL   LOW RESOLUTION GRAPHICS GENERATOR
00018
00019        PUSH /DD/DEFS/LOWDEFS
00020                              IFP1
00021                              INDC
00022
00024  0600  VIDEO                EQU   $0600      SCREEN LOCATION
00025
00026  0000 [illegible]           MOD   LENGTH,LOWRES,$11<8+$E,REENT+1,ENTRY,0
00027  0000 4C4F5752  LOWRES       FCB   /LOWRES/
```

```
00078
00079 D 0000                     ORG   0
00080 D 0000           RETURN    RMB   2
00081 D 0002           COUNT     RMB   2
00082 D 0004           PARM1     RMB   2
00083 D 0006           SIZE1     RMB   2
00084 D 0008           PARM2     RMB   2
00085 D 000A           SIZE2     RMB   2
00086
00087  0013 8C02       ENTRY     LDD   COUNT,S    TWO PARAMETERS?
00088  0015 [illegible]           CMPD  #2
00089  0019 263C                  BNE   ERROR      NO, THEN ERROR
00090  001B EC66                  LDD   SIZE1,S    IS FIRST PARM INTEGER?
00091  001D [illegible]           CMPD  #2
00092  0021 3634                  BNE   ERROR      NO, THEN ERROR
00093
00094  0023 [illegible] CHKLOC    LDY   [PARM2,S]  CHECK SCREEN LOCATION
00095  0027 [illegible]           CMPY  #$200      IS IT BEYOND END?
00096  002B 2504                  BLO   GETVAR     NO THEN GO ON
00097  002D [illegible]           LDY   #0         OTHERWISE SET TO BEGINNING
00098
00099  0031 ED6A       GETVAR     LDB   $02,S      GET VARIABLE SIZE
00100  0033 31A90600              LEAY  VIDEO,Y    SET Y TO OUTPUT
00101  0037 [illegible]           LDX   [PARM1,S]  SET X TO VARIABLE
00102
00103  003A A680       OUTPUT     LDA   ,X+        GET A BYTE
00104  003C A7A0                  STA   ,Y+        PUT IT ON SCREEN
00105  003E [illegible]           CMPY  #VIDEO+$200 ARE WE AT END OF SCREEN
00106  0042 2409                  BHS   FINISH     YES, THEN FINISH
00107  0044 335F                  LEAU  -1,U       DECREMENT COUNTER
00108  0046 [illegible]           CMPU  #0         IS ANY VARIABLE LEFT?
00109  004A 26EE                  BNE   OUTPUT     YES, DO IT AGAIN?
00110
00111  004C 1CFE       FINISH     ANDCC #%11111110 CLEAR CARRY BIT
```

```
Microware OS-9 Assembler RS Version 01.00.00   04/01/84 16:54:17   Page 002
LOWRES - LOW RESOLUTION GRAPHICS GENERATOR

00062  0042 31A9FFB0              LEAY  -VIDEO,Y   SAVE SCREEN LOCATION
00063  0052 [illegible]           STY   [PARM1,S]
00064  0056 39                    RTS               RETURN TO BASIC
00065
00066  0057 C630       ERROR      LDB   #$6        A PARAMETER ERROR
00067  0059 1A01                  ORCC  #[illegible]1 SET CARRY BIT
00068  005B 39                    RTS               RETURN TO BASIC
00069
00070  005C 7C1266                END
00071  005F           LABEND      EQU   *
00072                              END

00000 error(s)
00000 warning(s)
$005F 00095 program bytes generated
$0CBC 00812 data bytes allocated
$1CCC 07372 bytes used for symbols
```


abies
INFORMATICS LTD
10 Hanley Mews Piccadilly London W1 3PH Telephone 01 [illegible] Telex [illegible]

PRESS RELEASE

BBC Micro System for GPs

The ABIES GP system is now available on the BBC microcomputer using a Motorola 6809 second processor for less than £1000 including VAT.

General Practice is an important vertical market and over 30 different systems have been developed for GPs. However the take-up has been very slow. Only some 300 out of 12,000 practices are yet using computers in their surgeries. The main deterrent has been the entry cost (between £7,500 and £12,500 for the best-known systems).

The ABIES BBC-GP system costs £990 (including VAT) for software, second processor and 64 Kbytes of RAM. The total investment including BBC Model B microcomputer, 800 Kb disk drives, monitor and printer is under £2500 (including VAT).

Using a computer allows the GP to offer more efficient preventive medicine and repeat prescribing services. In general, GPs with

computers do more screening and preventive measures and prescribe relatively less. The principal beneficiary is the patient. The introduction of low cost systems such as the ABIES BBC-GP should spread these benefits far more widely.

ABIES GP software has been used on larger computers for over three years in practices in all parts of England. It is a fully developed and well-proven package providing comprehensive Age/Sex Register, Recall and Repeat Prescribing facilities. The data is held in an extremely compact way, allowing up to 3000 patient records to be stored on a single 400 Kbyte floppy disk.

The BBC version has been made possible by collaboration between ABIES and Cambridge Microprocessor Systems (CMS). It uses the CMS 6809 second processor which includes 64K RAM and is connected to the "tube" of the BBC micro. CMS developed special screen handling software to enable the ABIES system to run unchanged on the BBC micro. The ABIES BBC-GP package is the first application package to be marketed using a 6809 second processor on the BBC microcomputer.

For further details contact:

Tim Benson
ABIES Informatics Ltd
10 Barley Mow Passage
London W4 4PH
01-994 6477

or

Phil Taylor
Cambridge Microprocessor Systems Ltd
44a Hobson Street
Cambridge CB1 1NL
0223-324141

Francis MASSEN
8 Cite Strauss
L-9351 BETTENDORF/ Luxembourg, Europe
te : 808021

CONNECTING A H14 HEATHKIT PRINTER TO A MP-S2 SERIAL INTERFACE

Most of you will probably smile if they read me talking about old H14; at a time where printer is almost synonymous to EPSON there certainly whill not be very many H14 Heathkit rattle-machines left in operation.

We in our school are running such a printer for many years, driving it through a MP-S2 serial interface from a SWTPc 6809 computer at a leisurely baud-rate of 300; at that speed, no handshaking is needed. Going to a higher baud-rate does not give a tremendous improvement, but nevertheless the increase in speed is something like 30% for 9600 baud. Alas, the H14 uses a rather non-standard handshaking convention (but what in the RS-232 world IS standard?): when the printer is busy, the RTS line (pin 4 on the cannon connector) goes high (to a positive voltage). On the other side, the MP-S2 interface senses the pin 20 who is tied, via a 1489 RS-converter, to the CTS (negative logic) input of the ACIA. Pin 20 on the MP-S2 board is a true positive logic CTS input: the ACIA transmits if this pin 20 is at a positive voltage, just the opposite of what the H14 printer is expecting!

The remedial is quite simple: the CTS line on the MP-S2 board has to be inverted before entering the ACIA; this inversion is best done after the signal has been brought to TTL level, that is after the 1489 has done its job. Fortunately, the MP-S2 contains a 7402 NOR chip with one spare gate. So here are the steps to do:

1. Cut on the MP-S2 board the trace coming from pin 13 of the 1489, before it is going to the pin 24 of the ACIA.

2. Remove, on the bottom side of the board, pins 12 and 11 of the 7402 chip from ground and tie both of them to the trace coming from the 1489.

3. Connect the output pin 13 of the 7402 chip to the trace going to pin 24 of the ACIA.

Now you may run the H14 printer at the top speed of 9600 baud. Ok, the manual warns you from doing that; I preferred to ignore that advice, and our printer is smoothly running for over half a year at 9600 baud, without ever loosing a single caracter.

Here as a reminder the lines to be used:

- ground (pin 7 on the cannon-connector)
- transmit of MP-S2 tied to receive of H14
- CTS (pin 20) of MP-S2 tied to RTS (pin 4) of the printer.

Happy printing !

METALAB Z809

PHILIP C. NUNN, Consultant
Engineering & Research Management

301 Netherfield, N.W.
Comstock Park, Michigan 49321
Telephone (616) 784-5732

CP/M RESURRECTED

During March of this year, I had the opportunity to beta test the new version 2.0 of CP/M from Metalab. My overall impression is that it is a great improvement over the original version 1. I really liked its operating characteristics. Many readers may remember my criticism of MetaLab's version 1.

I field tested the METACPM2 on my usual work. I have a data processing system for project management in engineering and research firms. For system software I use Wordstar, dBASE II, Pascal MT+, and FORTRAN-80. The number of programs involved is about 200, and overlays are common. Data files for this system usually run at least 100k.

METACPM2 has many good features which make using it much more enjoyable than its predecessor. Perhaps its most outstanding feature is its flexability and adaptability. Source code for its BIOS (I/O module) is available and well documented. It reads and writes three disk formats: single-side single density for computer-to-computer transfer of data and programs, single-side double-density for users with single-head disk drives, and double-side double-density for those of us running double-head drives. The BIOS also has the ability to interface with a hard disk controller. This new and much improved BIOS takes no more memory space than its less powerful predecessor, the result of some skillful operating system programming.

METACPM2 comes with several useful utilities, some of which are accompanied by their source code in C. Outstanding are those written by Metalab which modify or work with the BIOS. For example, the CONFIG utility makes soft modification of I/O porting to printer or console. And, get this, it allows selection of disk drive stepping rate. I'm running Qume Datatrak-8's which have a 3ms stepping rate. Until METACPM2, I had only been able to cuss everytime I heard those precision drives rattling along at 15ms. Thank you Metalab.

The FORMAT utility for formatting contains some strokes of sheer genius. First of all, it goes to the BIOS in the operating system to get the parameters

for the disk format selected. This means that there is no chance for a format incompatibility even if we patch-in a different disk format. I, in fact, did this. I changed the double-side double-density disk format from 256 bytes/sector to 512 bytes/sector to increase my disk capacity. FORMAT tracked this change with no modification. In addition, FORMAT allows for soft modification of some parameters for all formats.

In addition to the system utilities written by Metalab, the disk is filled-out with public domain CP/M utilities such as D.COM, DU-77.COM, SWEEP.COM, TELL.COM, UNERA.COM and UNSPOOL.COM.

As my headline above suggests, my opinion is that Metalab has resurected CP/M for the 50-bus with this new version. Because of the adaptable hardware interfacing capabilities written into it and the staggering wealth of software it provides access to, I think METACPM2 stands above both OS-9 and FLEX for single user applications. It is without question a competitive operating system on the 50-bus.

Metalab's development comes at a time when even this magazine has expressed doubts about the future of the 50-bus because of the small amount of application software being developed and lack of 16-bit processors. It now provides an alternative to "junking-out" our expensive and well built computers because of the "end-of-the-road" syndrome. Regardless of all the hoopla, I think that there are many applications for 8-bit computers for up to 10 more years. I will even go so far as to predict that, hoopla aside, 8-bit logic is all that is necessary for single user applications. Multi-user applications and the 50-bus are another problem, entirely. Let's try to keep them in their proper perspective.

At the time I am writing this, Metalab has not advertised its new operating system. If by the time you read this review, they still have not advertised it, write to them for detail information at:

Metalab/Autonomics
6825 County Line Road
Longmont, CO. 80501

This is too good an opportunity to let pass for those of us who suffered through the original version of the Metalab operating system. Encourage them to put it into the 50-bus marketplace.

Philip C. Nunn

Philip C. Nunn

ELEKTRA

A Review By Bob Sims

ELEKTRA (tm) MOTHERBOARD

The Elektra(tm) motherboard is an extremely versatile motherboard designed to support the SS-50 and SS-50C buses. It offers a great deal of flexibility in configuring a standard (SWTPCo (tm), GIMIX, SSB, etc.) or your own "special" SS-50/SS-50C based system. For the person looking to upgrade from a 6800 SS-50 system to a 6809 SS-50C system, or for the person who needs more 50 pin slots for putting together a really big system, the Elektra (tm) motherboard is an excellent choice.

The motherboard is available as a bare board or as either a kit or assembled and tested. The assembled version is recommended for all but the experienced kit builder, due to a few minor difficulties encountered while assembling the motherboard. We will examine these difficulties later on, but first, let's take a look at the features available with this motherboard:

* 50 PIN SLOTS (SS-50/SS-50C)

There are eleven 50 pin slots available so that you can put together a rather large system. The first slot has SS-50C designations and the fifth and ninth slots have SS-50 designations to help you in configuring the motherboard for 6800 or 6809 CPU cards. There are provisions for filter capacitors on the power supply lines to prevent possible oscillations with the voltage regulators when the power supply is located more than a few inches away from the motherboard. Optional terminating resistor packs can be installed on the front edge of the motherboard, as some people feel that they improve noise immunity. The first four or eight slots can be cut off to reduce the length of the motherboard to either fourteen or ten inches, in case you want to shorten the motherboard, but in so doing, you remove the optional terminating resistor packs. The design of the SS-50/SS-50C bus provides for little or no noise, so loss of the terminating resistor packs may not be a hardship at all. For those systems which will be running Uniflex (tm) (once the user configurable version becomes available), which is very sensitive to noise (I'm told), the

loss of this termination might very well prove to be fatal.

* 30 PIN SLOTS (SS-30/SS-30C)

There are eight 30 pin slots with standard one inch spacing. Again, there are SS-30 and SS-30C designations for configuration. The I/O section is uniquely decoded for either four, eight, or sixteen addresses per slot and occupy either 32, 64 or 128 addresses without overlap up through the 4K block where it is assigned. Headers and Berg minijump plugs can be used to assign addresses or jumpers can be used if address assignments are not going to be changed. For systems using four addresses per I/O slot and using SWTPCo(tm) DC type disk controllers, the select line of slot five can be connected to slot six without using a user-defined line. There is an on-board optional baud rate generator with selection of any one of eight standard or two user selected baud rates, which can be assigned independently to each of the five baud rate lines on the I/O section by header or jumper selection. There is also high/low baud rate selection which multiplies all available baud rates by four. Baud rates available range from 75 baud to 34,800 baud. Also included on the I/O section is optional extended address decoding, which will allow you to move the I/O section between various 64K memory blocks. This feature would be very useful in multi-user enviorments. The extended addressing is also header or jumper selectable. An optional I/O slowdown circuit is also available allowing use of 1 mhz I/O cards with a processor card that has a clock speed greater than 1 mhz for systems that need this capability. One interesting header or jumper selectable option is an I/O disable feature using either user-defined line 1 or 2. It is interesting because there are no instructions for using this option, except a brief statement that reads "Do not install jumpers at J1 and J2". This option appears to have possible use in multi-user applications where a systems operator might want to deny users access to the system for whatever reason.

* CONSTRUCTION

The motherboard is double-sided and plated through with shielding to eliminate noise. It is also a BRUTE! Being 1/8 inches thick, it is definitely designed to hold up to a lot of wear and tear over a long period of time. There is a noticable lack of bending and flexing due to the thickness, which is a welcome change from some of the weaker motherboards one might have encountered in older SS-50 motherboard designs. The mounting holes are spaced so that the motherboard is firmly supported on all sides and consequently offers excellent rigidity when inserting and removing cards on both the 50 pin and 30 pin sections. The motherboard comes with square pin molex connectors instead of the round pin type, thus ensuring solid contact of all cards. Gold plated connectors are available as an option. As was mentioned earlier in this review, a few minor difficulties were encountered during the process of assembling the motherboard from a bare card. I say

minor difficulties because I am an experienced kit/bare board builder, and for myself, they were minor. To the inexperienced builder, the difficulties could very well be major. This is why I recommend the assembled and tested motherboard. Besides, the price differential between the kit and the assembled motherboard is relatively small, especially when the amount and precision of soldering required is taken into account. Some of the soldering is really tight. Most of the difficulties encountered were caused by the thickness of the motherboard and the size of many of the solder pads. IC or socket leads will simply not go all the way through a 1/8 inch thick board, so you have to be sure that solder flows around the pins to make a good connection within the plated-through holes. Also, the spacing of the header pins are rather close, and the size of most of the

solder pads could definitely be larger. I would have opted for dip switches instead of headers or jumpers, along with larger solder pads if I had to design this motherboard myself, but nevertheless, the headers or jumpers do the job. In all fairness, the documentation does advise the builder that a soldering iron with a fine tip should be used and addresses both the possible thickness and spacing difficulties with the recommended procedures to ensure proper construction. In my case, I was eager to get the motherboard up and running, and because of my indiscretion, I had to go back and resolder in a few places. Another area that poses possible problems for the inexperienced builder is in component placement. The motherboard does not have any markings or numbering for parts placement. The documentation does have a parts placement diagram accompanying it, but silkscreening parts placement right on the motherboard would be preferred, especially for a novice builder. The final problem I encountered was with the high/low baud rate selection header. Because of the closeness of the header to I/O slot #0, you cannot use a header and plug. Any card plugged into this slot will not seat properly because the programing plug stops the card from being seated all the way down. A wire jumper will have to be used instead. As I mentioned previously, the difficulties were minor, and experienced builders should not have any real problems.

* DOCUMENTATION

Accompanying the motherboard were ten pages of documentation consisting of two pages of assembly instructions, five pages of baud rate selection, I/O addressing, and jumper selection for configuration, one page showing SS-50/SS-50C and SS-30/SS-30C designations, a parts list, a parts placement diagram, and a block circuit diagram. Not what you would call a heavy manual, but it contains all necessary information you need to assemble and configure the motherboard for use. I know I am probably spoiled from having a theory of operation section from most manufacturers, but I think any product sold in a kit form should have a section covering theory of operation so that in case of problems, the builder can at least try to see where they might have made a mistake or where a component might be mis-placed or defective. Even though the motherboard was primarily designed for OEM applications where a theory of operation section is rarely needed, I would prefer to see one included in the case of kits or bare-boards, as OEM's almost never buy kits or bare-boards for their applications. They usually prefer to buy assembled modules or sub-sections and then configure them to suit their individual needs. Anyway, if you purchase the assembled motherboard, all you will need is the configuration section, along with the parts list and diagrams.

* SUMMARY

The Elektra(tm) motherboard is sturdy and reliable and would be an excellent choice for building a SS-50/SS-50C system. It has all the features that one would need in putting together a large system. The motherboard operates as designed and is very cost effective when compared to other motherboards. To date I have not experienced any problems with the operation of the motherboard and would not hesitate to recommend it to anyone for incorporation into their system. The Elektra(tm) motherboard is available from:

AAA Chicago Computer Center
120 Chestnut Lane
Wheeling, IL 60090
(312) 459-0450

* KUDOS

A word about AAA Chicago Computer Center and it's owner, Jerry Koppel. This company is a good example of a small company that started out as a part-time business that sold other companies products and little by little built itself up to the point that they now have introduced their

own product line. Jerry goes out of his way to provide something that seems to be getting rarer these days, customer service. Maybe that's why AAA's still around after many other SS-50 based companies have gone out of business. I guess there must be something to this customer service stuff. If the Elektra(tm) motherboard is any indication of the quality of the complete Elektra(tm) product line, than AAA should continue to grow and prosper. I give the Elektra(tm) motherboard a rating of AAA. Consult AAA's ad in this magazine for the current pricing on the Elektra(tm) motherboard and AAA's other fine products

THE NEW PRICE/PERFORMANCE STANDARD FOR RASTER GRAPHICS PROCESSORS

CT-1000

DRAWS SO FAST, MOTION IS CONTINUOUS

... WITH HIGH RESOLUTION AND LARGE OBJECT DATABASE RAM

2D OR 3D DISPLAY TRANSFORMS

GRAPHICS POWER ... YOUR SYSTEM EDGE

3711 S. George Mason Drive
Falls Chuch, VA 22041
(703) 671-3900

Privac Inc.

Dear Dan:

This letter is to point out a minor bug, and to suggest a related improvement in the program FILES published in the MAY 84 issue (p.45).

The bug is early in the program, between lines "FIL1 ..." and "FIL12 ...", and involves the ASRA instructions used to divide the screen width by eleven. While the ASRA (arithmetic right shift) instruction works ok for narrow widths, it fails down when the width is greater than 127 columns (as is frequently the case for printers). A better instruction to use for this purpose is the LSRA which performs basically the same function, except that it zero-fills the sign bit.

The improvement that I propose is related to the same area of the program. I find that it is nice to specify a lot of columns when I direct a listing to my line printer, but I still need to specify just a few for my 64 column screen. While it is not too hard to change the TTYSET "WID" parameter prior to printer listings, I thought that there might be an easier way. The accompanying listing shows a fragment of my version of the FILES program -- that which incorporates an addition. It is added between the lines reading "FIL15 STA NPL" and " JSR PCRLF". As suggested in the comments, this mod lets you specify in the command line how many columns you want it to generate. The syntax is sort of peculiar -- but it was easier than trying to figure out some of the logic involved in his command line interpretation.

When running the program, if you do nothing special, then it works as Terry intended (and figures out the number of columns from the TTYSET parameter). If you want to specify a different number of columns, then terminate the command name ("FILES") with a slash ("/") character, followed by the one digit number of columns you want. No spaces are allowed between command, slash, and number. The number of columns may range from 1 to 9. Some examples of use are as follow:

```
***FILES                default
***FILES/2              all files, 2 columns
***FILES/2 1            all sizes drive 1, 2 col
***FILES/B .Tst .BIN    specific filetypes, B col

I hope that some of your readers will find this useful.

                        Donald N. Norte  WB8IDU
                              2714 Thomas
                            Flint  MI  48504
---
PAGE:    1
listing of modifications to FILES pgm

                * MOD ADDED AT VER 3 BY DNN
                *  ALLOWS OPTIONAL FORCED SPECIFICATION OF
                *     NUMBER OF COLUMNS.
                *  START COMMAND BY:
                *     ***FILES/n <rest_of_arguments>
                *  WHERE n SPECIFIES NUM OF COLUMNS TO PRINT
                *  NO SPACES BETWEEN "FILES" AND THE "/" OR THE "n"
          CC1B  CURCHR  equ    FLEI+$C1B
C130 8A   CC11          lda    LSTTRM     get last terminator
C133 81   0D            cmpa   #$0D       is it a cr?
C135 27   2A            beq    FIL16      yes-no special, no args either
C137 81   CC02          cmpa   EOL        is it eol char?
C13A 27   18            beq    FIL16      yes-no special, no args either
C13C 81   2F            cmpa   #'/        is it option delineator?
C13E 26   17            bne    FIL16      no-process per T.Walters
C140 8D   CD27          jsr    NXTCH      get the num of columns
C143 81   39            cmpa   #'9       #s allowed is nine
C145 1022 0111          lbhi   SERR       too many !
C149 84   0F            anda   #$0F       ok- so asbin it binary
C14B B7   C113          sta    NPL        and store as num
C14E 8D   CD27          jsr    NXTCH      now pass the number
C151 8E   CC14          ldx    #LSTPNT    and update cmd line ptr
C154 8F   C111          stx    POINT
          C157  FIL16   equ    *
                * end of mod...
```


GIMIX INC 1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609 • (312) 927-5510 • TWX 910-221-4056

GIMIX TO INTRODUCE 15 USER "C" DEVELOPMENT SYSTEM
at WCC Booth #B4016

The GMX 6809 CPU 11t performs high-speed DMA transfers from memory to memory and uses memory attributes and illegal instruction trapping to protect the system and users from program crashes. If a user program crashes, only that user is affected; other users are unaware of the problem.

Intelligent Serial I/O Boards significantly reduce system overhead by handling routine I/O functions; freeing the host CPU for running user programs. This improves overall system performance and allows user terminals to be run at up to 19.2K baud.

The system also includes 1 Megabyte of CMOS Static RAM, a 72 Megabyte hard disk drive, a 6 Megabyte Removable pack hard disk drive, and a 96TPI double-sided double-density floppy disk drive.

The "C" Compiler runs under OS-9, a Unix like multi-user, multi-tasking operating system. Also included is an Editor, Assembler, Debugger, BasicO9, RMA, DD, and FLEX.

The total price for this system is $18,868.00. Delivery is from stock to 30 days A.R.O. Export models are available.

For further information contact:

Richard Don at (312) 927-5510

Dear John Click,

To set the record straight, my Epson/Coco interface article was written July 1981. It was published two months later about the time Radio Shack published an updated Color Basic manual with all the details of baud-rate (up to 2400) and wiring hookups.

I now have 2-64K Cocos and one 128K Coco. I presently am President of the local Atlanta Users group, teach a beginners assembly-language class, and head an Assembly-Language group.

I also do various hardware mods such as the following:

- an internal 300-19200 baud serial/parallel printer adapter (frees up RS-232 port)
- a pause button
- RGB driver board
- Video Descrambler (for high speed)
- A/D & D/A parallel interfacing
- 64K conversions
- reversed text
- external keyboards

Gary McConville
4144 Rebel Trails Drive
Douglasville, Ga. 30135
(404) 949-0369

Electronic Specialists, Inc.
171 South Main Street, Natick, Mass. 01760
(617) 655-1532

FOR MORE INFORMATION: FRANK STIFTER

KLEEN LINE CONDITIONERS

ELECTRONIC SPECIALISTS announces their KLR series of portable line conditioners designed to protect sensitive computer equipment. Available for 250, 500, 1000 and 2000 watt loads, the KLR line conditioners deliver 120 volt ± 3% regulation for 90 to 140 volt input variations. With 3% THD sine wave output, the KLR series offers input spike suppression, transformer surge suppression, wide-band pre-filtering and isolated winding line noise elimination.

CORRECTIONS

Dear Sir:

After I reviewed my letter in the «Bit Bucket» of the May, 1985, issue of '68' Micro Journal, I noticed to my chagrin that I had omitted two lines of code from the 6800 version of the lower case to upper case string conversion function. If you use the code as presented, your user function will never terminate. The correct routine follows:

```
        ORG $A100

        LDX $0026
        LDX 2,X
        STX COUNT
        LDX $0026
        LDX 0,X
LOOP    LDAB COUNT+1
        LDAA COUNT
        SUBB #1
        SBCA #0
        BCS OVER
        STAB COUNT+1  omitted from the May, 1985, issue
        STAA COUNT    ditto
        LDAA 0,X
        CMPA #'a
        BHI SKIP
        CMPA #'a
        BCS SKIP
        SUBA #32
SKIP    STAA 0,X
        INX
        BRA LOOP
OVER    RTS
COUNT   RMB 2

        END
```

Somebody once said that the best way to embarass yourself is to put it in writing. I agree.

Very truly yours,

William R. Hamblen

William R. Hamblen

CORRECTION OF THE PARAMMOD PROGRAM.

I refer to my program ParamMod, sent to Don Williams in the summer of 1983. I saw in my latest magazine that he was in hospital, and I really hope that he is well and back home by now.

The ParamMod program was published in your fine magazine, October 1983 issue, pages 13, 14 and 41.

Unfortunately, the program contained a small error that made it nonworking. To correct it, find the source code line in the customization area containing the text:

```
outname fcm .Help_B.
```

Replace it with this text (change only one character):

```
outname fcc .Help_B.
```

Now it will work correctly.

A REAL HELP_B UTILITY THAT USES THE PARAMETER MODIFIER.

To compensate for the error I am sending you a REAL Help_B program. It replaces the dummy Help_B utility published in the same issue of 68 micro journal.

The new Help_B requires a disk file containing the help messages. The file is proposed to be named /d0/helpmes but that can easily be tailored in the Help_B program.

A simple demonstration helpmes file is also included. You can make it complete by adding any messages you need. The seekwords must be lowercase letters only. Note that when the command 'help' is given without any parameter, the missing parameter is replaced with 'help'. Hence, the command 'help' is equivalent to 'help help'.

