68' MICRO JOURNAL

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

$2.95USA

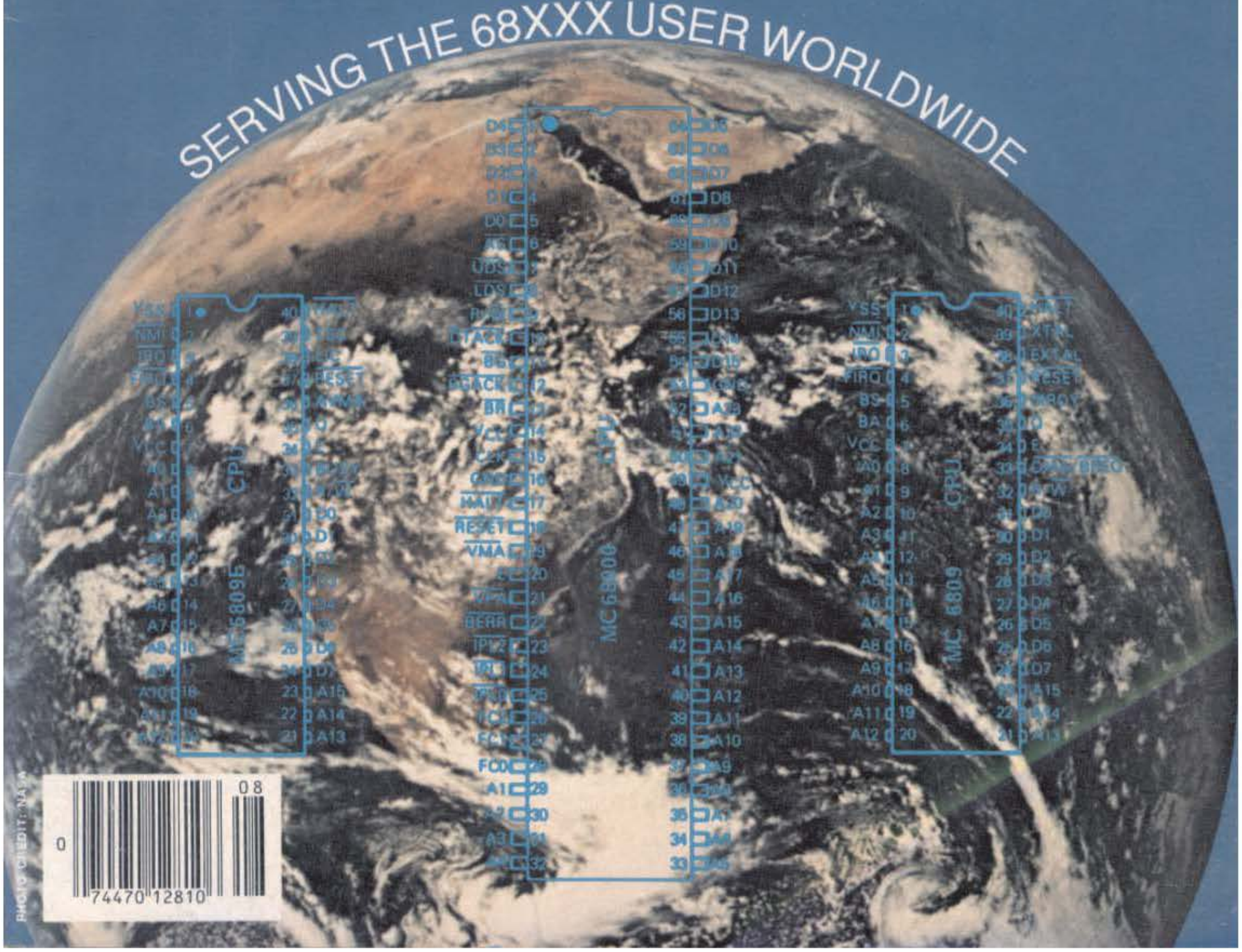
Motorola VME-MACINTOSH-S 50 & Other 68XXX Systems
6809 68008 68000 68010 68020 68030
OS-9 The Magazine for Motorola CPU Devices FLEX SK*DOS
A User Contributor Journal

This Issue:

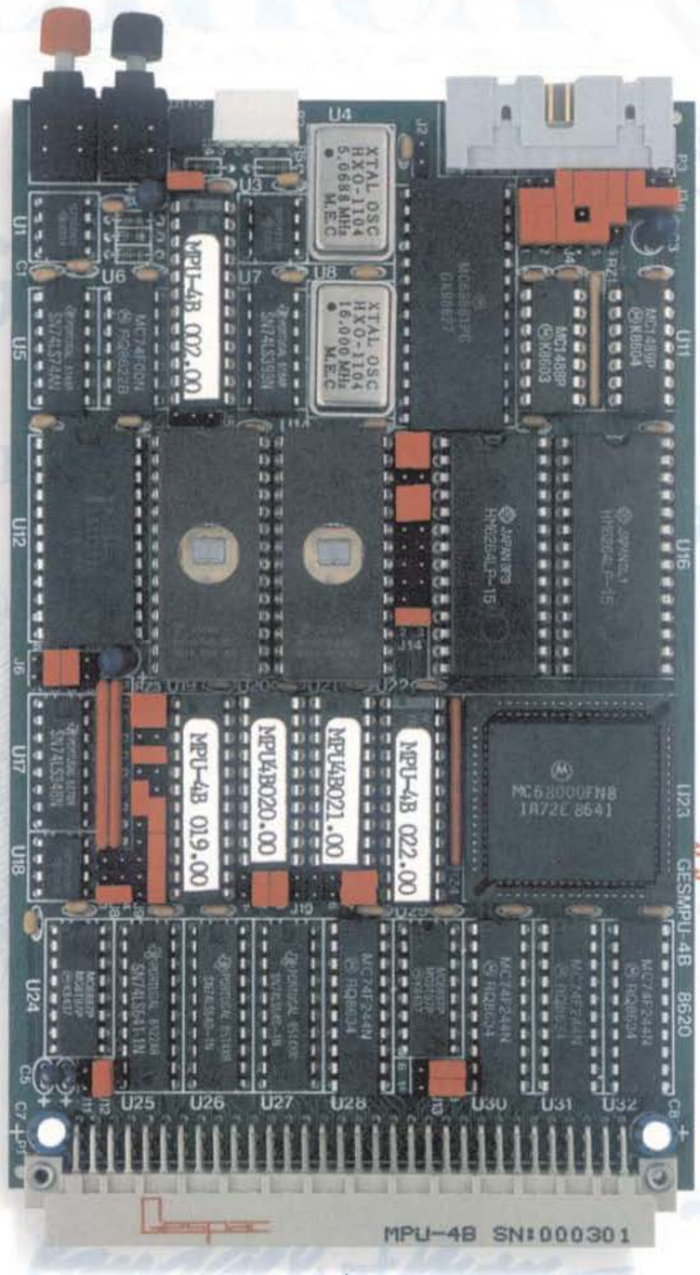
And Lots More!

VOLUME IX ISSUE VIII ● Devoted to the 68XXX User ● August 1987

The Grandfather of "DeskTop Publishing™"



GESPAC Gives You 68000 Performance at 8-bit Prices.



Actual Size

Introducing another Price / Performance breakthrough from GESPAC: A complete 68000 CPU module for $395 unit price. The GESMPU-4B.

Wherever you thought of using an 8-bit microprocessor to do a job, you can now use the 16/32-bit GESMPU-4B instead. It will do the job, better, faster, and best-of-all for the same money.

On a single height Eurocard, we have packed an 8 MHz 68000 microprocessor (16 MHz optional), four sockets for up to 64 Kilobytes of zero-wait-states CMOS RAM and up to 128 Kilobytes of EPROM, one RS-232 serial port, and three 16-bit timers.

The GESMPU-4B is fully expandable through the standard G-64 bus, to accommodate up to 16 Megabytes of external memory. You can add any of more than 300 I/O modules available from GESPAC and a growing numbers of independent G-64 bus vendors.

To make your programming tasks easier, GESPAC supports the GESMPU-4B with the OS-9® and PDOS® real-time, multi-tasking operating systems and most popular high level languages and software development tools.

$395.00
Single Quantity

$316 @ 100 pieces

If you too like the idea of getting more for less, contact us today to receive information on the GESMPU-4B and the G-64 bus concept from GESPAC—the leader in single Eurocard microcomputer products worldwide.

Call Toll Free 1-800-4-GESPAC or Call (602) 962-5559.