```
PROCEDURE help_B
0000    REM procedure that prints the content in a file /d0/helpmes
0036    REM from a line containing the parameter word processed by
0093    REM a line starting with a '.' .
0082    REM Printing continues up to the next line starting with a '.'
0091    REM Procedure called with the command dos:COM help_B:"word"
010C    PARAM word:STRING
0113    DIM loop:BOOLEAN
011A    DIM seekword:STRING
0121    DIM char:STRING[1]
012D    DIM ilINTEGER
0134    DIM line:STRING[80]
0140    DIM file:BYTE
0147    OPEN #file,"/d0/helpmes":READ
015D    IF word="" THEN seekword="help"
0173    ELSE seekword=word
017E    ENDIF
0190    REM unshift the letters in seekword
01A2    FOR i=1 TO LEN(seekword)
01B4      char=MID$(seekword,i,1)
01C2      IF char<"_" AND char>"`" THEN
01D7        char=CHR$(ASC(char)-$20)
01E5        seekword=LEFT$(seekword,i-1)+char+RIGHT$(seekword,LEN(seekword
            i-1)
0205      ENDIF
0207    NEXT i
0212    REM end unshift
0220    PRINT
0222    REM search - followed by seekword in file
024A    found=FALSE
0250    WHILE EOF(#file)=FALSE DO
025C      WHILE EOF(#file)=FALSE DO
0268        READ #file,line
0272      EXITIF LEFT$(line,i)="." THEN
0282      ENDEXIT
028A      ENDWHILE
028A      READ #file,line
02A4    EXITIF line=seekword THEN
02A1      found=TRUE
02A7    ENDEXIT
02AB    ENDWHILE
02AF    REM end search
02BC    IF found=FALSE THEN
02C7      PRINT seekword+" not found"
02D9      BYE
02DB    ENDIF
02DD    REM print file content until next .
02FF    WHILE EOF(#file)=FALSE DO
0308      READ #file,line
0315    EXITIF line="." THEN
0322    ENDEXIT
032A      PRINT line
032B    ENDWHILE
032F    PRINT
0331    BYE
```

```
.
help
Explain a certain command

SYNTAX: help <parameter>

help can be given on the following commands:

copy        help        mfree
.
copy
Copy a file into another

SYNTAX: copy <fromfile> <tofile>
.
mfree
Show free memory space

SYNTAX: mfree
.
```

Sincerely,

Bengt Allan Bergvall

Bengt Allan Bergvall
Blavirgeu, 1

HELP

Anyone interested in forming a user's group for the purpose of software and hardware support of the COMPACTA UNIB ARD sold by DIGITAL RESEARCH COMPUTERS is encouraged to contact myself at:
John Cooper
384 Goffstown Rd
Manchester NH 03102
603-627-9908

***

Help Wanted: Information on converting a Wave-Mate Jupiter 2 (6800) to 6809 and Flex 9. Also would like correspondence with any builder of the Digital Systems Uniboard computer.
Jay Kreinik
3053 N San Gabriel Bl #26
Rosemead CA 91770
213-280-6377

***

I would appreciate you including my request for "Help" in your next issue of 68 Micro. I would be interested in being contacted by anyone who has a Micro Works 2708 Eprom Programmer Model B-08 converted to burn single voltage 2716 Eprom's.
B K Anderson
95 Hillside Rd
Kensington CT 06037
203-224-7916

***

Please could someone help me with a good monitor program, Percom or similar, for my homebrew 6800 system. I lack the software expertise to complete the task for myself. Any help gratefully received.
R Swainston
64 Victory Rd
Laingholm Auckland N Zealand

***

CLASSIFIED ADVERTISING

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

***

For Sale: SWTPC 6809 System. Uniflex OS. Includes: 128K Ram; 20 Megabyte Hard Disk; two 8212 terminals, Duel 8" DSDD Drives; Miscellaneous Software, Furniture (Desk, Stand). Less than 1000 hours on time. 1 1/2 years old. Excellent condition. Will consider selling individual pieces. Make me an offer.
Call Marilyn (512)250-9844, Business Data Systems, 8300 Pilgrims Pl., Austin TX 78759

***

68 MICRO JOURNAL PROGRAMS - DISK

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, *Disksave.
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 Crtest, Lanpher (May 82)
Disk-9 Datecopy, Diskfix9 (Aug 82)
Disk-10 Home Accounting (July 82)
Disk-11 Dissembler (June 84)
Disk-12 Modem68 (May 84)
DISK-13 *Initmf68, Testmf68, *Cleanup, *Oskalign, *Leobug, Help
Disk-14 *Init, *Test, *Terminal, *Find, *Diskedit, Help

NOTE:
This is a reader service ONLY! No Warranty is offered or implied. The Disk Files 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 **$29.95** - 5" Disk **$24.95**

68 MICRO JOURNAL
POB 794
Hixson, TN 37343
615-842-4600

* indicates 6800; ** indicates BASIC SWTPC or TSC
6809 has no indicator.

MASTER CARD - VISA accepted
Foreign -- add 10% for surface or 20% for air!!

STWPC 6809, 56K,serial, parallel, clock, DSDD controller, 2 Qume DT-8, H-19, Centronics 306 w/lower case, truck load of software (over $1600)- a steal at $2500, make me an offer. Complete set of Micro Journal from V1.1 to present,offer.
(503)647-5878 Don Kinzer

***

New Verbatim Disks. Never been opened. I bought these from an office supply house that went out of business.

5"
4 Boxes 5" 525-10 SSDD 10 Sector $19.00 Box
4 Boxes 5" 525-16 SSDD 16 Sector $19.00 Box
30 Boxes 5" 550-01 DSDD Soft Sector 24.00 Box

8"
27 Boxes 8" 34-4001 DSDD Soft Sector $27.00 Box
29 Boxes 8" 34-8000 SSDD Soft Sector $24.00 Box
3 Boxes 8" 34-9000 SSSD Soft Sector 23.50 Box
3 Boxes 8" 65-1000 SSSD 32 Sector $29.00 Box

Tom Williams 1-800-338-6800

***

Perkin Elmer 9 track tape drive used only 1 hour for CAD/CAM system. $2400.
Call Jonathan (615)894-0328

***

For Sale: CoCo 2 yrs old, 32K Ext Basic. With color printer and data cassette. Lots of software. $275.00 complete.
Tony Lankford (615)894-7532

***

For Sale: Xebec model S1410 hard disk controller never powered up. $395.00.
Phillip Nunn (616)784-5732

***

Hazelwood Products: mainframe $1400.00, 5/8 DMA controller card $250.00, 6809 c.p.u. $200.00, 256K ram card $600.00. Qume dual 8 inch thin line DS drives with power and case $750.00, SWTPC X-12 terminal w/update $1100.00. Gimix PIO 28 controller $150.00. Gimix 32K static ram card $75.00.
Call Bob 414-271-6755 ext 403 days, 332-2101 eves.

***

COMPILER EVALUATION SERVICES
By: Ron Anderson

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

COMPILER COMPARISION AND EVALUATION REPORT

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

This service, with updates, will allow you who are wary or confused by the various claims of complier 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 WHIMISCAL PL/9

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

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

GOOD NEWS!

C for the 6809 WAS NEVER BETTER!

INTROL-C/6809, Version 1.5

Introl's highly acclaimed 6809 C compilers and cross-compilers are now more powerful than ever!

We've incorporated a totally new 6809 Relocating Assembler, Linker and Loader. Initializer support has been added, leaving only bitfield-type structure members and doubles lacking from a 100% full K&R implementation. The Runtime Library has been expanded and the Library Manager is even more versatile and convenient to use. Best of all, compiled code is just as compact and fast-executing as ever - and even a bit more so! A compatible macro assembler, as well as source for the full Runtime Library, are available as extra-cost options.

Resident compilers are available under **Uniflex, Flex** and **OS9.**

Cross-compilers are available for **PDP-11/UNIX** and **IBM PC/PC DOS** hosts.

Trademarks:
Introl-C, Introl Corporation
Flex and Uniflex, Technical Systems Consultants
OS9, Microware Systems
PDP-11, Digital Equipment Corp.
UNIX, Bell Laboratories
IBM PC, International Business Machines

For further information, please call or write.

INTROL
CORPORATION
647 W. Virginia St.
Milwaukee, WI 53204
(414) 276-2937

OMEGASOFT INDUSTRIAL STRENGTH PASCAL FOR 6809/68000

OMEGASOFT PASCAL

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 more critical that a high level language be used. Find out why over 1000 companies have switched to OmegaSoft Pascal for their demanding applications.

WHY PASCAL?

Pascal was designed to teach students how to write structured programs that are easy to read and maintain. In the past decade it has also proved to provide the same advantages in industrial applications.

EXTENSIONS

OmegaSoft has taken the Pascal framework and expanded 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 byte data type allows you to directly address bytes in memory or I/O devices. The common arithmetic operations can be used for bytes along with shift left, shift right, "and", "or", "eor", and complement operators. These operators are also available for integer and hex (2 byte unsigned) numbers.

Longintegers are four byte signed numbers useful for extended range arithmetic commonly needed for machine control. Functions have been added to allow conversion between the various data types. Dynamic length strings allow complex text manipulation and allow effective interactive I/O.

Variables can be placed either on the data stack (default), at an absolute address in memory (for I/O), in base page, relative to the program counter (for constant tables), or defined in another module.

FEATURES

The compiler generates assembly language for assembly and linking 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 much 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.

Full source code is included for the runtime library, the debugger, and other support utilities.

ISO COMPATIBILITY

OmegaSoft Pascal has been tested using the pascal Validation Suite. The Suite is a collection of over 400 Pascal programs designed to test the quality of Pascal Compilers and their runtime systems for compliance with the ISO (International Standards Organization) Pascal standard. OmegaSoft is the only supplier of 6809 native Pascal compilers that publishes this report in its Instruction manual.

DEBUGGER

The compiler package includes an interactive, symbolic debugger. The debugger allows setting of breakpoints, displaying and changing variables, and tracing statements. The debugger allows very fast turnaround for programs to be run on the host system.

6809 TARGET SYSTEM

The target system may be any 6809 system. No specific I/O devices are required. The output code is re-entrant and rom-able, perfect for single-board systems up to large development systems. There are no charges for use of the output of the compiler or the object of the runtime library in your products.

6809 HOST SYSTEM

The host system must be 6809 based and have at least 48K of ram (56K recommended) and run one of the following operating systems: MDOS, XDOS, OS—9, or FLEX. Priced from $425.

6809 SUPPORT PRODUCTS

The OmegaSoft Relocatable Assembler and Linking Loader is designed to support the Pascal Compiler Package and can also be used for general assembly language program development. Priced from $125.

OmegaSoft's Screen Editor supports smart terminals and comes complete with the Pascal source. Priced from $90.

For complex real-time applications, the Multi-tasking Kernel provides task scheduling, inter-task communications, and resource interlocking. The Kernel is a runtime library that is accessible as Pascal functions and procedures (with full source included). Priced from $175.

68000 SOFTWARE

A Cross Pascal package is available that runs on a 6809 host system and generates code for a 68000, 68008, or 68010. This package does not include a debugger, but does include a Relocatable Assembler and Linking Loader. Priced from $600.

A Resident 68000 Pascal package is available to run under VERSAdos, with support coming soon for OS-9/68000, CP/M-68K, and UNIX. This package will include the Compiler, Relocatable Assembler, Linking Loader, Debugger, and Screen Editor. Priced from $900.

Dealer and OEM inquiries invited. OmegaSoft products are also available from distributors in Australia and Western Europe, call or write for more information.

OmegaSoft products to run on Motorola development systems are available from Motorola systems distributors in Europe.

OMEGASOFT
CERTIFIED SOFTWARE CORP.
P.O. Box 642
Camarillo, CA 93010
Tel: (805) 987-6426
Telex: 467013

TM: OmegaSoft is a trademark of Certified Software Corporation. MDOS, XDOS, and VERSAdos are trademarks of Motorola. OS-9 and OS-9/68000 are trademarks of Microware. FLEX is a trademark of TSC. CP/M-68K is a trademark of DRI. UNIX is a trademark of A T & T.

TOLL FREE
1-800-338-6800
For Ordering

SOUTH EAST MEDIA

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

CoCo OS-9™ FLEX™
SOFTWARE

ASSEMBLERS

Southeast Media

ASTRUK09

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

A User reports

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

F, CCF - $99.95

TSC

Macro Assembler
The FLEX STANDARD Assembler. F,CCF $50.00

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

Great Plains Comp. Co.

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

OmegaSoft

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

Windrush Micro Systems

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

Computer Systems Consultants

SUPER SLEUTH

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

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

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

Computer Systems Center

DISMATE +

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

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

COMPILERS & DECOMPILERS

6809 "Structured" Assembly Lang. Compilers

Windrush Micro Systems

PL/9

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

F, CCF - $198.00

Whimsical Developments

WHERBICAL

Need the Ease of Design and Maintainability of "Structured Programming" AND the Speed and Control of Assembly Language? Then WHERBICAL was designed for you! This Single Pass, Recursive Descent Compiler provides the tool for developing simple Utilities to MAJOR Systems in Assembly Language. Supports 3 "Lex" Levels which allow one level of Procedure nesting, or more within "Modules". It is easy to develop programs written for other machines since you are working at the Assembly Language level. Features unified, user-defined I/O; produces ROMable, relocatable, recursive, re-entrant Code; Structured style and statements with Procedures and Modules; supports Byte and Double-Byte primitives with 3 types of Integers (up to 32 bit), Char and Boolean, and unlimited sized Arrays (vectors only); Interrupt handling; unlimited length Variable Names; Variable Initialization (defaults to $00); Include "Source File" directive; Conditional compiling; direct Code insertion; control of the Stack Pointer; etc. To quote Ron Anderson in his review of WHERBICAL in the Sept. '83 Issue of '68' Micro Journal that, except for the lack of floats, "..., I have to give this one VERY high rating, ...". It is a FAST Compiler which produces FAST Code (his "Primes" Benchmark ran at 9 secs. on a 2 MHz System).

F and CCF - $195.00

'C' Compilers

Windrush Micro Systems

C Compiler

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

F and CCF - $295.00

Introl

C Compiler

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

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

VISA

MasterCard

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

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

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

PASCAL Compilers

TSC

PASCAL Compiler

Native Code Compiler (UCSD Oriented).

F and CCF - $200.00

Lucidata

PASCAL Compiler

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

F and CCF 5" - $198.00
F 8" - $285.00

OmegaSoft

PASCAL Compiler

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

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

DECOMPILERS

Southwest Media

DUB (A UniFLEX "basic" De-Compiler)

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

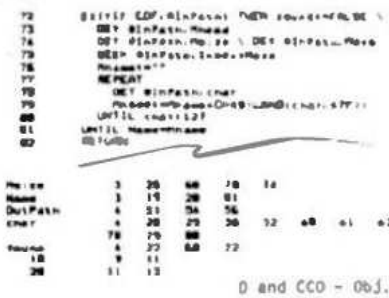
U - $219.95

UTILITIES

Southeast Media

BasicO9 XRef

This BasicO9 Cross Reference Utility is a BasicO9 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also included is a **Program List Utility** which outputs the listing without the overhead of building the cross reference table, which allows it to run considerably faster when only a "pretty printed" listing with line numbers is desired. Requires BasicO9 or Run9 for operation.



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

Southeast Media

OS-9 VDisk

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

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

TOLL FREE
1-800-338-6800
For Ordering

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

CoCo OS-9™ FLEX™
SOFTWARE

Southeast Media

O-F

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

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