Gespac

IN USA - CANADA
50A West Hoover Ave.
Mesa, Arizona 85202
Tel. (602) 962-5559
Telex 386575

INTERNATIONAL
3, chemin des Aulx
CH-1228 Geneva
Tel. (022) 713400
Telex 429989

G-64 and G-96 are registered trademarks of GESPAC SA   OS-9 is a registered trademark of Microware Corp.   PDOS is a registered trademark of Eyring Research Institute Inc

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

GMX MICRO 20 PRICE LIST

MICRO 20 (12.5 MHz) W/1 SAB..............$2565.00
MICRO 20 (16.67 MHz) W/1 SAB.............$2895.00
MICRO 20 (20 MHz) W/1 SAB................$3295.00

OPTIONAL PARTS AND ACCESSORIES

68881RC12................................$ 295.00
68881RC16................................$ 395.00
MOTOROLA 68020 USERS MANUAL..............$ 18.00
MOTOROLA 68881 USERS MANUAL..............$ 18.00

SBC ACCESSORY PACKAGE (M20-AP)......$1690.00
The package includes a PC-style cabinet with a custom backpanel, a Xebec 1410A hard disk controller, a 25 Megabyte (unformatted) hard disk, a 5 1/4" DSDD 80 track floppy disk drive, a 150 watt power supply, cooling fan, panel mounted reset and abort switches and all necessary internal cabling
BACK PANEL PLATE (BPP-PC) For Above.......$ 44.00
2nd 5"80 FLOPPY & CABLES FOR M20-AP, ADD..$ 250.00
SECOND 25MB HARD DISK & CABLES, ADD.......$ 780.00
TO SUBSTITUTE 85MB HD FOR 25MB HD, ADD....$1800.00

I/O EXPANSION BOARDS

16 PORT SERIAL CARD ONLY (SBC-16S)........$335.00
The SBC-16S extends the I/O capabilities of the GMX Micro-20 68020 Single-board Computer by adding sixteen asynchronous serial I/O ports. By using two SBC-16S boards, a total of thirty-six serial ports are possible.

RS232 Adapter(SAB-25,SAB-9D or RJ-45).....$165.00
The board provides level-shifting between TTL level and standard RS-232 signal levels for up to 4 serial I/O ports.

60 LINE PARALLEL I/O CARD (S -60P).......$398.00
The GMX SBC-60P uses three 68230 Parallel Interface/Timers (PI/Ts) to provide up to forty-eight parallel I/O lines. The I/O lines are buffered in six groups of eight lines each, with separate buffer direction control for each group. Buffer direction can be fixed by hardware jumpers, or can be software programmable for bidirectional applications.

PROTOTYPING BOARD (SBC-WW)................$75.00
The SBC-WW provides a means of developing and testing custom I/O interface designs for the GMX Micro-20 68020 Single-board Computer. The board provides areas for both DIP (D al Inline Package) and PGA (Pin Grid Array) devices, and a pre-wired memory area for up to 512K bytes of dynamic RAM.

I/O BUS ADAPTER (SBC-BA)..................$195.00
The SBC-BA provides an interface between the GMX Micro-20 68020 Single-board Computer and the Motorola Input/Output Channel (I/O bus). With the I/O bus, up to sixteen off-the-shelf or custom peripheral devices (I/O modules) can be connected to the GMX Micro-20.

ARCNET LAN board w/o Software (SBC-AN)....$475.00
The SBC-AN provides an interface between the GMX Micro-20 68020 Single-board Computer and the ARCNET modified token-passing Local Area Network (LAN) originally developed by Datapoint orp. The ARCNET is a baseband network with a data transmission rate of 2.5 Megabits/second. The standard transmission media is a single 93 ohm RG-62/U coaxial cable. Fiber optic versions are available as an option.

OS9 LAN Software Drivers for SBC-AN.......$ 120.00

GMX MICRO 20 SOFTWARE

020 BUG UPDATE - PROMS & MANUAL..........$ 150.00

THESE 68020 OPERATING SYSTEMS ARE PRICED WHEN PURCHASED WITH THE MICRO-20. PLEASE ADD $150.00 IF PURCHASED LATER FOR THE UPDATED PROMS AND MANUALS.

OS9

OS9/68020 PROFESSIONAL PAK................$ 850.00
Includes O.S. "C" EDITOR, ASSEMBLER, DEBUGGER, development utilities, 68881 support.
Other Software for OS-9/68020
BASIC (included in PERSONAL PAK)..........$ 200.00
C COMPILER (included in PROFESSIONAL PAK).$ 500.00
PASCAL COMPILER...........................$ 500.00

UNIFLEX

UniFLEX (when ordered with Board).........$ 450.00
UniFLEX WITH REAL-TIME ENHANCEMENTS.......$1000.00
Other Software for UniFLEX
UniFLEX BASIC W/PRE COMPILER..............$ 300.00
UniFLEX C COMPILER........................$ 350.00
UniFLEX COBOL COMPILER....................$ 750.00
UniFLEX SCREEN EDITOR.....................$ 150.00
UniFLEX TEXT PROCESSOR....................$ 200.00
UniFLEX SORT/MERGE PACKAGE................$ 200.00
UniFLEX VSAM MODULE.......................$ 100.00
UniFLEX UTILITIES PACKAGE I...............$ 200.00
UniFLEX PARTIAL SOURCE LICENSE............$1000.00

GMX EXCLUSIVE VERSIONS,CUSTOMIZED FOR THE MICRO-20, OF THE BELOW LANGUAGES AND SOFTWARE ARE ALSO AVAILABLE FROM GMX.

BSOFT FORTRAN (UniFLEX)..................$1500.00

SCULPTOR (specify UniFLEX or OS9).........$ 995.00

FORTH (OS9)...............................$ 595.00

DYNACALC (specify UniFLEX or OS9).........$ 300.00

GMX DOES NOT GUARANTEE PERFORMANCE OF ANY GMX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURES PRODUCT.

ALL PRICES ARE F.O.B. CHICAGO

GMX, Inc. reserves the right to change pricing, terms, and products specifications at any time without further notice.

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 under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account number 73-32033.

CONTACT GMX FOR MORE INFORMATION ON THE ABOVE PRODUCTS

GMX STILL SELLS OIMIX SS0 BUS SYSTEMS, BOARDS & PARTS. CONTACT GMX FOR COMPLETE PRICE LIST.

A Member of the CPI Family

68 Micro Journal

10 Years of Dedication to Motorola CPU Users

6800 6809 68000 68010 68020

The Originator of "DeskTop Publishing™"

Publisher
Don Williams Sr.

Executive Editor
Larry Williams

Production Manager
Tom Williams

Office Manager
Joyce Williams

Subscriptions
Kristi Hart

Contributing & Associate Editors

Ron Anderson
Ron Voights
Doug Lurie
Dr. E.M. "Bud" Pass
Art Weller
Dr. Theo Elbert
& Hundreds More of Us

Contents

68 MICRO JOURNAL

"Contribute Nothing - Expect Nothing" DMW 1986

COMPUTER PUBLISHING, INC.

"Over a Decade of Service"

"World Wide"

68 MICRO JOURNAL
Computer Publishing Center
5900 Cassandra Smith Road
PO Box 849
Hixson, TN 37343
Phone (615) 842-4600 Telex 510 600-6630

68 Micro Journal is the *original* "DeskTop Publishing" product and has continously published since 1978 using only micro-computers and special "DeskTop" software. Using first a kit built 6800 micro-computer, a modified "ball" typewriter, and "home grown" DeskTop Publishing software. None was commercially available at that time. For over 10 years we have been doing "DeskTop Publishing"! *We originated what has become traditional "DeskTop Publishing"!* Today 68 Micro Journal is acknowledged as the "Grandfather" of "DeskTop Publishing" technology.

68 Micro Journal is published 12 t mes a year by Computer Publishing Inc. Second Class Postage pa d ISSN 0194-5025 at Hixson, TN. and additional entries. Postmaster: send form 3597 to 68 Micro Journal, POB 849, Hixson, TN 37343.

Subscription Rates

1 Year $24.50 USA, Canada & Mexico $34.00 a year. Others add $12.00 a year surface, $48.00 a year Airmail, USA funds. 2 years $42.50, 3 years $64.50 plus additional postage for each additional year.

Items or Articles for Publication

Articles submitted for publication must include authors name, address, telephone number, date and a statement that the material is original and the property of the author. Articles submitted should be on diskette, OS-9, SK*DOS, FLEX, Macintosh or MS-DOS. All printed items should be dark type and satisfactory for photo-reproduction. No blue ink! No hand written articles - please! Diagrams o.k.

Please - do not format with spaces any text indents, charts, etc. (source listing o.k.). We will edit in all formatting. Text should fall flush left and use a carriage return only to indicate a paragraph end. Please write for free authors guide.

Letters & Advertising Copy

Letters to the Editor should be the original copy, signed! Letters of grip as well as praise are acceptable. *We reserve the right to reject any letter or advertising material, for any reason we deem advisable.* Advertising Rates: Commercial please contact 68 Micro Journal Advertising Department. Classified advertising must be non-commercial. Minimum of $15.50 for first 15 words. Add $.60 per word thereafter. No classifieds accepted by telephone.

EXCITING SOFTWARE FROM THE LEADER...

microware

OS-9 ELECTRONIC MAIL

Flash your message on Electronic Mail Mail is a screen- or line-oriented program that runs on your OS-9/680X0 systems or over OS-9/NET. You can use distributed mailing lists or consecutive mailing list to get your message delivered. And received mail can be sent directly to your printer for immediate printout, spooled on a multiuser system or saved to a file. Mail features on-line help and complete, easy-to-understand documentation.
Electronic Mail $150.00.

PRINT SPOOLER

Spool it and Print it! Someone beat you to the printer? Don't blow your top while you cool your heels--get the OS-9/68000 Print Spooler and relax. The full-featured Print Spooler automatically routes and monitors the status of your devices and the output files to be spooled. Now you can have a complete print spooling management system at an affordable price.
OS-9 Print Spooler $150.00

FORTRAN

Crunch It! with Our New FORTRAN 77 Compiler Now you have a powerful new tool to take full advantage of the 68000 family of microprocessors. With Microware's FORTRAN 77 Compiler you can generate code that uses system-wide modules instead of linking redundant copies of the standard library to each program. Result: less memory, less disk space, faster loading and external updating!
FORTRAN 77 Compiler $750.00

OS-9/ST

NEW for your Atari ST! Now you can have the power of OS-9 on your Atari 520 or 1040 ST. A true multi-tasking environment for professional real-time results. OS-9/ST is available in two configurations: Personal and Professional. Choose either version for true multi-user support...And all at a price that puts UNIX to shame.

Personal OS-9/ST combines the power of OS-9 with an interactive, structured Basic. **$150.00**

Professional OS-9/ST has a powerful Assembler, Linker and User Debugger and the tools to turn your Atari ST into a full C Language workstation. **$600.00**

OS-9/68020 C COMPILER

NEW "speed demon" C Compiler! Now you can get your hands on a highly optimized C Language power tool--the OS-9/68020 C Compiler. When coupled with the MC68881 math co-processor, this compiler will let you'll blast through complex math functions in the blink of an eye. All compiler/assembler/linker options are controlled by an intelligent compiler executive that spares you from memorizing compiler options and module-calling sequences. And the compiler includes library functions for memory management and system events, and much, much more! The new OS-9/68020 C Compiler is included with the Professional OS-9/68020 System Software Package.
New C Language Compiler $750.00

To order these exciting NEW products or for more information... ***CALL TODAY!***

Microware Systems Corporation
1900 N.W. 114th Street • Des Moines, Iowa 50322
Phone 515-224-1929 • Telex 910-520-2535

West Coast Office
4401 Great American Parkway • Suite 220
Santa Clara, California 95054

Microware Japan. Ltd.
41-19 Honcho 4 Chome, Funabashi City • Chiba 273,
Japan • Phone 0473 (28) 4493 • Telex 781-299-3122

Micromaster Scandinavian AB
S:t Persgatan 7
Box 1309
S751-43 Uppsala
Sweden
Phone: 018-138595
Telex: 76129

Dr. Rudolf Keil, GmbH
Porphystrasse 15
D-6905 Schriesheim
West Germany
Phone: (0 62 03) 67 41
Telex: 465025

Elsoft AG
Zelweg 12
CH-5405 Baden-Dattwil
Switzerland
Phone: (056) 83-3377
Telex: 828275

Vivaway Ltd.
36-38 John Street
Luton, Bedfordshire, LU1 2JE
United Kingdom
Phone: (0582) 423425
Telex: 825115

Microprocessor Consultants
92 Bynya Road
Palm Beach 2108
NSW Australia
Phone: 02-919-4917

Microdata Soft
97 bis, rue de Colombes
92400 Courbevoie
France
Phone: 1-768-80-80
Telex: 615405

Microware is on the move. We have openings for Technical and Marketing Professionals.
Send your resume (in confidence) today and find out more about these exciting opportunities.

OS-9 and BASIC09 are trademarks of Microware and Motorola. UNIX is a trademark of Bell Laboratories, Inc.

MUSTANG -020 Super SBC ™

DATA-COMP proudly presents the first Under $5000 "SUPER MICRO".

The MUSTANG-020™

CP I
Mustang-020™
A DATA-COMP
Hi-Speed
Product

MUSTANG-020™

The MUSTANG-020 68020 SBC provides a powerful, compact, 32 bit computer system featuring the "state of the art" Motorola 68020 "super" micro-processor. It comes standard with 2 megabyte of high-speed SIP dynamic RAM, serial and parallel ports, floppy disk controller, a SASI hard disk interface for intelligent hard disk controllers and a battery backed-up time-of-day clock. Provisions are made for the super powerful Motorola MC68881 floating point math co-processor, for heavy math and number crunching applications. An optional network interface uses one serial (four (4) standard, expandable to 20) as a 125/bit per second network channel. Supports as many as 32 nodes.

The MUSTANG-020 is ideally suited to a wide variety of applications. It provides a cost effective alternative to the other MC68020 systems now available. It is an excellent introductory tool to the world of hi-power, hi-speed new generation "super micros". In practical applications it has numerous applications, ranging from scientific to education. It is already being used by government agencies, labs, universities, business and practically every other critical applications center, worldwide, where true multi-user, multi-tasking needs exist. The MUSTANG-020 is UNIX C level V compatible. Where low cost and power is a must, the MUSTANG-020 is the answer, as many have discovered. Proving that price is not the standard for quality!

As a software development station, a general purpose scientific or small to medium business computer, or a super efficient real-time controller in process control, the MUSTANG-020 is the cost effective choice. With the optional MC68881 floating point math co-processor installed, it has the capability of systems costing many times over it's total acquisition cost.

With the DATA-COMP "total package", consisting of a heavy duty metal cabinet, switching power supply with rf/line by-passing, 5 inch DS/DD 80 track floppy, Xebec hard disk controller, 25 megabyte winchester hard disk, four serial RS-232 ports and a UNIX C level V compatible multi-tasking, multi-user operating system, the price is under $5000, w/12.5 megahertz system clock (limited time offer). Most all popula high level languages are available at very reasonable cost. The system is expandable to 20 serial ports, at a cost of less than $65 per port, in multiples of 8 port expansion options.

The system SBC fully populated, quality tested, with 4 serial ports pre-wired and board mounted is available for less that $3000. Quantity discounts are available for OEM and special applications, in quantity. All that is required to bring to complete "system" standards is a cabinet, power supply, disks and operating system. All these are available as separate items from DATA-COMP.

A special version of the Motorola 020-BUG is installed on each board. 020-BUG is a ROM based bebugger package with facilities for downloading and executing user programs from a host system. It includes commands for display and modification of memory, breakpoint capabilities, a powerful assembler/disassemble and numerous system diagnostics. Various 020-BUG system routines, such as I/O handlers are available for user programs.

Normal system speed is 3-4.5 MIPS, with burst up to 10 MIPS, at 16.6 megahertz. Intelligent I/O available for some operating systems.

Hands-on "actual experience sessions", before you buy, are available from DATA-COMP. Call or write for additional information or pricing.

DATA-COMP
Installed Systems World-Wide
OVER 10 YEARS OF DEDICATED QUALITY!

CP I
A [illegible]
[illegible]
[illegible]
Hixson, Tn 37343
Telephone 615 842-4600
Telex [illegible]

Mustang-020 Mustang-08 Benchmarks

```
                                                   32 bit     Register
                                                   Integer    Long

IBM AT 7300 Xenix Sys 3                              9.7
AT&T 7300 UNIX PC 68010                              7.2        4.3
DEC VAX 11/780 UNIX Berkley 4.2                      3.6        3.2
DEC VAX 11/750       "        "                      5.1        3.2
68008 OS-9 68K 8 Mhz                                18.0        9.0
68000 OS-9 68K 10 Mhz                                6.5        4.0
MUSTANG-08 68008 OS-9 68K 10 Mhz                     9.8        6.3
MUSTANG-020 68020 OS-9 68K 16 Mhz                    2.2        0.88
MUSTANG-020 68020 MC68881 UniFLEX 16 Mhz             1.8        1.22

Main()
    {
            register long i;
            for (i=0; i < 999999: ++i);
    }
Estimated MIPS - MUSTANG-020 ..... 4.5 MIPS,
Burst to 8 - 10 MIPS: Motorola Specs
```

12.5 Mhz (optional 16.6 Mhz available) MC68020 full 32-bit wide path
32-bit wide data and address buses, non-multiplexed
on chip instruction cache
object code compatible with all 68XXX family processors
enhanced instruction set - math co-processor interface
68881 math hi-speed floating point co-processor (optional)
direct extension of full 68020 instruction set
full support IEEE P754, draft 10.0
transcendental and other scientific math functions
2 Megabyte of SIP RAM (512 x 32 bit organization)
up to 256K bytes of EPROM (64 x 32 bits)
4 Asynchronous serial I/O ports standard
optional to 20 serial ports
standard RS-232 interface
optional network interface
buffered 8 bit parallel port (1/2 MC68230)
Centronics type pinout
expansion connector for I/O devices
16 bit data path
256 byte address space
2 interrupt inputs
clock and control signals
Motorola I/O Channel Modules
time of day clock/calendar w/battery backup
controller for 2, 5 1/4" floppy disk drives
single or double side, single or double density
35 to 80 track selectable (48-96 TPI)
SASI interface
programmable periodic interrupt generator
interrupt rate from micro-seconds to seconds
highly accurate time base (5 PPM)
5 bit sense switch, readable by the CPU
Hardware single-step capability

Mustang Hi-Speed Systems
Only from Data-Comp Div.
68008-68030

Don't be mislead! ONLY Data-Comp delivers the Super MUSTANG-020

These *hi-speed* 68020 systems are presently working at *NASA, Atomic Energy Commission, Government Agencies as well as Universities, Business, Labs, and other Critical Applications Centers, worldwide, where speed, math crunching and multi-user, multi-tasking UNIX C level V compatability and low cost is a must!*