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

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

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

F and CCF - $79.95

Southeast Media

COPYMULT

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

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

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

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

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

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

Completely documented source files included.

ALL 4 Programs (8" or 5") **$99.50**

Southeast Media

CHESS 6809

Requires FLEX and DISPLAYS **On Any Type Terminal**

Features:

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

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

F and CCF - $79.95

VISA MasterCard

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

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

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

Southeast Media

DIET-TRAC Forecaster

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

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

F - $59.95
U - $89.95

Southeast Media

XDATA

A **COMMUNICATION** Package
for the UniFLEX Operating System

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

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

U - $299.99

Southeast Media

-JUST

Text Formatter

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

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

F and CCF - $49.95

Lucidata

PASCAL UTILITIES

Requires LUCIDATA Pascal ver 3.

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

F and CCF - $25.00

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

F and CCF - $25.00

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

F and CCF - $25.00

Lucidata

COPYCAT

Pascal NOT required

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

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

Computer Systems Consultants

FLEX DISK UTILITIES

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

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

F and CCF - $50.00

WORD PROCESSORS

Alford and Associates

SCREDITOR III

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

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

Great Plains Computer Co.

STYLOGRAPH

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

SPECIAL. CCF - $195.00

F and O - $295.00 U - $395.00

SPELL

Fast Computer Dictionary.

F, CCF, OS/9 - $125.00 U - $175.00

MAIL MERGE

Greatly extends the power and flexibility of STYLOGRAPH.

F, CCF, O - $145.00 U - $195.00

"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

Great Plains Computer Co.

MAIL MERGE

Greatly extends the power and flexibility of STYLOGRAPH. Allows Multiple Text files to be printed out as one large document. Provides for merging information from the Text File during printing (such as different names and addresses), etc.

F, CCF, O - $145.00
U - $195.00

Southeast Media

SPELL8 "Computer Dictionary"

OVER 120,000 words!

No more "Let your fingers do the walking through the Dictionary" while you are entering Text with your favorite Editor or Word Processor. SPELL8 is more than just "another Spelling Checker"; it allows you to look up a word from within your Editor or Word Processor so that you **KNOW** it is right WHEN YOU TYPE IT IN with the SPH.CMD Utility (which operates in the FLEX Utility Space). Yes, it ALSO allows you to check and update the Text after you are finished; along with allowing you to ADD WORDS to the Dictionary, "Flag" questionable words in the Text for evaluation later, "View a word in context" before changing or ignoring, etc. 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

Great Plains Computer Co.

SPELL

Fast Computer Dictionary -- allows directly changing the Text File, adding words to the dictionary, etc. 75,000 words in less than 400 sectors.

F, CCF, **OS/9** - $125.00
U - $175.00

DATA BASE MANAGEMENT SYSTEMS

Southwestern Applied Business Systems

XDMS

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

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

Great Plains Computer Co.

REMARK DBMS

An XBASIC, Menu Driven, DBMS with "Built-In" Audit Tracking, Extremely Powerful Report & Format Capabilities, etc. This **Time Proven** DBMS will become the "Work Horse" of your Software Stable.

F and CCF $295.00
U $395.00

ACCOUNTING PACKAGES

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

TOLL FREE
1-800-338-6800
For Ordering

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343

for information
call (615) 842-4601

CoCo OS-9™ FLEX™

SOFTWARE

Computer Systems Consultants

BASIC UTILITY PROGRAMS

Ten BASIC Programs to:

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

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

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

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

F and CCF - $25.00
UniFLEX - $50.00

Computer Systems Consultants

FULL SCREEN INVENTORY/MRP

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

F and CCF - $100.00, U - $150.00

BUSINESS FORECASTING

The Virginia Company ***Bizpack***

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

F and CCF - $135.00
with Source - $250.00

*** .SPECIAL ***

Purchase XBASIC and BIZPACK together for $221.50
-- a Savings of $13.50 --

VISA

MasterCard

TOLL FREE
1-800-338-6800
For Ordering

SOUTH EAST MEDIA

5900 Cassandra Smith Rd.
Hixson, TN 37343
info (615) 842-4601

CoCo OS-9™ FLEX™

SOFTWARE

Availability Legends ----

F = FLEX, **CCF** = Color Computer **FLEX**
D = OS-9, **CCD** = 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

*** SPECIAL ***
Purchase XBASIC and BIZPACK together for $221.50
— a Savings of $13.50 —

Computer Systems Consultants
TABULA RASA SPREADSHEET

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

F and CCF - **$100.00.** U - **$125.00**

Computer Systems Center
DYNACALC

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

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

ODDS & ENDS

Computer Systems Consultants
FULL SCREEN FORMS DISPLAY

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

F and CCF - **$50.00.** U - **$75.00**

Computer Systems Consultants
FULL SCREEN MAILING LIST

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

F and CCF - **$100.00.** U - **$110.00**

COLOR COMPUTER SOFTWARE

Stearns Electronics
FORTH

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

Color Computer TAPE - $58.95

Custom Software Engineering, Inc.
Color Computer **GRAPHIC SCREEN PRINT** Programs

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

CSPR	for R.S. LP-VII/VIII & DMP 100/200/400	$7.95
CSPRE	for Epson w/ Graftrax and Graftrax +	$9.95
CSPRG	for Gemini 10 and 15	$9.95
CSPRP	for the Prowriter Printers	$9.95

Custom Software Engineering, Inc.
DATE-O-BASE CALENDAR Program

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

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

Custom Software Engineering, Inc.
That's INTERESTing

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

TAPE - **$29.95**

Custom Software Engineering, Inc.
DISK DATA HANDLER 64K

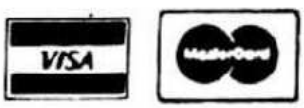
An EXTENDED BASIC Data Management System w/ Mach. Lang. Routines. Allows a max of 246 Chars. and 14 Fields per Record, and another Record can be linked to the first 8 Char. Field Names, up to 99 Chars. per Field. Powerful On-Screen editor for input and update, flexible Output capabilities including output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

DISK - **$54.95**

Custom Software Engineering, Inc.
DISK DOUBLE ENTRY

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

DISK - **$44.95**



*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

Computer Systems Center

OS9SHARE

— Multi-User, Multi-Tasking with **FLEX** —

Southeast Media is now shipping OS9SHARE **FROM STOCK** - the multi-user, multi-tasking capability of **OS9SHARE** allows FLEX users the advantages of more sophisticated and time saving computer usage without having to buy or learn a new language or Operating System syntax. **OS9SHARE**, as its name implies, allows true "time-sharing" operation under the popular FLEX Operating system, and also allows each user to run two simultaneous jobs (multi-tasking); even on single-user systems. For example, while in EDIT, you can list another file or examine a directory. Or, you might look up an item in a Data Base while a Sort is in progress! **OS9SHARE** also provides some fringe benefits that will be greatly appreciated by FLEX users, including typeahead, command line editing, and instant response to "escape".

OS9SHARE is the painless method! Use your existing Flex computer by simply adding 64K of RAM for each user and/or task. Fact is, you still use FLEX just like you always have! **OS9SHARE** is not intended as competition to UniFLEX. It does not improve on the speed of FLEX, and does not offer password protection or other niceties of a full-blown multi-user system. What **OS9SHARE** does do is give FLEX users a **low-cost way to use existing software in a multi-user, multi-tasking environment**, so your existing FLEX versions of BASIC, XBASIC, editors, assemblers, disassemblers, sort/merge packages, word processors, compilers, DYNACALC spread-sheet package, and so on are still good.

NOTE -- The **initial release** of OS9SHARE is for SMITH 8/09 computers, but versions will also be available for other popular extended-memory (up to **512K**) systems, such as HELIX and CMDX. A minimum of **128K** of **RAM** will be required with **ALL** versions. **OS9SHARE** requires **64k of RAM for each active task**; thus a 256k system could allow foreground-background operation on two terminals, or foreground-only operation on four terminals.

AVAILABLE NOW from **Southeast Media** - $200.00

AUTHORS - PROGRAMMERS
QUALITY SOFTWARE NEEDED
FLEX - UniFLEX - OS/9 - Color Computer

For the past several months, we at the **Southeast Media Division** of **Computer Publishing, Inc. (CPI)**, the parent company of **'68' MICRO JOURNAL** and **COLOR MICRO JOURNAL**, have debated expanding our software distribution business. Many other magazines have been doing so for years (in fact, MOST were in the Software Distribution Business BEFORE they began to publish a Magazine). Presently there are many fine examples of software that has been developed by YOU, our readers, that will never see the "light of day" due to the **Cost of Advertising** and **TIME and Cost** involved in the **production, distribution,** and **Customer SUPPORT** of that software unless SOMEONE, with enough exposure and the willingness to continually advertise, runs with the ball.

Software is the "backbone" for the REAL utilization of any Computer System, and ours are no exception! This has been no simple decision. While we realize that there could be some conflict with some of our advertisers, we ALSO hear a LOUD and CONTINOUS cry for **HELP** from our Readers. From day one, the **foremost concern** of '68' MICRO JOURNAL has been it's **READERS**! Therefore, our **Southeast Media Division** will accept, for appraisal for possible Distribution, 6809 software: Games, Utilities, Software Development, Business Application Programs, etc.

In the past there has been too much software offered that was not quite ready. We will strive to eliminate that element. But, right up front, we tell you only that we will do our very best; nothing more. Also, we will strive to keep cost to a bare minimum, while securing for the author a fair return in royalty payments, promptly paid, and in customer support for his product.

Of course, we will expect, no -- **DEMAND**, that the author keep the product free of errors (bugs), and maintain it in a prompt and business like manner. Also we shall require that authors be willing to furnish 'source' for those programs that justify, by price and utility, inclusion of same. The lack of source code, properly commented, is a continual complaint we hear. Not all programs will be sold with source, but where necessary, we will insist that it be included.

In some instances the program may be small or short and not justify itself as a "single" sale product. In this event it will be combined with other like programs, and offered as a package. In that event, the royalties will be split between the various authors.

If you have software that you feel will qualify under this program, please contact one of the people below. Remember, if your software has any problems or "funnies" — **GET IT STRAIGHT BEFORE YOU CONTACT US!!** Also get your source code in proper shape and well commented; there is too much 99% code already drifting around.

If your software is **READY** contact:
Bob Bay, Don Williams, or Tom Williams

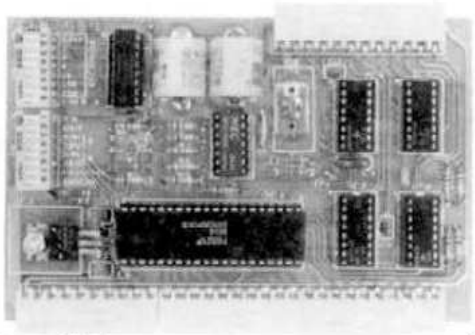
Southeast Media is a division
of **Computer Publishing, Inc. (CPI)**,
a family of 100% 68XX support facilities.

VISA

*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

CALENDAR-CLOCK / TIMER / PARALLEL PORT



Calendar - Clock CLK68-1

- Keeps date and time whether or not the computer is on
- All clock functions software selectable
- On card battery trickle-charged and electrically isolated even for months
- Day of week, month/day/year, hour:minute :11/12 hr

Interval Timer

- 256 prioriy encoding, multi-cascaded, etc.
- Compatible with OS-9** and Flex*
- 65.5 msec tasked with Clock.. down with inputs used as SYNC or.t
- Consumes interrupt internally from the backplane in 2 MHz use.

Parallel I/O Port -- Fully buffered 8 bit parallel port

- 8 bit address select input or output buffering (as jumpers to the board)
- Compatible with parallel printer drivers in most workbees or BASIC

Construction -- Fully socketed, solder masked, & silk screened

Manual -- Well documented - 36 pages

Dealer & OEM discount available

Assembled and tested	$119.95	Kit	$89.95
Goldplated bus conn	2.50	2 MHz option	2.50
Disk 5 or 8 in. SSB or Flex* OS-9 Available NOW			14.95

* OS-9 is a trademark of Microware Systems Corporation
* Flex is a trademark of Technical Systems Consultants, Inc.

ROBERTSON ELECTRONICS
1003 Warm Sands Dr. SE
Albuquerque, NM 87123

Phone (505) 294-0025
NM residents add 4% tax
Add $3 Shipping & Handling

COMPARE

our EPROM PROGRAMMER with the field.

All data taken directly from manufacturer's current advertising. Software, interfaces, or personality modules may also be required at additional cost.

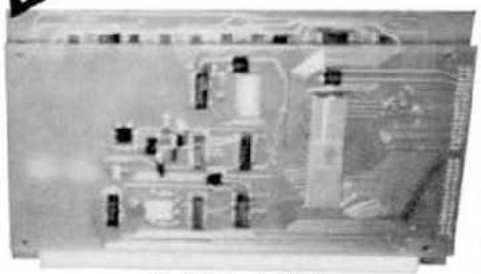
- Triple voltage EPROM
- Supplied in kit form

	A	B	C	D	E	F
INTERFACE	S30	PAR	PAR	SER	S30	SER	SER