OS-9

OS-9 Professional Ver	$850.00
*Includes C Compiler	
Basic09	300.00
C Compiler	500.00
68000 Disassembler (w/source add: $100.00)	100.00
Fortran 77	750.00
Microware Pascal	500.00
Omegasoft Pascal	900.00
Stylo-Graph	495.00
Stylo-Spell	195.00
Stylo-Merge	175.00
Stylo-Graph-Spell-Merge	695.00
PAT w/C source	229.00
JUST w/C source	79.95
PAT/JUST Combo	249.50
Sculptor+ (see below)	995.00
COM	125.00

UniFLEX

UniFLEX (68020 ver)	$450.00
Screen Editor	150.00
Sort-Merge	200.00
BASIC/PreCompiler	300.00
C Compiler	350.00
COBOL	750.00
CMODEM w/source	100.00
TMODEM w/source	100.00
X-TALK (see Ad)	99.95
Cross Assembler	50.00
Fortran 77	450.00
Sculptor+ (see below)	995.00

Standard MUSTANG-020™ shipped 12.5 Mhz.	
Add for 16.6 Mhz 68020	375.00
Add for 16.6 Mhz 68881	375.00
Add for 20 Mhz 68020/RAM	750.00
16 Port exp. RS-232 requires 1 or 2 Adapter Cards below.	335.00
RS232 Adapter Each card supports 4 additional ser. ports (total of 36 serial ports supported)	165.00
60 line Parallel I/O card Uses 3 68230 Interface/Timer chips, 6 groups of 8 lines each, separate buffer direction control for each group.	398.00
Prototype Board areas for both dip and PGA devices & a pre-wired memory area up to 512K DRAM.	75.00
SBC-AN Interface between the system and ARCNET modified token-passing LAN, fiber optics optional - call.	475.00
LAN software drivers	120.00
Expansion for Motorola I/O Channel Modules	$195.00

Special for complete MUSTANG-020™ system buyers - Sculptor+ $695.00. SAVE $300.00

Software Discounts

All MUSTANG-020™ system and board buyers are entitled to discounts on all listed software: 10-70% depending on item. Call or write for quotes. Discounts apply after the sale as well.

THE PRO!

Only the "PRO" version of OS-9 supported!

This is *HEAVY DUTY* Country!

UPGRADES
Write or Call for Professional OS-9 "Full Bore" Upgrade Kit

For a limited time we will offer a $400 trade-in on your old 68XXX SBC. Must be working properly and complete with all software, cables and documentation. Call for more information.

MUSTANG-020 SBC		$2490.00
Cabinet w/switching PS		$299.95
5" 80 track floppy	DS/DD	$269.95
Floppy cable		$39.95
OS-9 68K Professional Ver.		$850.00
* Includes C Compiler ($500.00)		
Winchester cable		$39.95
Winchester Drive 25 Mbyte		$895.00
Hard Disk controller		$395.00
Shipping USA UPS		$20.00
Total:	Save $1000.00	$5,299.80

complete sytem $4,299.80

UniFLEX	Less	$100.00
MC68881 f/p math processor	Add	$275.00
16.67 Mhz MC68020		$375.00
16.67 Mhz MC68881		$375.00
20 Mhz MC68020 Sys		$750.00

Note all 68881 chips work with 20 Mhz Sys

NOTE: Only Professional OS-9 now available (68020 Version) Includes ($500.00) C Compiler 68020 & 68881 supported

complete
25 Mbyte HD System
$4299.80
85 Mbyte HD System
$5748.80

Data-Comp Division
A Decade of Quality Service
Systems World-Wide
Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 · Telex 510 600-6630 Hixson, Tn 37343

PAT - JUST

PAT

With 'C' Source

$229.00

All OS-9
68XXX
Systems

service For Over a Decade!
1,200 Titles +

A Division of
CPI

S.E. Media
5900 Cassandra Smith Rd.
Hixson, Tn 37343
Telephone 615 842-6809
Telex 510 600-6630

[illegible]10-48020

PAT FROM S. E. MEDIA -- A FULL FEATURED SCREEN ORIENTED TEXT EDITOR with all the best of PIE. For those who swore by and loved PIE, this is for **YOU!** All PIE **features & much more!** Too many features to list. And if you don't like ours, change or add your own. **C source included.** Easily configured to your CRT terminal, with special configuration section. No sweat!

68008 - 68000 - 68010 - 68020 OS-9 68K $229.00

COMBO

PAT/JUST

Special $249.00

JUST

JUST from S. E. MEDIA - - Text formatter written by Ron Anderson; for dot matrix printers, provides many unique features. Output formatted to the display. User configurable for adapting to other printers. Comes set-up for Epson MX80 with Graflex. Up to 10 imbedded printer control commands. Compensates for double width printing. Includes normal line width, page numbering, margin, indent, paragraph, space, vertical skip lines, page length, centering, fill, justification, etc. Use with **PAT** or any other text editor. The ONLY stand alone text processor for the 68XXX OS-9 68K, that we have seen. And at a very **LOW PRICE!** Order from: S.E. MEDIA - see catalog this issue.

68008 - 68000 - 68010 - 68020 OS-9 68K
With 'C' source $79.95

An Ace of a System in Spades! The New

MUSTANG-08/A™

Now with 4 serial ports standard & speed increase to 12 Mhz CPU + on board battery backup and includes the PROFESSIONAL OS-9 package - including the $500.00 OS-9 C compiler! This offer won't last forever!

NOT 128K, *NOT* 512K *FULL* 768K No Wait RAM

The **MUSTANG-08**™ system took every hand from all other 68008 sysems we tested, running OS-9 68K!

The **MUSTANG-08** includes OS9-68K™ and/or Peter Stark's SK*DOS. SK*DOS is a single user, single tasking system that takes up where *FLEX™ left off. SK*DOS is actually a 68XXX FLEX type system *(Not a TSC product.)*

SALE!!

The OS-9 68K system is a full blown multi-user, multi-tasking 68XXX system. All the popular 68000 OS-9 software runs. It is a *speed whiz on disk I/O*. Fact is: the MUSTANG-08 is faster on disk access than some other 68XXX systems are on memory cache access. Now, that is fast! And that is just a small part of the story! **See benchmarks.**

System includes OS-9 68K or SK*DOS - Your Choice

Specifications:

CPU	MC68008	12 Mhz
RAM	768K	256K Chips
	No Wait States	
PORTS	4 - RS232	MC68681 DUART
	2 - 8 bit Parallel	MC6821 PIA
CLOCK	MK48T02	Real Time Clock Bat. B/U
EPROM	16K, 32K or 64K	Selectable
FLOPPY	WD1772	5 1/4 Drives
HARD DISK	Interface Port	WD1002 Board

Now more serial ports - faster CPU Battery B/U - and $850.00 OS-9 Professional with C compiler included!

***$400.00**

See Mustang-02 Ad - page 5 for trade-in details

Mustang Hi-Speed Systems
Only from Data-Comp Div.
68008-68030

MUSTANG-08

LOOK

```
                               Seconds   32 bit    Register
                                         Integer   Long
Other      68008  8  Mhz  OS-9  68K...18.0...9.0
MUSTANG-08  10  Mhz  OS-9  68K....9.8...6.3
           Main()
              {
                        /* Int I; */
C Benchmark Loop        register long I;
                        for (I=0; I < 999999; ++I);
              }
```

Now even faster! with 12 Mhz CPU

C Compile times: OS-9 68K Hard Disk	
MUSTANG-08 8 Mhz CPU	0 min - 32 sec
Other popular 68008 system	1 min - 05 sec
MUSTANG-020	0 min - 21 sec

25 Megabyte Hard Disk System

$1,998.90

Complete with PROFESSIONAL OS-9 includes the $500.00 C compiler, PC style cabinet, heavy duty power supply, 5" DDDS 80 track floppy - Ready to Run

Unlike other 68008 systems there are several significant differences. The MUSTANG-08 is a full 12 Megahertz system. The RAM uses NO wait states, this means full bore MUSTANG type performance.

Also, allowing for addressable ROM/PROM the RAM is the maximum allowed for a 68008. The 68008 can only address a total of 1 Megabytes of RAM. The design allows all the RAM space (for all practical purposes) to be utilized. What is not available to the user is required and reserved for the system.

A RAM disk of 480K can be easily configured, leaving 288K free for program/system RAM space. The RAM DISK can be configured to any size your application requires (system must have 128K in addition to its other requirements). Leaving the remainder of the original 768K for program use. Sufficient source included (drivers, etc.)

FLEX is a trademark of TSC MUSTANG-08 is a trademark of CPI

Data-Comp Division

A Decade of Quality Service™

Systems World-Wide

Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

* Those with SWTPC Hi-density FLEX 5' - Call for special info.

The C Programmers Reference Source. Always Right On Target!

C User Notes

By: Dr. E. M. 'Bud' Pass
1454 Latta Lane N.W.
Conyers, GA 30207
404 483-1717/4570
Computer Systems Consultants

A Tutorial Series

INTRODUCTION

This chapter continues the discussion of the conversion of Technical Systems Consultants BASIC and Microware BASIC09 programs into C programs begun in the previous chapter.