INTELLIGENT	NO	NO	NO	YES	NO	YES	YES
PROGRAMS							
2704*			●				●
2508	●			●	●	●	
2708*			●		●		●
2758	●	●	●	●		●	
2516	●	●	●	●	●	●	
2716	●	●	●				●
2716*			●		●		●
2532	●		●	●	●	●	●
2732	●		●	●	●	●	●
2732A	●		●	●	●	●	●
2564	●		●	●	●	●	●
2764	●		●	●	●	●	●
2528	●			●			
27128	●						
2816							●
68764			●				
6748					●		
6749					●		
TOTAL	11	3	12	8	11	11	11
PRICE	$125	$45*	$169	$289	$375	$489	$575

EPROM Programmer, $125. Personality module for 2508, 2758, 2516, and 2716 included. Specify CPU, disk size, and operating system (TSC's FLEX or SSB's OS9) when ordering. Manual only, $10; refundable with EPROM purchase.

UNITEK • P.O. Box 671 • Emporia, VA 23847

SUPER CPU . . . only from LSI 68008



449.95 KIT

We're mighty proud of our new processor card. We're giving you the ability to go 68000 without major changes to your system. Our new CPU gives you these advanced features:

- Dynamic partitioning memory management unit with bound check register.
- On-board timer for multi-user/multi-tasking applications.
- On-board boot-strap EPROM and Monitor EPROM space.
- Vectored priority interrupt generator.
- On-board wait state generator.
- User selectable bus options that includes a new higher bandwidth bus mode.
- And many more.

S68K/08-CPU-ASSEM & TESTED 549.95
S68K/08-CPU-KIT FORM 449.95
(KIT INCLUDES PROCESSOR, CRYSTAL, SOCKETS AND CONNECTORS)
DISK CONTROLLER SUPPORTED DC3, DC4, DMF2, SDC8.

Announcing . . .

THE *SHELL* FOR FLEX 9™ *

We are pleased to announce the *SHELL*, a UNIX ++ like shell that supports I/O redirection, pipes, macro substitution and programmable shell scripts! The shell will work with all your existing programs and utilities. Requires 56K of user ram, FLEX 9™ * version 2.0 and above. The shell occupies the top 8K of user ram. An excellent tool for the 6809 community.

FLX/SHO9-8 . . .8 inch version 90.00
FLX/SHO9-5 . . .5.25 inch version . . . 90.00
ONE YEAR MAINTENANCE 22.50

LSI STANDS BEHIND ITS PRODUCTS
1 YEAR LIMITED WARRANTY
ON ALL OUR PRODUCTS

ANNOUNCING THE LSI 68000 USERS GROUP

Join the Group by sending us your name and address. You will receive our monthly publication with free public domain user programs and software updates.
New members of the users group will receive a $30.00 discount on THEIR FIRST LSI hardware purchase. Anyone that donates a program to the group will receive our current user group software on a formatted CP/M readable disk.

256K RAM CARD
Using the latest LSI technology this 256K RAM CARD makes a perfect addition to your S68K system. Uses MRDY for refresh arbitration.
S68K/08-256K . . . $750.00

SDC8 CONTROLLER
A SS50 DMA disk controller for use with either 68000 bus modes or 6809 bus modes. Features a high reliability digital data separator. (No analog circuits to drift) and full 1 Megabyte addressing range.
S68K/08-SDC8-$550.00

CP/M-68K +
DRI's famous CP/M made for the 68000. It includes a C compiler, relocatable assembler, linking loader, librarian and many utilities. It is source compatible with all other CP/M operating systems.
CP/M-68K + ...$350.00

N.Y. residents add sales tax.

VISA master charge

and COD accepted

*FLEX 9™ is a trademark of Technical Systems Consultants Inc.
+ CP/M-68K is a registered trademark of Digital Research, Inc.
++ UNIX is a registered trademark of Bell Labs.
All prices and offers subject to change without notice.

LSI Enterprises Ltd.

PO Box 1227
Woodhaven, NY 11421
(212) 423-5596

TEN MOST-ASKED QUESTIONS about DYNACALC™

THE ELECTRONIC SPREAD-SHEET FOR 6809 COMPUTERS

1. **What is an electronic spread-sheet, anyway?**
Business people use spread-sheets to organize columns and rows of figures. DYNACALC simulates the operation of a spread-sheet without the mess of paper and pencil. Of course, corrections and changes are a snap. Changing any entered value causes the whole spread-sheet to be re-calculated based on the new constants. This means that you can play, 'what if?' to your heart's content.

2. **Is DYNACALC just for accountants, then?**
Not at all. DYNACALC can be used for just about any type of job. Not only numbers, but alphanumeric messages can be handled. Engineers and other technical users will love DYNACALC's sixteen-digit math and built-in scientific functions. You can build worksheets as large as 256 columns or 256 rows. There's even a built-in sort command, so you can use DYNACALC to manage small data bases — up to 256 records.

3. **What will DYNACALC do for ME?**
That's a good question. Basically the answer is that DYNACALC will let your computer do just about anything you can imagine. Ask your friends who have VisiCalc™, or a similar program, just how useful an electronic spread-sheet program can be for all types of household, business, engineering, and scientific applications. Typical uses include financial planning and budgeting, sales records, bills of material, depreciation schedules, student grade records, job costing, income tax preparation, checkbook balancing, parts inventories, and payroll. But there is no limit to what YOU can do with DYNACALC.

4. **Do I have to learn computer programming?**
NO! DYNACALC is designed to be used BY non-programmers, but even a Ph.D. in Computer Science can understand it. Even experienced programmers can get jobs done many times faster with DYNACALC, compared to conventional programming. Built-in HELP messages are provided for quick reference to operating instructions.

5. **Do I have to modify my system to use DYNACALC?**
Nope. DYNACALC uses any standard 6809 configuration, so you don't have to spend money on another CPU board or waste time learning another operating system.

6. **Will DYNACALC read my existing data files?**
You bet! DYNACALC has a beautifully simple method of reading and writing data files, so you can communicate both ways with other programs on your system, such as the Text Editor, Text Processor, Sort/Merge, STYLOGRAPH™ word processor, RMS™ data base system, or other programs written in BASIC, C, PASCAL, FORTRAN, and so on.

7. **How fast is DYNACALC?**
Very. Except for a few seldom-used commands, DYNACALC is memory-resident, so there is little disk I/O to slow things down. The whole data array (worksheet) is in memory, so access to any point is instantaneous. DYNACALC is 100% 6809 machine code for blistering speed.

8. **Is there a version of DYNACALC for MY system?**
Probably. You need a 6809 computer (32k minimum) with FLEX™, UniFLEX™, or OS-9™ operating system. You also need a decent crt terminal, one with at least 80 characters per line, and direct cursor addressing. If your terminal isn't smart enough for DYNACALC, you probably need a new one anyway. The UniFLEX and OS-9 versions of DYNACALC allow you to mix different brands of terminal on the same system. There's also a special version of DYNACALC for Color Computers equipped with FLEX (Frank Hogg or Data-Comp versions).

9. **How much does DYNACALC cost?**
The FLEX versions are just $200 per copy; UniFLEX version $395; OS-9 version (works with LEVEL ONE or LEVEL TWO) $250. Orders outside North America add $7 per copy for postage. We encourage dealers to handle DYNACALC, since it's a product that sells instantly upon demonstration. Call or write on your company letterhead for more information.

10. **Where do I order DYNACALC?**
See your local DYNACALC dealer, or order directly from CSC at the address below. We accept telephone orders from 10 am to 6 pm, Monday through Friday. Call us at 314-576-5020. Your VISA or MasterCard is welcome. Please specify diskette size for FLEX or OS-9 versions. Software serial number is required for the UniFLEX version.

Order your DYNACALC today!

Foreign Dealers:

Australia & Southeast Asia: order from Paris Radio Electronics, 161 Bunnerong Road (PO Box 380) Kingsford, 2032 NSW Australia. Telephone: 02-344-9111.

United Kingdom: order from Compusense, Ltd., PO Box 169, London N13 4HT. Telephone: 01-882-0681.

Scandinavia: order from Swedish Electronics hk AB, Murargatan 23-25, Uppsala S-754 37 Sweden. Telephone: 18-25-30-00.

Computer Systems Center
13461 Olive Blvd.
Chesterfield, MO 63017
(314) 576-5020

UniFLEX software prices include maintenance for the first year.

DYNACALC is a trademark of Computer Systems Center

VisiCalc is a trademark of VisiCorp.
STYLOGRAPH is a trademark of Great Plains Computer Co.
RMS is a trademark of Washington Computer Services.
FLEX and UniFLEX are trademarks of TSC.
OS-9 is a trademark of Microware and Motorola.

DYNAMITE+™

"THE CODE BUSTER"

disassembles any 6809 or 6800
machine code program into beautiful source

- Learn to program like the experts!
- Adapt existing programs to your needs!
- Convert your 6800 programs to 6809!
- Automatic LABEL generation.
- Allows specifying FCB's, FCC's, FDB's, etc.
- Constants input from DISK or CONSOLE.
- Automatically uses system variable NAMES.
- Output to console, printer, or disk file.
- Available for all popular 6809 operating systems.

FLEX™ $100 per copy; specify 5" or 8" diskette.
OS-9™ $150 per copy; specify 5" or 8" diskette.
UniFLEX™ $300 per copy; 8" diskette only.

For a free sample disassembly that'll convince you DYNAMITE + is the world's best disassembler, send us your name, address, and the name of your operating system.

Order your DYNAMITE+ today!

See your local DYNAMITE + dealer, or order directly from CSC at the address below. We accept telephone orders from 10 am to 6 pm, Monday through Friday. Call us at 314-576-5020. Your VISA or MasterCard is welcome. Orders outside North America add $5 per copy. Please specify diskette size for FLEX or OS-9 versions.

Foreign Dealers:

Australia & Southeast Asia: order from Paris Radio Electronics, 161 Bunnerong Road (PO Box 380) Kingsford, 2032 NSW Australia. Telephone: 02-344-9111.

United Kingdom: order from Compusense, Ltd., PO Box 169, London N13 4HT. Telephone: 01-882-0681.

Scandinavia: order from Swedish Electronics hk AB, Murargatan 23-25, Uppsala S-754 37 Sweden. Telephone: 18-25-30-00.

Computer Systems Center
13461 Olive Blvd.
Chesterfield, MO 63017
(314) 576-5020

UniFLEx software prices include maintenance for the first year.

DYNAMITE + is a trademark of Computer Systems Center.

FLEX and UniFLEX are trademarks of TSC.
OS-9 is a trademark of Microware and Motorola.

Dealer inquiries welcome.

'68' MICRO JOURNAL

★ The only ALL 6800 Computer Magazine.

★ More 6800 material than all the others combined:

MAGAZINE COMPARISON
(2 years)
Monthly Averages

	6800 Articles			TOTAL
KB	BYTE	CC	DOBB'S	PAGES
7.8	6.4	2.7	2.2	19.1 ea. mo.

Average cost for all four each month: $6.53
(Based on advertised 1-year subscription price)

68' cost per month: $2.04

That's Right! Much, Much More
for About
1/3 the Cost!

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: _____

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

SUBSCRIPTION RATES

USA
1 Year $24.50, 2 Year $42.50, 3 Year $64.50

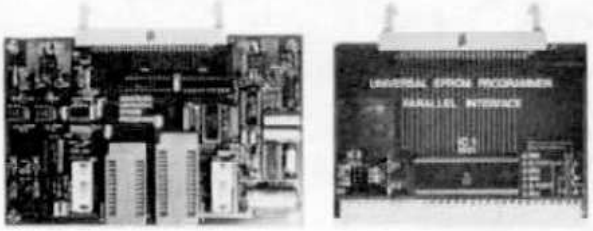
*FOREIGN SURFACE Add $12.00 per Year to USA Price

*FOREIGN AIRMAIL Add $36.00 per Year to USA Price

**CANADA & MEXICO Add $5.50 per Year to USA Price
Cash (USA) or drawn on a USA Bank!!!

WINDRUSH MICRO SYSTEMS

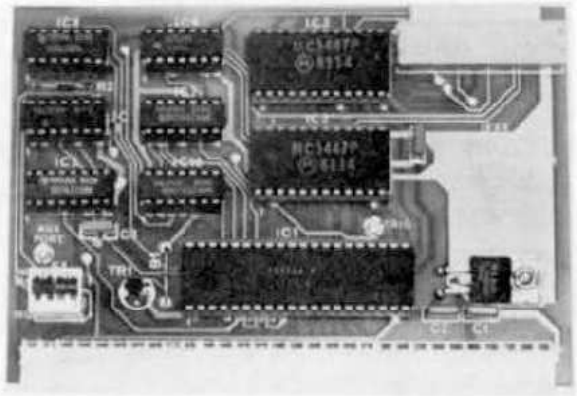
UNIVERSAL EPROM PROGRAMMER



- PROGRAMS and VERIFIES 2508, 2708, 2516, 2716, 2532, 2732A, 2564, and 2764 EPROMS. Minor hardware mods are required to program the INTEL 27128.
- Tri-volt and Single Volt 2508/2708 and 2516/2716 devices are supported.
- ZIF sockets with mode selector switches eliminate 'personality modules'.
- Twin boards with five feet of twisted pair planar cable puts the programmer out on the bench where it belongs.
- SS-30 and EXORCISOR interfaces are available.
- Menu driven software provides the following facilities:
  a. MOVE blocks of memory within the buffer.
  b. READ an EPROM into the buffer.
  c. VERIFY an EPROM against the buffer.
  d. EXAMINE and change the contents of the buffer.
  e. DUMP the contents of the buffer in HEX and ASCII.
  f. FILL a selected area of the buffer with a specified character.
- Software available for all versions of ESB DOS, FLEX 2, FLEX 9 and OS-9. Assembly language source files supplied on disk....enables customizing.
- Well documented users manual provides step-by-step adaptation and operating instructions.

AVAILABLE FROM GIMIX IN THE U.S.A.

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.
- Low level assembly language drivers suitable for 6800, 6801, 6802, 6803, 6808 and 6809 are supplied in the form of listings. These drivers have been extensively tested and are GUARANTEED to work!
- Single SS-30 board (4, 8, or 16 addresses per port), fully socketed, gold plated bus connectors, and IEEE interface cable assembly.

PL/9

EDITOR/COMPILER/DE-BUGGER

- Friendly inter-active environment where you have INSTANT Access to the Editor, the Compiler, and the Trace-Debugger, which, amongst other things, can single step the the program a SOURCE line at a time. You also have direct access to any FLEX utility and your System Monitor.
- 250 page manual is 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), ((OR)XOR), (NOT), (SHIFT), (SWAP)
- Logical operators: (.AND.), (.OR.), (.EOR./.XOR.).
- Control statements: IF..THEN..ELSE, IF..CASE..CASE2..ELSE, BEGIN..END, WHILE.., REPEAT..UNTIL, REPEAT..FOREVER, CALL, JUMP, RETURN, BREAK, GOTO.
- Direct access to (ACCA), (ACCB), (ACCD), (CCR) and (IREG).
- FULLY supports the MC6809 SWI, SWI2, SWI3, NMI, FIRQ, IRQ and RESET vectors. Writing a self-starting (from power-up) program that uses ANY, or ALL, of the MC6809 interrupts is an absolute snap!
- 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 function modules are supplied: IOSUBS, BITIO, HEXIO, FLEXIO, SCIPACK, STRSUBS, and REALCOM.

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

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

MACE/XMACE

A co-resident EDITOR/ASSEMBLER for the 6809 written by Graham Trott which takes most of the pain out of assembly language program development:

- Friendly inter-active environment where you have INSTANT access to the Editor, the Assembler, FLEX and your System Monitor.
- MACE can also produce ASMPROC's for PL/9 with the assembly language source passed to the output file as comments.
- Includes XMACE a co-resident 6800/1/2/3/8 EDITOR/CROSS ASSEMBLER.

'C'

This is the FLEX version of the James McCosh 'C' compiler that is also available on UNIFLEX from SWTP and OS-9 from Microware:

- The FLEX implementation supports the full Kernighan and Ritchie 'C' specification except 'floats', 'doubles', and 'bit-fields'.
- Produces very efficient 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.
- The TSC relocating assembler/linking loader (SP09-17) is REQUIRED.

MACE	(6809 FLEX only).....	$ 98.00