CONVERTING BASIC PROGRAMS TO C

Expressions

The table below presents a list of the TSC BASIC operators, in decreasing hierarchical order:

```
( )     parentheses
 fun()   functions
 ^       exponenation
 -  +    unary negative/positive
 *  /    multiplication/division
 +  -    addition/subtraction
 +       string concatenation
  <=    >=    <>    <    >    =    relational
         comparisons
 NOT     logical complement
```

The table below presents a list of the BASIC09 operators, in decreasing hierarchical order:

```
( )      parentheses
 fun()   functions
  -   +   NOT   unary negative/positive,
         logical complement
 ^   **  exponenation
 *   /   multiplication/division
 +   -   addition/subtraction
  <=   =<   >=   =>   <>   ><   <   >   =
         relational comparisons
 AND     logical conjunction
 OR      logical disjunction
 XOR     logical exclusive disjunction
```

The table below presents a list of the standard (K & R) set of C language operators, in decreasing hierarchical order:

```
( ) [ ] -> .  parentheses, brackets, structure
              designations
!  ~  +  ++  -  --  (type)  *  &  sizeof()
      unary operators
 *  /  % multiplication, division, remainder
 +  -       addition, subtraction
 <<  >>      shift
 <  <=  >  >= relational operators
 ==  !=       relational operators
 &            logical conjuction
 ^            logical exclusive disjunction
 |            logical disjunction
 &&          boolean conjuction
 ||          boolean disjunction
 ? :         ternary
 =  +=  -=  *=  /=  %=  >>=  <<=  &=  ^=  |=
             assignment
 ,            left to right evaluation
```

TSC BASIC and BASIC09 allow Boolean expressions to be used in arithmetic contexts, returning non-zero for TRUE and zero for FALSE. All implement logical operators as bitwise, rather than TRUE/FALSE. They have somewhat different operator hierarchies. They interpret a binary '+' operator in a string context to represent concatenation. All evaluate expressions involving operators of equal precedence on a left to right basis, except for those involving exponentiation, which are evaluated right to left.

The differences in arithmetic operator hierarchies, as indicated in the tables above, may cause subtle problems in terms of incorrectly calculated C expressions which are correctly calculated by BASIC. Such problems are minor in most programs and may always be solved by manually forcing expression evaluation order with parentheses, although the required modifications may be tedious and error-prone. Such changes should be performed before conversion, not afterward, whenever possible, to be able to ensure that the program is correct before the changes represented by the conversion are attempted.

Essentially any expressions involving BASIC string variables will require modifications in the C program which cannot be performed until after the low-level conversions are performed, since they have no corresponding BASIC statements and expressions.

Strings

An area of potential major conversion problems concerns the differences in the handling of strings. TSC BASIC supports dynamic allocation of strings of arbitrary content and length of zero to 32767 bytes. The current length and pointer to the string are explicitly stored, associated with the string variable, vector, or matrix. BASIC09 supports a fixed allocation of strings of length zero to 65535 bytes. The contents are not arbitrary, since the hex-ff character code represents the end of the string. The maximum length and pointer are explicitly stored, but the current length is only implicitly determined as the length of the string up to but not including the hex-ff, or the maximum length, whichever is less.

C normally provides a fixed length allocation of strings, with the maximum length implicitly determined by a hex-00 character code. This may have an especially serious effect on the logic of programs using large string arrays, since BASIC09 and C programs normally allocate the maximum length for each string in the array, whereas TSC BASIC allocates only the necessary length for each string, and performs automatic allocation and deallocation for the programmer.

One solution would be to recode every reference to a BASIC string as a C function call. This would provide the same string format and string processing environment available in the original BASIC program. However, since the string format and string processing environment of the C compiler and libraries will not agree with the user's, additional overhead would be introduced in translating formats between the two sets of functions.

Environment-Dependent Operations

BASIC09 supports the PEEK and ADDR functions and the POKE statement. TSC BASIC supports the DPEEK, PEEK, and PTR functions and the DPOKE and POKE statements. These operations are all highly dependent upon their environment and should not be expected to be easily converted to any other language or environment.

If they are to be maintained at all, the TSC BASIC functions DPEEK and DPOKE should be changed to PEEK and POKE functions before the conversion process is attempted. DPEEK(n) is the same as the following:

((PEEK(n) << 8) OR PEEK(n + 1))

and DPOKE n,m is the same as the following:

```
POKE n,(m >> 8): POKE (n + 1),m AND 255
```

The uses of the (D)PEEK functions and (D)POKE statements are very dependent upon the hardware and software configuration under which the program is expected to run. They may be used to inspect and modify locations in a BASIC interpreter to modify its internal operations; such uses are obviously incorrect when using a C compiler. Another common use of PEEK and POKE is to perform I/O operations on memory-mapped I/O locations; such code may or may not perform the expected operations in another environment on the same type of computer and almost certainly will fail on another type of computer.

The PTR function is used to obtain the address of a string descriptor or a variable, and the ADDR function is used to obtain the address of a string or variable. The length and format of a BASIC floating point or integer number are different from the length and format of a C floating point or integer number. Thus, any BASIC program logic involving the ADDR or PTR function probably must be rewritten during the conversion of the program to the C language.

Input/Output

I/O file numbers are used in similar manners in TSC BASIC and in BASIC09, although there are several differences in interpretation. They are used in the OPEN, CLOSE, INPUT, PRINT, GET, PUT, FIELD, and DIM statements and in the INCH$ function in TSC BASIC to designate the logical channel from which data is to be read or to which data is to be written.

A difference between the versions of BASIC concerns file number zero. TSC BASIC interprets file number zero to be the user's terminal, unless the file is opened for output, in which case it is interpreted to be a printer, or unless the file is opened for input, in which case the input prompts to the terminal are deleted.

BASIC09 interprets file number zero to be the standard input file (normally the user's keyboard), file number one to be the standard output file (normally the user's screen), and file number two to be the standard error output file (normally the user's screen). These files need not be opened; however, they may be closed and redirected to another device, if required. This usage is consistent with the use of low-level file descriptors in the C language.

The OPEN statement in TSC BASIC accepts as a file number a constant or a variable containing a value. The OPEN and CREATE statements in BASIC09 place a file number into a variable. Thus, all BASIC09 file numbers used in statements and functions should be variables, unless they reference file numbers zero, one, or two, in which they may be constants or variables.

File-Naming

A difference between the systems discussed here concerns file-naming conventions. File naming rules are generally more dependent upon operating system requirements than upon BASIC conventions, and may cause problems in conversion of programs from one operating system to another, even when using the same language.

TSC BASIC FLEX file specifiers reference disk files. They are composed of an optional drive number (0-3), a file name of 1 to 8 characters, and an optional suffix of 1 to 3 characters. The file name and suffix must start with a letter and may be composed of letters, digits, and certain special characters. The drive number, if present, is separated from the file name with a period. The suffix, if present, is also separated from the file name with a period. Letter case is significant. If drive number is omitted, the default work drive is assumed. If file number zero is opened for output, the file name is assumed to be a printer driver. Other than file number zero, this usage is compatible with C compilers under FLEX.

TSC BASIC UNIFLEX file specifiers reference any device. They are composed of an optional path name and an optional file name. The file name should start with a letter and may be composed of letters, digits, and certain special characters. Letter case is significant. Not both path name and file name may be omitted. If the path name is omitted, the current default directory is used. This usage is compatible with C compilers.

BASIC09 OS-9 file specifiers reference any device. They are composed of an optional path name and an optional file name. The file name should start with a letter and may be composed of letters, digits, and certain special characters. Letter case is not significant. Not both path name and file name may be omitted. If the path name is omitted, the current default data directory is used. This usage is compatible with C compilers.

This discussion is continued in the next chapter.