PL/9	(6809 FLEX only)...(a steal at this price!).....	$198.00
'C'	(a 56K system and the TSC SP09-17 package is req'd)....	$295.00
IEEE-488	with IEEE-488 cable assembly.....	$298.00
SS-30	Universal EPROM programmer w/one version of software...	$375.00
EXORciser	Universal EPROM programmer w/one version of software...	$395.00
SOFTWARE	Drivers for a 2nd, 3rd, or 4th operating system.....	$ 25.00

ALL PRICES INCLUDE AIR MAIL POSTAGE

WE STOCK THE FOLLOWING COMPANIES PRODUCTS: GIMIX, SSB, PHI, MICROWARE, TSC, LUCIDATA, AND ALFORD & ASSOCIATES.

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

An SS-50C all CMOS 256K STATIC RAM board will be available SOON!

Write for details & pricing.

WORSTEAD LABORATORIES, NORTH WALSHAM, NORFOLK, ENGLAND. NR28 9SA.
TEL: (0692) 405189
TLX: 97360 SHARET G

FEATURES THE POWERFUL, THIRD GENERATION, MOTOROLA 6809 PROCESSOR!

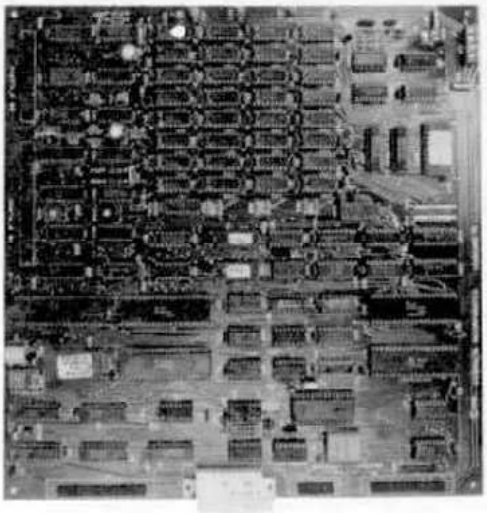
THE 6809 "UNIBOARD"™

SINGLE BOARD COMPUTER KIT

New!

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

64K RAM! DOUBLE DENSITY FLOPPY DISK CONTROLLER!



BLANK PC BOARD
$99.95
WITH PAL'S, AND TWO EPROMS.

FOR 5-1/4 OR 8 INCH SOURCE DISKETTE ADD $10.

$329.00
COMPLETE KIT! FULLY SOCKETED.

ALL OPTIONS ARE STANDARD. NO EXTRAS TO BUY!

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

FEATURES:

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

YOUR CHOICE OF POPULAR DISK OPERATING SYSTEMS:

FLEX™ from TSC	$149
OS9™ from Microware	$199

Specify 5-1/4 or 8 Inch

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

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

Digital Research Computers
(OF TEXAS)
P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 271-3538

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

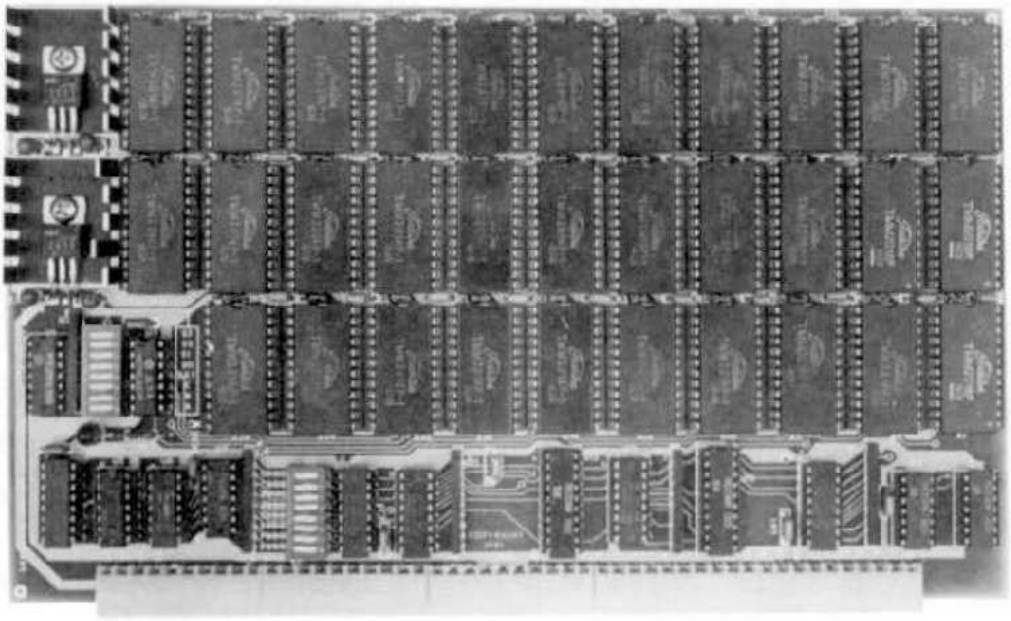
64K SS-50 STATIC RAM

NEW!

$179.00 (48K KIT)

NEW!

LOW POWER!



RAM OR EPROM!

BLANK PC BOARD WITH DOCUMENTATION $52

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

ASSEMBLED AND TESTED **ADD $50**

FEATURES:

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

56K	$219
64K	$249

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 401565 • GARLAND, TEXAS 75040 • (214) 271-3538

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.

OS/9, FLEX, COCO, UNIFLEX SOFTWARE

SUPER SLEUTH DISASSEMBLER EACH $99-FLEX, $101-OS/9, $100-UNIFLEX
interactively generates source on disk with labels, includes xref, label definition, binary file editing, etc.
specify 6800,1,2,3,5,8,9/6502 version or Z-80/8080/85 version

(OBJECT ONLY) EACH $50-FLEX & OS/9, $49-COCO DOS
COCO DOS available in 6800,1,2,3,5,8,9/6502 version only

CROSS-ASSEMBLERS EACH $50-FLEX, $55-OS/9, $60-UNIFLEX, ALL $100
specify for 6800/1, 6502, 6805, Z-80, or 8080/48/85
OS/9 version requires Microware RMA or FHL OSM macro assembler
FLEX version requires TSC ASM5 or FHL ASM or OSM macro assembler

DEBUGGING SIMULATORS EACH $75-FLEX, $100-OS/9, $80-UNIFLEX
specify 6800/1, 6805/146805, 6502, or 6809 OS/9 only)

OS/9 AND UNIFLEX SIMULATORS EACH $100-FLEX
debug OS/9 and UNIFLEX application programs under FLEX using TSC DEBUG facility

6502 TO 6809 ASSEMBLER TRANSLATOR $75-FLEX, $85-OS/9, $80-UNIFLEX
translates 6502 programs to 6809, noting inexact conversions

6800 TO 6809 & 6809 PIC TRANSLATORS $50-FLEX, $75-OS/9, $60-UNIFLEX
translates 6800 programs to 6809, 6809 programs to PIC

FULL-SCREEN FLEX AND UNIFLEX TSC XBASIC PROGRAMS FOR 6809
(with complete cursor control)

DISPLAY GENERATOR/DOCUMENTOR $50-FLEX, $75-UNIFLEX
MAILING LIST SYSTEM $100-FLEX, $110-UNIFLEX
INVENTORY WITH MRP $100-FLEX, $110-UNIFLEX
TABULA RASA SPREADSHEET $100-FLEX, $120-UNIFLEX

DISK UTILITY PROGRAM LIBRARY $50-FLEX
edit disk sectors, sort directory, maintain master catalog, etc.

CMODEM TELECOMMUNICATIONS PROGRAM $50-FLEX & OS/9 & UNIFLEX
provides menu-driven telecommunications facilities, with terminal mode, up/down load, MODEM7 protocol, etc.

5.25" SOFT-SECTORED DISKETTES EACH SET OF 50 $75-SSDD, $85-DSDD
with Tyvek jackets, hub rings, labels

Specify operating system, computer make and type, terminal type.
Programs provided in source form on diskette, specify size and density.
Contact CSC for full catalog and dealer info, printed manuals provided with products.
For VISA and MASTER CARD, give account, exp. date, phone. US funds only.
Add 5% for shipping software, but not for diskettes.
(UNI)FLEX trademark Technical Systems Consultants. OS/9 trademark Microware.

Computer Systems Consultants, Inc.
1454 Latta Lane, Conyers, GA 30207
Telephone Number 404-483-1717/4570

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

!!! FREE !!!

Published Monthly by Computer Publishing Inc., Hixson, TN.

$1.95

Bulk Rate
U.S. Postage
PAID
Chattanooga, TN
Permit No. 357

Color Micro Journal

The Color Computer Monthly Magazine

$1.95 per issue Vol. 1, Issue 2 October, 1983

THIS 'N THAT

The BIG NEWS this month is that OS-9 has finally arrived for the Color Computer. The ASTOUNDING part of the Radio Shack OS-9 Package, besides the price, is the [illegible]. You 'Old Time Radio Shack Followers' will not believe what you see. Jon Shirley has been telling us that the main reason for the "lack" of documentation with a lot of their products was the restrictions placed on releasing that information by Microsoft: I

OS-9 on the COLOR COMPUTER

One of the "Operating Systems of the Future" is **now available** for the "little old Color Computer": OS-9. Freely translated, OS-9 means "Operating System for the 6809" (OS-9 is now being written for the 6809, also). Since it is fairly obvious that UNIX and "UNIX-Type" Operating Systems will be running on just about every computer to come out in the next few years, a whole new language is beginning to appear on the horizon.

Color Computer OS-9; the Package

We had been running a preliminary release of OS-9 on the Color Computer for a few weeks, and received the "Official Radio Shack" version for Review a couple of days ago. To put it mildly, this package is IMPRESSIVE! For $69.95 (Radio Shack Catalog Number 26-3038), you receive a 9 1/2" x 7 5/8" x 2" package containing 4

FREE SAMPLE ISSUE

1-800-338 6800

MON.-FRI. 9-5 E.S.T.

USA-$12.50 per year. Canada& Mexico-$19.50 per year

Surface Foreign-$24.50 per year. Airmail Foreign-$48.50 per year

Color Micro Journal™

TM Color Micro Journal is a trademark of Computer Publishing Inc.

5900 Cassandra Smith Rd. Hixson,TN. 37343

6809 Word Processing System

stylograph™

SPELLING CHECKER
Checks all words against an internal user-expandable dictionary of over 42,000 words.
OS9, FLEX $145 UniFLEX $195

STYLOGRAPH 2.0
The "User Friendly" word processing system. Fewer key strokes by the operator make it easier to learn.
OS9, FLEX $295 UniFLEX $395
COLOR COMPUTER FLEX $195

MAIL MERGE
Inserts names and addresses into form letters and mailing lists. Appends files at print out time. Handles files longer than memory.
OS9, FLEX $125 UniFLEX $175

Inquire about our other software
- Business Programs - G/L, A/R, A/P
- Data Base Management System
- Assemblers

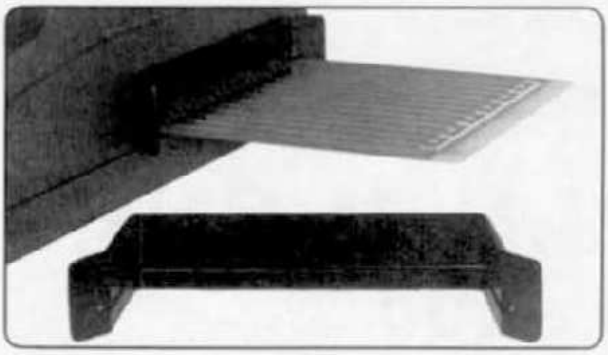
Also, Daisy Wheel Printers $599.

Great Plains Computer Company Inc.
P.O. Box 916
Idaho Falls, Idaho 83401

(208) 529-3210

Flex and Uniflex are trademarks of Technical Systems Consultants, Inc.
OS9 is a trademark of Microware.

6809 SYSTEM DEVELOPMENT



EXPANSION HARDWARE FOR THE TRS-80 COLOR COMPUTER

XPNDR1™

CoCo Expander Card

Gold edge connector plugs into the CoCo cartridge connector. Signals are labeled on the bottom (wire side) with ground and power buses, plated through holes. The 4.3 × 6.2 inch glass/epoxy card is drilled for ICs and components. The finest bare breadboard for your CoCo. Includes 8 page *Application Notes* to help you get started.

$19.95 each or 2 for $36

SuperGuide™

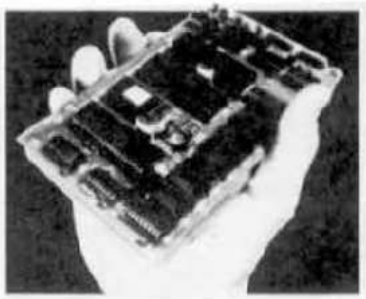
Precision molded plastic insert designed specifically to align and support printed circuit cards in the CoCo cartridge slot, an unbreakable removable card guide. Patent Pending.

$3.95 each

Available now from:

ROBOTIC MICROSYSTEMS

BOX 30807 SEATTLE, WA 98103



NEW!

Compact Flexible 6809 Computer

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

- Small enough to hold in your hand! (Eurocard size: 3.9" x 6.3")
- Two board "system" for greater versatility than single board computers.
- CPU Board — powerful 6809E processor, 16K or 64K RAM, 2K-8K EPROM, 2 RS232 serial ports with software programmable baud rates, 16 bit counter/timer. Run the CPU board all by itself, or plug your own custom board or our FDC board into the expansion connector.
- FDC Board — double-sided/double-density floppy disk controller with adjustment free digital data separator and write precompensation, 2-8 bit parallel ports, 2-16 bit counter/timers, prototyping area.
- Runs FLEX/9, OS-9 under development (trademarks of Technical Systems Consultants, and Microware, respectively).
- Available as bare PC boards or partially assembled boards. All have solder mask both sides plus silkscreened component overlay.
- Special introductory offer includes the ST-2900 CPU and FDC boards (bare boards), documentation, and an EPROM (with monitor Program) for only $85 (US funds) plus $5 shipping/handling ($10 overseas). Canadian residents $85 in Canadian funds plus $4 s/h — B.C. residents add 7% tax. Prices go up after Aug. 15th. Deliveries to begin July '84.

Send money order to:

SARDIS TECHNOLOGIES

Write for free brochure and complete price list.

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

OS-9®
SOFTWARE TOOLS!

- **ADLIB 50.00**
  Allows shared source code
- **ISAM 350.00**
  Indexed Sequential File Access Method
- **SMATH 95.00**
  String Arithmetic
- **SORTC 150.00**
  Full-record disk sort
- **EXAMOD/CHGREV 50.00**
  Determines contents of a module
- **IN-DATA/REPORT-GEN/MENU 75.00**
  Data entry/Menu & Report generator
- **LOOKUP/SLOOKUP 75.00**
  "Wild Card" directory searcher
- **VID 75.00**
  Display/input data edit package
- **XRF 200.00**
  Low-overhead database emulator
- **TERMINAL 95.00**
  Communicates with other machines
- **LDMAC 75.00**
  Assembly code routines "tool box"

For more information, or to place an order, contact:
Dept. 68 8
The JBM Group, Inc.
Continental Business Center
Front & Ford Streets
Bridgeport, PA 19405
Tel: 215-337-3138/TWX: 510-660-3999

VISA/MC accepted: PA residents please add 6% sales tax
U.S. orders please add 5.00 postage/handling

® Registered trademark of Microware

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

OS-9® SOFTWARE

from The JBM Group, Inc.

GENUS

A new stand-alone sort for OS-9 that keeps asking for more data!

- Easy-to-understand prompts ask you for all information necessary to sort your data files
- Allows sorting in ascending or descending order
- Allows unlimited number of sort keys of different types
- Saves both disk and memory space by summing numeric fields as it sorts

CAL

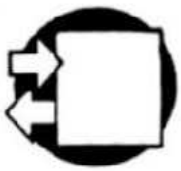
Appointment calendar program to help keep your important dates straight.

- Simple command structure
- Lists appointments by day, week or month
- Revises dates, times, relevant information on existing schedule entries
- Default calendar file for each user
- Unlimited alternate calendar files
- System-wide master calendar file for common-appointment scheduling

THE MESSAGE

The only interoffice memo system currently available for OS-9 Level II users.

- Eliminates lost and confused interoffice communications
- Interfaces easily with your existing word processor/editor; no new commands needed
- Allows hardcopy printout, verification of received messages, immediate reply to sender
- Transmit new messages OR existing files to single user, or 'broadcast' to all users
- Preview Line indicates number, size and topic of messages
- Saves time, money, energy; cuts down on 'office aggravation'



JBM'S MIDWARE

MasterCard VISA

YES! I want OS-9 SOLUTIONS!

Please send me:

GENUS (49.00) _____

CAL (69.00) _____

THE MESSAGE (89.00) _____

PA Sales Tax 6% _____

Postage/Handling $5 _____

TOTAL _____

☐ I'm interested, need more information

SHIP TO: _____

Make Check/Money Order
payable to: The JBM Group, Inc.
Or by VISA/MASTERCARD
Acct. # _____
Exp. date: _____
Signature: _____

Required Distribution: CoCo (Y/N): ____
5¼ 35 track _____
5¼ 40 track _____
5¼ 80 track _____
8" 77 track _____

ORDER FROM

Dept. 68 11
The JBM Group, Inc.
Continental Business Center
Front & Ford Streets
Bridgeport, PA 19405

Tel: 215-337-3138/TWX: 510-660-3999

* OS9 is a registered trademark of Microware Corp.

ARCADE 50

POWERFUL COLOR GRAPHICS
Uses the new TMS9918A Video Display processor. High resolution 256 x 192 pixel display with 15 colors. 16K B tes of onboard AM does not reduce user memory. 32 graphic images can be individually moved with simple X-Y commands for smooth animation. External Video input allow subtitling. NTSC composite video output.

SOUND EFFECTS AND MUSIC
- Three AY3-8910 Programmable Sound Generators
- Nine simultaneous voices
- Three independent noise sources
- Onboard stereo amplifier drives two 8 ohm speakers

ADDITIONAL I/O CAPABILITIES
- Eight analog inputs with 8 bit resolution
- Supports four joysticks with pushbutton switches
- Eight bit parallel I/O port
- Entire unit maps into 256 bytes of memory

FBASIC

TERMINUS DESIGN INC. in conjunction with Microware Systems Corporation, is proud to announce FBASIC an enhancement of Microware's 6800/BASIC. Their fast compiled BASIC has been adapted for 6809 users with added video and sound features for ARCADE 50 users. FBASIC is a true compiler that produces optimized machine language modules which are ROMable and require no Run-Time package. FBASIC requires less memory overhead and runs hundreds of times faster than BASIC interpreters. It supports standard BASIC instruction including String functions. Disk I/O and fast integer arithmetic with multiple-precision capability. Graphics verbs and functions fully support the Arcade 50.

ARCADE 50 assembled and tested	$325.00
Video and Audio connecto set	15.00
4 Joystick connector set	15.00
2 Radio Shack Joysticks	24.00
Gold Molex connectors	12.00
A/BASIC for 6800	110.00
FBASIC for 6809	110.00
FBASIC (with ARCADE 50)	75.00
ARCADE 50 R B	375.00
LABVIDEO (Motorola ExORBus)	375.00
NEW MV09 6809 Processor Board	225.00
256K Dynamic Memor Board	795.00
256K Dynamic Memory Board (w/64K)	395.00
64K Dynamic Memor Board	295.00

TERMINUS DESIGN INC
16 SCARBROUGH ROAD
ELLENWOOD, GA 30049
(404) 474-4866

OS9
APPLICATION SOFTWARE

ACCOUNTS PAYABLE **$349**

GENERAL LEDGER with CASH JOURNAL **$449**

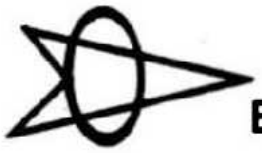
PAYROLL **$549**

ACCOUNTS RECEIVABLE **$349**

SMALL BUSINESS INVENTORY **$349**