EXAMPLE C PROGRAM

Following is this month's example C program; it paginates a listing of a list of files.

```
#include <stdio.h>
#include <ctype.h>
#include <time.h>

FILE *input;
unsigned char **argt, ch, line[256], *p, date[135];
unsigned form[64][512], sesc[256][8];
long timex;
short int i, j, k, l, files, lines, pages, q, r, ff, dt, pg = 59;
short int asis, td, fl, trunc, recl = 78, escape, escflg[128];

main(argc,argv)
int argc;
unsignod char **argv;
{
        for (argt = argv; *(p = *(argt + 1)) == '-'; )
        {
                ++files;
                ++argt;
                while (*++p)
                {
                        switch(*p)
                        {
                        case 'A':
                        case 'a':
                                ++asis;
                                break;
                        case 'D':
                        case 'd':
                                ++dt;
                                break;
                        case 'F':
                        case 'f':
                                ++ff;
                                break;
                        case 'E':
                        case 'e':
                                recl = 93;
                                break;
                        case 'P':
                        case 'p':
                                pg = 80;
                                break;
                        case 'S':
                        case 's':
                                recl = 79;
                                break;
                        case 'T':
                        case 't':
                                ++trunc;
                                break;
                        case 'W':
                        case 'w':
                                recl = 131;
                                break;
                        case 'Z':
                        case 'z':
                                if (i = *++p & 0x7f)
                                {
```

```
                                for (++escape, j = k = 0; *++p; )
                                {
                                    if (isdigit(*p))
                                        k = (k << 4) | (*p & 0x0f);
                                    else
                                    if ((*p >= 'A') && (*p <= 'F'))
                                        k = (k << 4) + (*p - 'A' + 10);
                                    else
                                    if ((*p >= 'a') && (*p <= 'f'))
                                        k = (k << 4) + (*p - 'a' + 10);
                                    else
                                    if (*p == '~')
                                    {
                                        i |= 128;

                                    j = 0;
                                    continue;
                                }
                                else
                                    break;
                                if (j & 1)
                                {
                                    if (j < 16)
                                        sesc[i][j >> 1] = k;
                                    k = 0;
                                }
                                ++j;
                            }
                            --p;
                            break;
                        }
                default:
                        printf(
                            "\n%s formats input files to output\n\n",
                            argv[0]);
                        printf("Usage: %s [-options] [filenames]\n",
                            argv[0]);
                        printf(" -a output files as input\n");
                        printf(" -d add page heading\n");
                        printf(" -e fold page at 94 characters\n");
                        printf(" -f suppress final form-feed\n");
                        printf(" -p page at 80 lines (default 60)\n");
                        printf(" -t truncate, do not fold lines\n");
                        printf(" -s fold page at 80 characters\n");
                        printf(" -w fold page at 132 characters\n");
                        printf(
                            " -zxssss-tttt set escape sequences\n");
                        printf("    for |x ... |x text template\n");
                        printf("    as hex values ssss and tttt\n");
                        printf("    for starting and terminating\n");
                        printf("    codes (each < 8 characters)\n\n");
                        exit(1);
                }
        }
}

for (ch = 0; ++files < argc; )
{
        if(!(input = fopen(*++argt,"r")))
        {
                fprintf(stderr,"page: can't open %s\n",*argt);
                continue;
        }
        date[0] = 0x20;
        strcpy(date + 1,*argt);
        for (q = strlen(date); q < 45; ++q)
                date[q] = 0x20;
        timex = time((long *) 0);
        strcpy(&date[q],ctime(&timex));
        date[strlen(date) - 1] = 0x00;
        pages = 0;
        lines = 999;
        td = dt;
        fl = 0;
        ch = '\f';
        putc('\n',stdout);
        while (fgets(line,255,input))
        {
                r = expand_line();
                if (trunc)
                        r = 1;
                if (line[0] == '\f')
                        fl = 1;
                if (asis)
                {
                        fputx(form[1],stdout);
                        continue;
                }
                if ((!fl) && ((lines += r) > pg))
                        {
                                if (lines < 999)
                                        fputx("\f\n",stdout);
                                if (td)
                                {
                                        fprintf(stdout,"%s %d\n\n",
                                                date,++pages);
                                        lines = 2;
                                }
                                else
                                        lines = 0;
                        }
                        for (q = 1; q <= r; ++q)
                                fputx(form[q],stdout);
                }
                if ((ch != '\f') && (!ff))
                        putc('\f',stdout);
                fclose(input);
        }
}

expand_line()
{
        unsigned char *p, *q, *r, m;
        short int i, j, k, l = 0;

        if (((j = line[i = strlen(line) - 1]) == '\n') || (j == '\r'))
                line[i] = '\0';
        for (p = line, l = 1, j = k = 1, q = p + 1, m = 0; ; p = q, ++q)
        {
                switch (*p)
                {
                case '\0':
                        form[i][j++] = '\n';
                        form[i][j] = '\0';
                        return i;
        case '\r':
        case '\n':
                form[i][j++] = '\n';
                form[i++][j] = '\0';
                if (j = k = 1)
                        form[i][0] = 0x20;
                break;
        case '\t':
                if (*q == '\0')
                {
                        form[i][j++] = '\n';
                        form[i][j] = '\0';
                        return i;
                }
                do
                {
                        if (k++ >= recl)
                        {
                                form[i][j++] = '\n';
                                form[i++][j] = '\0';
                                if (j = k = 1)
                                        form[i][0] = 0x20;
                                break;
                        }
                        form[i][j++] = 0x20;
                }
                while (k & 0x07);
                break;
        case '|':
                if (asis || (!escape) || (!(m = *q)))
                        goto passthru;
                p = q++;
                if (m == '|')
                        goto passthru;
                if (!(escflg[m] = !escflg[m]))
                        m |= 128;
                for (r = seac[m]; *r; ++r)
                        form[i][j++] = *r;
                break;
                case '\b':
                        k -= 2;
                default:
passthru:
                        form[i][j++] = *p;
                        if (k++ >= recl)
                                if (*q && (*q != '\b'))
                                {
                                        form[i][j++] = '\n';
                                        form[i++][j] = '\0';
                                        if (j = k = 1)
                                                form[i][0] = 0x20;
                                }
                }
        }
}

fputx(s, iop)
register char *s;
register FILE *iop;
{
        register int c;

        while (c = *s++)
        {
                putc(c, iop);
                if ((c != '\n') && (c != '\r'))
                        ch = c;
        }
}
```

EOF

EXAMPLE C PROGRAM

This example C program provides an interactive file inquiry and modification facility. It should work with any C compiler supporting a linear FSEEK function. Depending upon the compiler, the MODER and MODEW defines may require specification to provide binary read and binary update capabilities.

```
/*
 *      filedit filename-list
 */

#include <stdio.h>
#include <ctype.h>

#ifndef MODER
#define MODER "r"
#endif
#ifndef MODEW
#define MODEW "r+"
#endif

FILE *input, *output;
char *p, *q, hold[256], loci[256];
long where, loc;
short int offs, c, i, j, l, m, n, s, w, v, x, z;

main(argc, argv)
short int argc;
char *argv[];
{
    printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
    printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
    if (argc < 2)
    {
        printf("Usage: %s [-n] file-list\n", argv[0]);
        printf(" interactively dumps files in file-list\n");
        printf("  -n option dumps files continuously\n");
        exit(0);
    }
    for (j = 1; j < argc; ++j)
        n += (*argv[j] == '-');
    for (j = 1; j < argc; ++j)
    {
        if (j < 1)
            j = 1;
        if (*argv[j] == '-')
            continue;
        if (!(input = fopen(argv[j], MODER)))
        {
            if (!n)
            {
                printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
                printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
            }
            printf("\nCannot find %s.", argv[j]);
            fflush(stdout);
            if ((!n) && (!fgets(hold, 256, stdin)))
                j = argc;
            continue;
        }
        for (where = 0L, v = -1; ; where += 256L)
        {
            printf("\n\n%s:", argv[j]);
            if (w)
            {
                printf("    \"");
                for (w = 0, x = (l < 16) ? l : 16; w < x; ++w)
                    if (m)
                        dspchr(loci[w]);
                    else
                        putchar(loci[w]);
                if (v >= 0)
                    printf("\" found at %lx", loc);
                else
                    printf("\" not found");
            }
            putchar('\n');
            dspint((int)((where >> 16) & 0xffffL));
            printf("| 00 01 02 03  04 05 06 07  ");
            printf("08 09 0a 0b  0c 0d 0e 0f  ");
            printf("0123456789abcdef\n");
            printf("----| -- -- -- --  -- -- -- --  ");
            printf("-- -- -- --  -- -- -- --  ");
            printf("----------------\n");
            for (w = offs = 0, x = 16, z = (where >> 8);
                offs < 256; offs += 16)
            {
                dspchr(z);
                dspchr(offs);
                putchar('|');
                *hold = 0;
                if ((x == 16) && (x = fread(hold, 1, 16, input)))
                {
```

```
                for (*((p = hold) + x) = i = 0, s = 53;
                    i < x; ++i, ++p)
                {
                    if (!(i & 0x03))
                    {
                        putchar(' ');
                        --s;
                    }
                    dspchr(c = *p & 0xff);
                    putchar(' ');
                    if ((c < 0x20) || (c > 0x7e))
                        *p = '.';
                    s -= 3;
                }
                while (s--)
                    putchar(' ');
            }
            printf("%s\n", hold);
        }
        if (n)
            if (x < 16)
                break;
            else
                continue;
        printf("\nb=back, c=curr file, n=next file, ");
        printf(p=prev file, q=quit,\n");
        printf("a[xxxx]=find alpha, x[hhhhhh]=find hex, ");
        printf("m aa vv..vv=modify\n");
        fflush(stdout);
        if (!fgets(q = hold, 256, stdin))
            *hold = 'q';
        switch (c = (isupper(*q) ? tolower(*q) : *q))
        {
        case 'a':
            if (*++q >= ' ')
                for (l = m = 0, p = loci; (*q >= ' '); ++l)
                    *p++ = *q++;
            goto research;
        case 'x':
            if (*++q >= ' ')
                for (l = z = 0, p = loci, i = m = 1;
                    (*q >= ' '); ++q)
                {
                    if (*q == ' ')
                        continue;
                    z = hexval(*q, z);
                    if (i = 1 - i)
                    {
                        *p++ = z;
                        ++l;
                        z = 0;
                    }
                }
research:
                if (l)
                {
                    if (x < 16)
                        goto redisplay;
                    loc = where + (long)v + 1L;
                    v = -1;
reseek:
                    if (fseek(input, loc, 0))
                        goto redisplay;
                    for (z = *(p = loci) & 0xff; ; ++loc)
                    {
                        if ((c = getc(input)) == EOF)
                            goto redisplay;
                        if (c == z)
                            break;
                    }
                    for (i = 1; i < l; ++i)
                    {
                        if ((c = getc(input)) == EOF)
                            goto redisplay;
                        if (c != (z = *++p & 0xff))
                        {
                            ++loc;
                            goto reseek;
                        }
                    }
                    where = loc & 0xffffff00L;
                    v = loc & 0xffL;
                    v = 1;
                    where += 256L;
                    goto redo;
                }
redisplay:
                c = 'b';
redo:
                where += 256L;
                x = 0;
            case 'b':
                if (c == 'b')
                    v = -1;
                if (x < 16)
                {
                    fclose(input);
                    if (!fopen(argv[j], MODER))
                    {
                        j = argc;
                        break;
                    }
                    x = 16;
                }
                if (fseek(input, where -= (where ? 512L : 256L), 0))
                {
                    fclose(input);
                    if (!fopen(argv[j], MODER))
                    {
                        j = argc;
                        break;
                    }
                    where = 0L;
                }
                where -= 256L;
                break;
            case 'm':
                for (p = q = hold; *++p == ' '; );
                if ((!isdigit(*p)) && (!isalpha(*p)))
                    break;
                z = hexval(*p++, 0);
                if (*p > ' ')
                    z = hexval(*p++, z);
                while (*p >= ' ')
                {
                    while (*p == ' ')
                        ++p;
                    i = hexval(*p++, 0);
                    if (*p > ' ')
                        i = hexval(*p++, i);
                    *q++ = i;
                }
                if (!(i = q - hold))
                    break;
                if ((!(output = fopen(argv[j], MODEM))) ||
                    fseek(output, where + (long)z, 0) ||
                    (!fwrite(hold, 1, i, output)))
                    printf("could not change %s.\n", argv[j]);
                else
                {
                    printf("changed %d byte(s) starting ", i);
                    printf("at %lx in %s.\n",
                        where + (long)z, argv[j]);
                }
                if (output)
                    fclose(output);
                x = 0;
                    goto redisplay;
                case 'q':
                    j = 32767;
                case 'p':
                    --j;
                case 'c':
                    --j;
                case 'n':
                    x = 0;
                    if (c != 'c')
                        v = -1;
                    break;
                case '!':
                    system(hold + 1);
                    goto redisplay;
                }
                if (x < 16)
                    break;
            }
            fclose(input);
        }
        printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
        printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
        exit(0);
    }

    dspnyb(v)
    int v;
    {
        v &= 0x0f;
        putchar(((v += '0') > '9') ? (v + 0x27) : v);
    }

    dspchr(v)
    int v;
    {
        dspnyb(v >> 4);
        dspnyb(v);
    }

    dspint(v)
    int v;
    {
        dspchr(v >> 8);
        dspchr(v);
    }

    hexval(c, z)
    int c, z;
    {
        c &= 0xff;
        return ((z << 4) | (isdigit(c) ? (c - '0') :
            ((c >= 'a') && (c <= 'f')) ? (c - 'a' + 10) :
            ((c >= 'A') && (c <= 'F')) ? (c - 'A' + 10) : 0));
    }
```

Basically OS-9

Dedicated to the serious OS-9 user.
The fastest growing users group world-wide!

6809 - 68030

A Tutorial Series

By: Ron Voigts
2024 Baldwin Court
Glendale Heights, IL

CRACKING THE OS-9 SHELL

This month I was thinking, what should I write about? This question can sometimes be answered instantly. Occasionally, it takes thought. Many times I page through the manuals looking for an idea. Other times it comes to me like a flash. This month it hit me. Eggs! Now you think, he has gown off the deep end. The job has finally gotten to him. But there is something that OS-9 has in common with an egg. They are both surrounded by a shell.

There is a difference between the egg and OS-9. (Besides the obvious!) The egg's shell is there to protect it from the environment. The OS-9 shell is there to make it accessible to the environment. It prompts you for input. It executes command lines you enter. It allocates memory. It handles multi-tasking. It redirects input and output. It even has a number of built in commands. Without it, not much would be accomplished.

Many times I find myself referring to things that the shell does as OS-9. But in reality the shell is special program that reads and processes command lines. It brings the OS-9 system to our finger tips. The shell does this by simply printing the familiar OS-9 prompt.

OS9:

We input things like commands, programs and procedure files. With these we add parameters that we want to pass to the executing program. And we can append certain modifiers to be used by the shell.

The first thing entered after the prompt is the verb. .This is some command or program that we want to execute. It can be any of the standard OS-9 commands. Or it can be program we want to execute. The shell will run it as new process. It will then wait until the process is finished. It will print another prompt and wait for the next request. For such a useful program, it spends most of its time waiting .

The shell uses F$FORK to create a new process. A copy of the parameter area is made. The X register is pointing to the module name in the command line. The shell appends a carriage return character to the line. The Y register contains the parameter area size. And U has the beginning of the parameter area. A call is made to F$FORK and a new process is created. It is the "child" of the shell that created it. The shell or "parent" deactivates itself, going into a WAIT state.

To test this out, enter "SHELL" a few times to the OS-9 prompt. You should have at least three shell existing. The one you are in is a child of the previous shell or the "grandchild" of the the first one. (If you have entered the shell command more then twice, the lineage could go way back.) To see this enter,

OS9: PROCS

If you have a level 2 system something like the following will appear.

```
   Parnt User                            Mem Stack
ID  ID  Numbr Pty Age Sts Signl Siz  Ptr   Primary Module
--- --- ----- ---- --- --- ------ --- ----- ----------------
-
  2   1      0 128 129 $80     0   1 $87E2 SysGo
  3   2      0 128 129 $80     0   3 $85E2 Shell
  4   3      0 128 129 $80     0   3 $83E2 Shell
  5   4      0 128 129 $80     0   3 $81E2 Shell
  6   5      0 128 128 $80     0   6 $05F3 Procs
```

Notice the last shell in the list. It is process ID #5. Its parent is #4, the previous shell. ID #4's parent is #3, who also is the grandparent to # 5. Now level 1 users will have a listing that looks like this.

```
Usr #   id pty sta mem pri mod
----- --- --- --- --- -------
     0   4   0 act   6 Shell
     0   3   0 wai   6 Shell
     0   2   0 wai   6 Shell
     0   1   0 wai   1 SysGo
```

This listing is a little more brief, then the last one. Se how three shells have created. Again the child is # 4. Its parent is # 3. And finally the grandparent is #2. The previous two shells are in a WAIT state. Entering the escape twice will kill off the two youngest shells.

CUSTOMIZING INPUT LINES

Parameters may also be passed to the process that the shell started for you. Take a look at the following line.

cc test.c -m=4k

This line has for its verb, "cc". This is the module that starts the C compiler. Two parameters have been passed to it. They are the file to be compiled, which is "test.c" and "-m=4k", which tells the compiler to add 4K to the stack space. This method makes for simple program execution. For example if I am using STYLO word processor, I would use a line like,

stylo column

Stylo would then execute and load the file "column". What could be easier? I have worked with other word processors that require executing them, selecting from a menu the LOAD command, and finally specifying a filename. I prefer the first method. (By the way, Stylo will still let you select from the menu if you like.)

In the C compiler example, I redirected the error output to a file called "errors". I/O redirection is one of the many modifiers available from the shell. Here is a more complete list.

! Pipes and filters
# Specify memory size
; Command separator
& Concurrent execution
> Redirect standard output
< Redirect standard input
>> Redirect error output

I covered pipes and filters in a previous column. But it never hurts to review. The ! is used to pipe the output of one process to another. This can be very useful, especially with filters. Here a simple example.

list myfile ! strip

The LIST command usually display the file to the terminal screen. However, in this example, "myfile" is channeled to a process called STRIP. It removes unwanted, non-printable characters. STRIP prints to the standard output path. So "myfile" is printed to the screen, minus nuisance characters.

The memory size specifier is handy for changing a process' memory allocation. All OS-9 executable modules specify the amount of memory they need. But in many cases it is a minimal amount. This modifier is useful, since it lets the shell change the amount of memory used. Take the COPY command. It uses 4K of memory by default. Now imagine doing a single drive copy of a 20K long file. It would require at least 6 passes, which is more work than I care to do. Append a #20k on the line. Now it takes only two passes. Even if you have two drives, giving it more memory will definitely speed things up.

Another modifier is redirection of I/O. The standard output can be redirected to another path with the>. If you entered,

list myfile

the file called "myfile" in the current working directory would list to your terminal screen. However, entering it this way:

list myfile >/p

will send the same file to the printer. The standard output redirection is perhaps the most used.

The other two can prove useful also. Redirecting the standard input path can be done. It uses the symbol <. What do you think this line does?

solve </d0/instructions

It will start a process, running the debugger, "solve". Instead of taking instructions from the keyboard, they will come from the file "instructions" located on /d0. If you use a trick like this remember to place something at the end of the file to terminate the process. The last line of the file used by SOLVE is "Q". If you don't do this, the process may be run forever or until you turn things off.