COMPLETE DOCUMENTATION $19.95

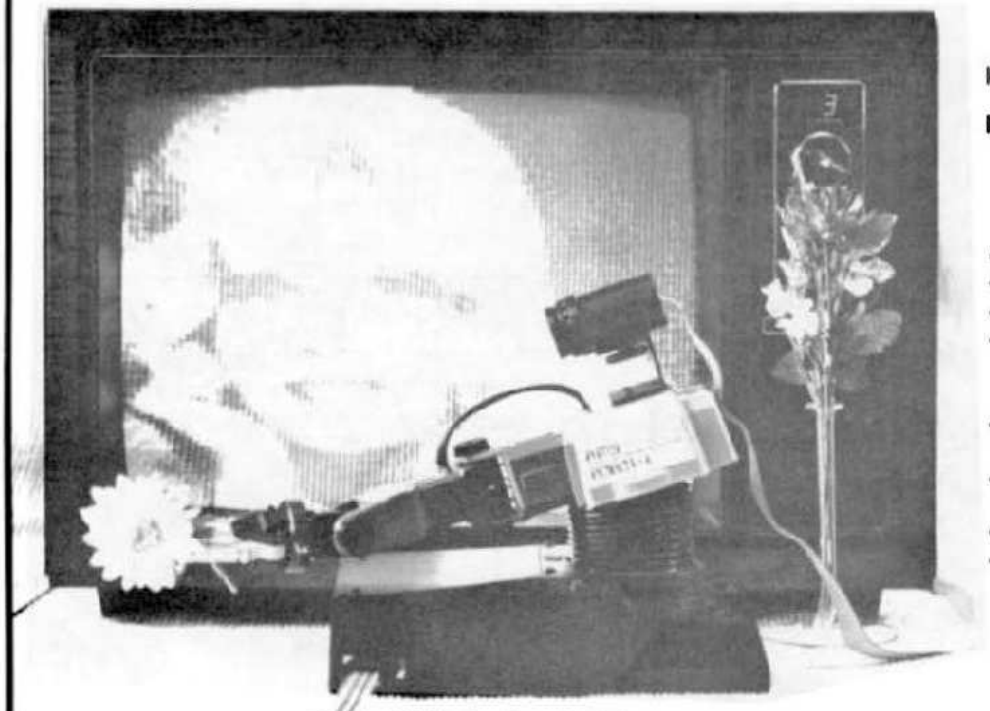
OS9 & BASIC 09 ARE TRADEMARK OF MICROWARE, INC. & MOTOROLA CORP.



SPECIALTY ELECTRONICS

(405) 233-5564
2110 W. WILLOW – ENID, OK 73701

Computer Servo Controlled Robot Arm



Call or Write for Free Catalog

Analog Micro Systems

5660 Valmont Road . Boulder, Colorado 80301 . Tel: (303) 444-6809

Robot-1

Keyboard or Joystick Control

Remembers Everything It Did & does it again

Typical System Includes:
- Robot-1 & Cables
- 6 Channel Servo Controller
- Power Supply
- All Software with source code

Modular Robotic Accessories:
- Mobile Cart for Traveling Robot
- Radio Links between all Functions
- Robot-mounted MicronEye
- Ultrasonic Range Finder

Robot-1 Series

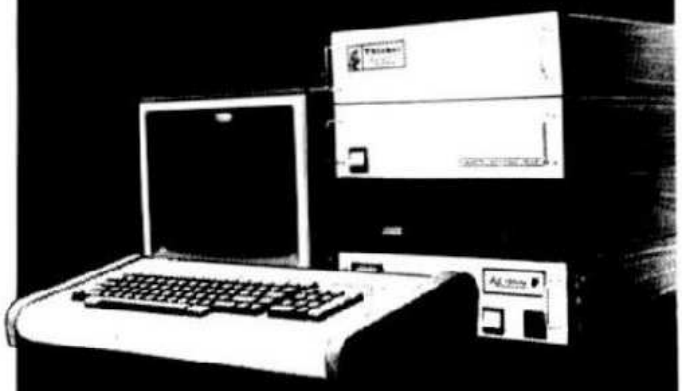
starting at $289.00
for the Color Computer
and 6809 SS50 Computers

Additional Systems Available

Robot-1R for Radio Control Systems

ACORN

COMPUTER SYSTEMS 68-50C



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/regs. & cables less DRIVES	200.00	250.00
MOTHER OARD, 6 SS-50c, 8 SS-30c NMI button	225.00	325.00

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 - EXteeded Addressing BAUD gen. PIA port	29.95	69.95	89.95
MB8 - 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
FD8K - 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	------
TAIAN 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

MONOLINK - 20 Mhz Monochrome video driver		15.00	20.00
CC3O PORT BUS w/power supply 5 SS-30, 2 Cart		169.95	199.95
POWER BOX 6 switched outlets transient suppression		29.95	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 ADVERTISERS INDEX

'68' MICRO JOURNAL 49,60
AAA CHICAGO COMPUTER CENTER 36,37
ACORN COMPUTER SYSTEMS 70
ANALOG MICRO SYSTEMS 69
ANDERSON COMPUTER CONSULTANTS 67
COLOR MICRO JOURNAL 65
COMPILER EVALUATION SERVICES 49
COMPUTER PUBLISHING INC... 5
COMPUTER SYSTEMS CENTER 59,60
COMPUTER SYSTEMS CONSULTANTS, INC.....64
DATA-COMP .....IBC
DIGITAL RESEARCH COMPUTERS 62,63
GIMIX, INC... 3,72
GREAT PLAINS COMPUTER CO... 66
HAZELWOOD COMPUTER SYSTEMS .....OBC
INTROL CORP... 50
JBM 67,68
LSI ENTERPRISES LTD... 58
MICROWARE SYSTEMS CORP... 1,4,12
OMEGASOFT CERTIFIED SOFTWARE CORP. ...51
PERIPHERAL TECHNOLOGY 71
ROBERTSON ELECTRONICS 58
ROBOTIC MICROSYSTEMS 66
SARDIS TECHNOLOGIES 66
SMOKE SIGNAL BROADCASTING 6,7
SOUTH EAST MEDIA 52,53,54,55,56,57
SOUTHWEST TECHNICAL PRODUCTS INC. ...IFC
SPECIALTY ELECTRONICS 69
STAR-KITS 67
TALBOT MICROSYSTEMS 64
TERMINUS DESIGN, INC... 69
UNITEK 58
WESTCHESTER APPLIED BUSINESS SYSTEMS .71
WINDRUSH MICRO SYSTEMS LIMITED 61

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

6809 COMPLETE SYSTEM 6809

ADVANCED TECHNOLOGY ECONOMICAL

The PT69 COMPLETE SYSTEM features the proven PT69 single board computer. Powerful performance + reliability + an unbeatable combination! Complete Systems feature:

- Double Sided/Double Density Drives (40 or 80 track) Cabinet, Power Supply
- Optional CRT and Printer
- 1MHz 6809E Processor
- 2 RS232 Serial Ports
- 2 8-Bit Parallel Ports
- 56K RAM; 4K EPROM; 4K I/O

• COMPLETE SYSTEM with PT69 Board, 2 DS/DD 5¼" 40 Track Drives, Cabinet, and Power Supply	$999.95
• PT69, Assembled, Tested, with Power Supply and Cabinet	$399.95
• PT69, Assembled and Tested	$299.95

— Kits are available —

— Write for information on CRT and Printer —

OEM Inquiries Welcome

PERIPHERAL TECHNOLOGY

"Supplying Your Computer Needs Since 1978"

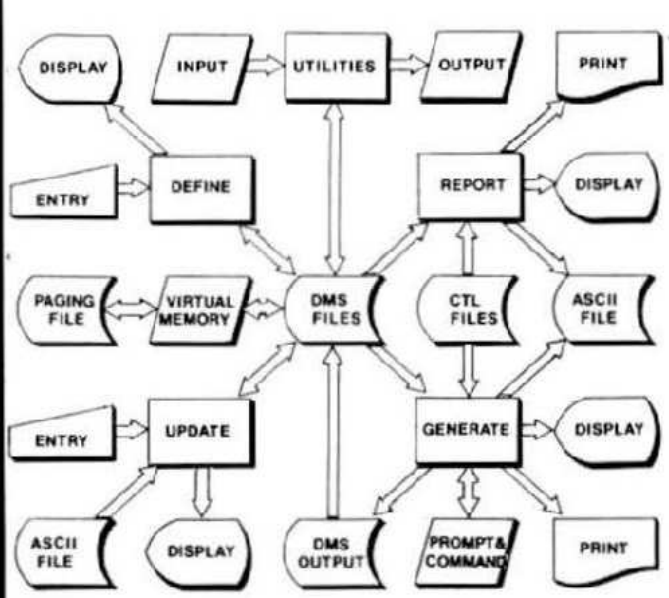
3760 Lower Roswell Road

Marietta, Georgia 30067

VISA/MASTERCARD/CHECK/COD 404/973-0042

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, VMDEX utility and System Documentation for level II. 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 merges, 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, graphic 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). . . $ 25.00

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 seperately 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). . $249.95
XACC System Documentation only ($10, credit toward purchase). . . $ 25.00

WESTCHESTER Applied Business Systems
Post Office Box 187, Briarcliff Manor, N.Y. 10510

All software is written in micro/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). NY Res add sales tax. Specify 5" or 8".

Sales: S. E. MEDIA, 1-800-338-6800, Consultation: 914-941-3552 (evens).

FLEX is a trademark of Technical Systems Consultants, Inc.

GIMIX HAS THE 6809 SYSTEM TO SUIT YOUR NEEDS

HARDWARE

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

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

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

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

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

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

SOFTWARE

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

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

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

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

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

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

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

The **#79** super system now includes (in addition to the above): the **GMX 6809 CPU III, a 256K CMOS Static RAM Board (#72), and a 3-port intelligent Serial I/O Processor (#11).**

The **GMX 6809 CPU III** can perform high-speed DMA transfers from memory to memory and uses memory attributes and illegal instruction trapping to protect the system and users from program crashes. If a user program crashes, only that user is affected; other users are unaware of the problem.

The 3-Port Intelligent Serial I/O Board (#11) significantly reduces system overhead by handling routine I/O functions; freeing the host CPU for running user programs. This improves overall system performance and allows user terminals to be run at up to 19.2K baud.

with dual 40 track DSDD drives	$5998.79
with dual 80 track DSDD drives	$6198.79
with **#88** dual 8" DSDD drive system	$7698.79
with **#90** 19MB Winchester subsystem and one 80 track	$8898.79
with a 47MB Winchester subsystem and one 80 track	$10,898.79
with a 47MB plus a 6MB removable pack Winchester subsystem and one 80 track drive	$12,398.79

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account #73-32033.

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.

OS-9 GMX I / FLEX SYSTEMS #49

The **#49** systems include 64KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DSDD drives	$3998.49
with dual 80 track DSDD drives	$4198.49
with **#88** dual 8" DSDD drive system	$5698.49
with **#90** 19MB Winchester subsystem and one 80 track	$6898.49

OS-9 GMX II / FLEX SYSTEMS #39

The **#39** systems include 128KB static RAM, #05 CPU, #43 2 port serial board.

with dual 40 track DS DD drives	$4498.39
with dual 80 track DSDD drives	$4698.39
with **#88** dual 8" DSDD drive system	$6198.39
with **#90** 19MB Winchester subsystem and one 80 track	$7398.39

GIMIX DOES NOT GUARANTEE PERFORMANCE OF ANY GIMIX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

EXPORT MODELS: ADD $30 FOR 50Hz. POWER SUPPLIES.

GIMIX, Inc. reserves the right to change pricing, terms, and products specifications at any time without further notice.

ALL PRICES ARE F.O.B. CHICAGO

Contact GIMIX for price and availability of UniFLEX and UniFLEX GMXIII Systems.

NOTE on all drive systems: Dual 40 track drives have about 700KB of formatted capacity; dual 80's about 1,400KB; dual 8" about 2,000KB. The formatted capacity of hard disks is about 80% of the total capacity.

Want to expand your system to a megabyte of Static RAM and 15 users?

Simply add additional memory and I/O boards. Your **GIMIX** system can grow with your needs. Contact us for a complete list of available boards and options.

#72 256KB CMOS STATIC RAM board with battery back up	$1898.72
#64 64KB CMOS STATIC RAM board with battery back up	$526.64
#57 64KB STATIC RAM board	$478.67
#11 3 port intelligent serial I/O board	$498.11
#43 2 port serial I/O board	$126.43
#42 2 port parallel I/O board	$88.42
#95 cable sets (1 needed per port), specify board	$24.95

TRADE UP YOUR CoCo!

GIMIX will allow you up to $1100.00 credit toward the purchase of any GIMIX system when you trade-in your working Color Computer, peripherals, and original software. The trade-in value is limited to 110% of the RADIO SHACK™ list price at the time your order is placed. You pay the freight. This offer is good only in the Continental U.S.; is limited to the first 100 orders; and expires on 9/30/84. Only one trade-in per customer.

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

©1984 GIMIX, INC. 4-84

The Original FLEX™ for Color Computers

- Upgrade to 64K
- RS to FLEX, FLEX to RS file transfer ability
- Create your own character set
- Automatic recognition of single or double density and single or doubled sided
- All features available for either single or multiple drive systems
- Settable Disk Drive Seek Rates
- Faster High Resolution Video Display with 5 different formats
- Save RS Basic from RAM to Disk
- Move RS Basic to RAM
- Load and save function on FLEX disk
- 24 Support Commands 12 with Source Text
- External Terminal Program

DATA-COMP has everything you need to make your TRS-80C Color Computer WORK for YOU; from Parts and Pieces to Full Ready To Use Systems. DATA-COMP designs, sells, services, and SUPPORTS Computer SYSTEMS, not just Software. CALL DATA-COMP TODAY to make your Computer WORK FOR YOU!

System Requirements

FLEX9 Special General Version + Editor & Assembler (which normally sell for $50.00 ea.) $250.00

F-MATE(RS) FLEX9 Conversion Rout. for the RS Disk Controller
when purchased with Special General FLEX9 Sys. $ 49.95
when purchased without the General FLEX9 Sys. $ 59.95

COLOR COMPUTER II WITH 64K AND EXT. BASIC $ 239.95

SPECIAL SYSTEM PACKAGES

64K Radio Shack COLOR COMPUTER, Radio Shack COLOR DISK CONTROLLER, a Disk Drive System, Special General Version of FLEX9, F-MATE(RS), and a Box of 10 Double Density Diskettes, a COMPLETE ready to run SYSTEM on your Color TV Set.

$ 929.85

64K Radio Shack COLOR COMPUTER, Radio Shack COLOR DISK CONTROLLER, a Disk Drive System, and a Box of 10 Double Density Diskettes, a COMPLETE ready to run SYSTEM on your Color TV Set.

$ 629.90

NEW PRICE REDUCTION

DISK SYSTEMS FOR THE COLOR COMPUTER

THESE PACKAGES INCLUDE DRIVE, *CONTROLLER, POWER SUPPLY & CABINET, CABLE, AND MANUAL.

* SPECIFY WHAT CONTROLLER YOU WANT J&M, OR RADIO SHACK.

PAK #1 - 1 SINGLE SIDED, DOUBLE DENSITY SYS.	$389.95
PAK #2 - 2 SINGLE SIDED, DOUBLE DENSITY SYS.	$639.95
PAK #3 - 1 DOUBLE SIDED, DOUBLE DENSITY SYS.	$439.95
PAK #4 - 2 DOUBLE SIDED, DOUBLE DENSITY SYS.	$699.95
PAK #5 - 2 THINLINE DOUBLE SIDED, DOUBLE DENSITY SYS.	$659.95

PRINTERS

EPSON RX-80	$325.00
EPSON RX-80FT	$375.00
EPSON MX-100	$675.00
EPSON FX-100	$749.00
EPSON FX-80	$549.00
EPSON HX-20	$230.00

SERIAL BOARDS

MX-Series	$119.95
RX-FX-Series	$ 99.95

H J L Keyboards $ 74.95

FLOPPY DISKETTES

	MEMOREX	VERBATIM
5" Soft Sector Disks		
Single Sided Single Density	$2.40ea.	$2.75ea
Single Sided Double Density	$2.60ea	$2.75ea
Double Sided Double Density		$4.92ea
Plastic Storage Box		$2.00ea
8" Soft Sector Disks		
Single Sided Single Density		$3.75ea
Single Sided Double Density		$4.10ea
Double Sided Double Density		$4.75ea
Plastic Library Box		$5.00ea

PARTS AND PIECES

Radio Shack Disk Controller	$ 134.95	Single Drive Disk Cable for RS Controller	$ 19.95
1 Single Sided, Double Density Disk Drive Tandon	$229.95	Double Drive Disk Cable for RS Controller	$ 24.95
1 Double Sided, Double Density Disk Drive Qume	$319.95	Micro Tech. Prods. Inc. LOWER CASE ROM Adapter	$ 74.95
1 Qume Thinline Double Sided, Double Density	$249.95	Radio Shack BASIC Version 1.1 ROM or 1.2 ROM	$ 34.95
1 Tandon 40 Track SSDD with Cabinet and Power Supply	$259.95	Radio Shack Extended Basic ROM	$ 39.95
Single Drive Cabinet with Power Supply	$ 79.95	Radio Shack Disk Basic ROM 1.1	$ 29.95
Double Drive Cabinet with Power Supply	$109.95	Screen Clean-Clears up Video Distortion On Your Color Computer	$ 39.95
J & M Disk Controller	$144.95	Set of Eight 64K RAM Chips w Mod Instructions	$ 49.95

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

LAST OF PRODUCTION!!!

!! LIMITED QUANTITY !!

SWTPC 8212 & 8212(W) Intelligent Terminals

New & Demo Models

At Discount Prices

Remaining Supply of SWTPC 8212 CRT Terminals

QUME QVT-102 TERMINAL

DATA-COMP

P.O. Box 794 HIXSON, TN 37343

1-615-842-4601

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

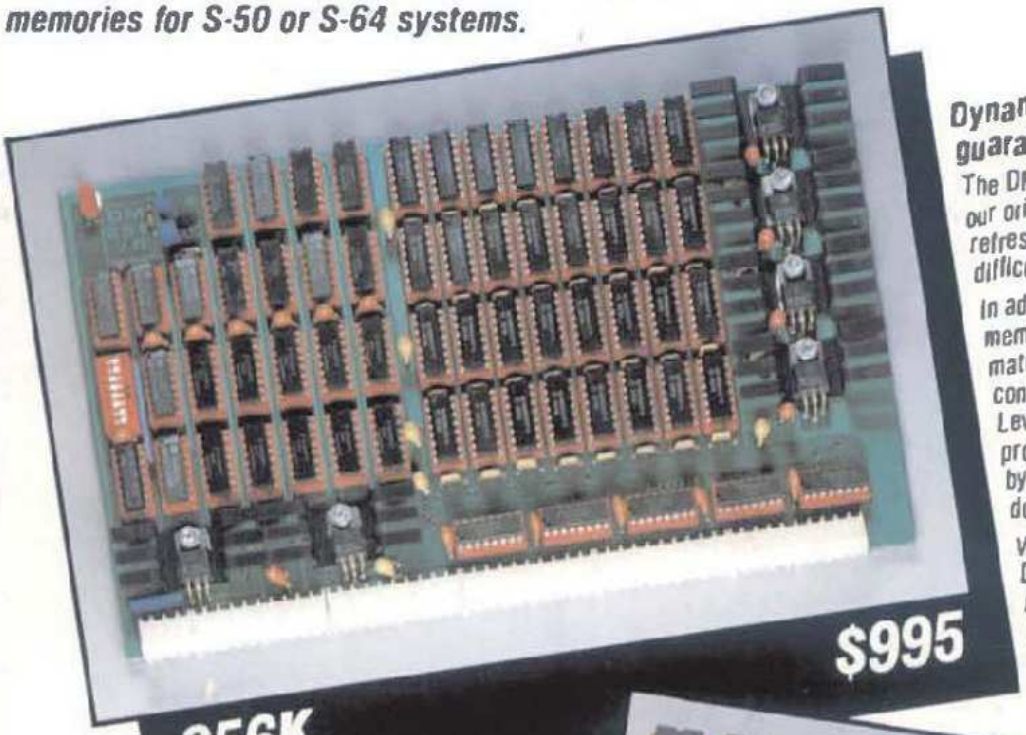
Data-Comp—South East Media & 68 Micro Journal Are Divisions of CPI

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

Second Class Postage Paid
At Chattanooga, TN
ISSN 0194-5025

MEMORY • MEMORY • MEMORY • MEM

Hazelwood Computer Systems presents a pair of advanced memories for S-50 or S-64 systems.



256K

$995

Dynamic memory with guaranteed 2.5 MHz operation

The DM-256 utilizes the same proven refresh method as our original DM-64. This technique allows fully transparent refresh while maintaining full bus speed and eliminating difficult timing problems.

In addition, individual power regulators for each row of memory ICs enhance reliability by providing the ultimate in noise decoupling. Flexible addressing mode controls make the entire 256K available to OS9 Level 2 or UniFLEX. For FLEX, the furnished utility program HYPERDSK,™ formats memory not used by FLEX as a virtual, superfast disk, thus tremendously improving total system performance.

With several hundred already in use, the DM-256 is the best choice for economical, reliable memory.

Assembled, tested, and burned in

Order: DM-256

64K

$395

Static memory with on-board battery backup

Utilizing the latest in low power CMOS static RAMs, the CM-64 has an on-board NiCad battery to maintain memory contents in the absence of bus power. Generous timing margins allow operation at bus speeds in excess of 2 MHz. Zener diode power loss sensing inhibits write operations to prevent data loss during power transitions. Write access can be explicitly controlled by means of an on-board jumper, a remote mounted switch, or an external logic signal. This allows the memory to appear as ROM to the computer after the contents have been "programmed" using normal computer operations. The economical price makes the advantages of non-volatile memory available to any system.

Assembled, tested, and burned in

Order: CM-64

HAZELWOOD COMPUTER SYSTEMS

907 East Terra, O'Fallon, MO 63366, 314-281-1055

HELIX™

HYPERDSK is a trademark of Hazelwood Computer Systems • OS9 is a trademark of Microware • FLEX and UniFLEX are trademarks of TSC