The >> is used to redirect the error output path. Normally you want to see the errors generated on the terminal screen. But many times it is more convenient to have error go somewhere else. When running the C compiler, I will start it with a line like:

```
cc program.c >>errors
```

When the compilation is finished, I can check for errors in the file "errors". This much easier then trying to take notes as the errors scroll by.

The command separator is the semicolon. With it multiple commands can be entered on a single line. This line:

```
copy /d0/file1 /d1/file2 ; del /d0/file1 ; list /d1/file2
```

will copy a file from one drive to another. It will delete the original. And it will list the new one. Everything is executed concurrently. The shell starts the first process and goes into a WAIT state. When it is finished, it starts the next one. This continues until the entire line is processed.

BUILT-IN COMMANDS AND OTHER USEFUL ITEMS

The shell has few built in commands. Unlike the other OS-9 commands, these won't be found in the commands directory. If you execute MDIR, you won't see them in the module directory either. They can be thought of as pseudo commands since they are really part of the shell.

Two of them were touched on in last months column. They are CHD and CHX. These allow you to change working data directories and execution directories. Let us say you wanted to change disks and use a different execution directory. After inserting the new disk into the drive you would enter:

```
CHX /D0/CMDS
```

To change working directories enter:

```
CHD /D0
```

This would place you in the top directory level on the disk in drive /D0. As a rule when first booting up the system is placed into default directories. They are usually on drive /D0, although this can vary. Users with a hard disk may find themselves on /H0.

Another item the shell provides is EX. This command will EXecute a program. But it causes a process to be created without have an extra process hanging around. Here is an example. If you entered:

```
basic09
```

The shell would create the process and then go into a wait state. Now it is sitting there using up valuable memory. If you are running Level 1, your are expectally aware of the value of memory. Now execute the same using

```
ex basic09
```

This time the shell starts the process, BASIC09 and then it dies. There is small disadvantage. When exiting the process that was executed, there will be no waiting shell. A new shell will be created. The new shell will be as fresh as if you just rebooted the system.

By the way, if the shell is part of your OS-9 boot, then using EX removes the shell as a process, but still leaves it in the system memory. For level 1 this means, you'll probably get back only the allocated data area. In level 2 the process area is being returned, which means an extra 64K. But then again, you most likely have more memory at your disposal and are not overly concerned with the issue.

This bring us to the next built-in command. It is a very final sounding one, KILL! What it does is send an "abort" signal to specified process. I find this one useful when I have something running as a background task that I want to terminate. Usually I use PROCS to get the process ID number. For example, if the process ID is 5, I would enter:

```
kill 5
```

Process #5 would then die. There are two things to keep in mind. First, only the user who started the process can kill it. If you're on a system by yourself, you don't need to worry about this. Second, the process will not die immediately if there is a pending I/O operation. Say your program has gone amok and is wildly writing to the disk drive, using KILL may not be sufficient to stop it! But in most cases, things can be stopped.

SETPR is the last built command. It permits the changing of a process' priority. In the last example, we killed the process #5. Maybe we only wanted to slow it down. We used PROCS and saw that it has the priority #128. Entering:

```
setpr 5 120
```

will reduce its priority. Processes with higher status will get CPU time first. I usually find things to go along a decent rate, even when multi-tasking. So I rarely change the priorities. But if you find something is bogging down the works, try changing a few processes' priorities. See what happens!

There are a few other items of interrest. They are:

```
W  wait for any process to terminate
*  text comment line
X  abort on error
-X do not abort on error
P  turn prompt and messages on
-P turn prompt and messages off
T  all input lines are copied to output
-T input lines are not copied to output
```

The shell comes up in certain default modes. Normally, it will abort on an error. It prompts with the "OS9:" and prints messages. Input lines are not copied to the output. This last one means that when a line is inputted, you see it as it typed without it being re-echoed.

Using these items from the list can come in handy. Consider creating procedure files. Many times a procedure file will be carrying out some simple, but otherwise boring task. For example, you might have a procedure file that is copying the files of one directory to another. What if an error occurs? Everything after it would not get done. Toss in a -X at the start of the procedure file and things will continue on course. You might want a T at the beginning too! Then everything being done will get echoed to the terminal screen. By the way, don't forget to undo things when finished. I will not go into any great depths on these for now. Play around with them, next time you are creating a procedure file. See what they can do.

I considered retitling this months column, "Everything You Want To Know About The OS-9 Shell, But Was Afraid To Ask." I covered it in great detail. My intention was to show how much you can do with the shell. Many times I refer to the things that OS-9 can do. In reality the OS-9 provides the means, but the shell makes them accesible. The systems has things like PIPE, PIPER, and PIPEMAN, but the shell creates the pipes for you. The system provides multitasking. The shell will set up tasks to run simutaneously. I could on, but the point is, the shell brings OS-9 to you.

DOUBLE STRIKE

This month I put together a simple filter that you might want to add to your commands directory. This filter takes a listing sent to the printer and causes it to be printed twice. The second printing of the line is done over the

printer and causes it to be printed twice. The second printing of the line is done over the first. So the line is intensified. If your ribbon is in good shape, you'll find the line is darker than the usual printing. For a finsihed copy, it will look a lot better. If you have a printer that does not have a double strike mode built in, you might find this one useful. Even if it does have double strike, you might like this filter. My printer has a double strike mode, but I find certain other features are lacking when I use it. For example, I cannot use subscripts or super scripts. Plus, when the ribbon gets old, this will liven up the copy.

The filters in the listings are written in C. They are simple enough that they could easily be translated into assembly code and compiled into a object file. I chose C (and will do so more often), because it is universal. The code in these listings can compiled on a OS-9 level 1 or 2 system. They can easily be used on the OS-9 68K system. In fact, any system that is capable of using filters could probably use these programs. Obviously machine dependencies and differences in compilers should be taken into acccount. If you thought a assembler to be a necessary part of your system, then a C compiler should included too. More and more C will be used. Not just here, but in other areas as well.

Listing 1 shows the first attempt at double strike. It is more obvious approach. It inputs a line from the standard input path. It prints it to the standard output path, does a carrige return and prints it again. Finally it does a carriage return and a line feed. You must have a printer than can respond to carriage returns and linefeeds separately. If your printer descriptor has been set to execute a linefeed with a carriage return, then this must changed. This can be done with:

```
xmode /p -lf
```

There are three characters of interest here. They are:

\n	EOL
\r	carriage return
\l	linefeed

It turns out that the \n and\r are equivalent in OS-9. But I chose to use the more correct version. Should this be used on some other system, it should work correctly. The line:

```
%s \r %s \r \l
```

passed to "printf", prints the line, does a carriage return, prints the line again, does anther carriage return and a linefeed.

I got feeling guilty about the filter. It required a particular printer. Your printer may or may not be configurable as I have just described. So, after giving it some thought, I present the second version of the double strike program in Listing 2. This one does not use the carriage return as before. Rather it simulates it by generating back spaces according to the number of characters in the line. This one should work on all types of printers. I used the ascii representation for backspace (Octal 10). If you printer uses something else you will have to change this.

That brings us to the end of another month of BASICALLY OS-9. I hope I have helped some of you, elightened others and given inspiration to the rest. If you have any questions, comments or gripes send them my way. I answer all letters. But please include a self addressed stamped envelope. Until next time, have fun!

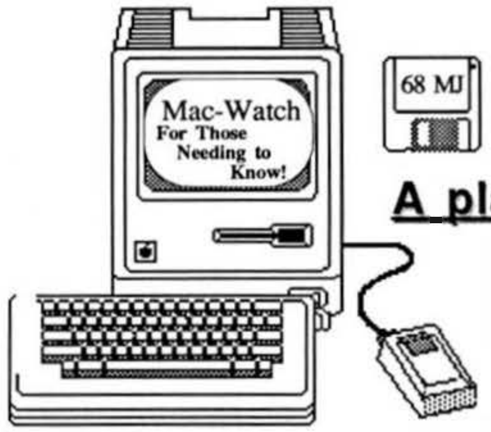
LISTING 1

```
/* *************************

    Name: Double.c
    Date: 2-MAY-87
    Author: R. D. Voigts

     *************************

    Version 1.00          2-MAY-87
     Original version.

     *************************

    Function:
    This is a filter for creating
    "double strike" lines on the
    printer.

    Usage:

    list afile ! double >/p

     *************************  */

#include <stdio.h>
#define LENGTH 81

char line[LENGTH];

main()
{
        while( gets(line) != NULL )
             printf("%s\r%s\r\l", line,
line );
}
```

LISTING 2

```
/* *************************

    Name: Double.c
    Date: 2-MAY-87
    Author: R. D. Voigts

     *************************

    Version 1.00          2-MAY-87
     Original version.

    Version 2.00          3-MAY-87
    Uses back space instead of
     carriage returns and linefeeds.

     *************************

    Function:
    This is a filter for creating
    "double strike" lines on the
    printer.

    Usage:

    list afile ! double >/p

     *************************  */

#include <stdio.h>

#define LENGTH 81
#define BS "\010" /* back space */

char line[LENGTH];

main()
{
    while( gets(line) != NULL ) {
        printf("%s", line );
        backup(strlen(line));
        printf("%s\n", line );
    }
}

/* *************************

    Function:
    Prints backspaces according
    to the passed argument.

     *************************  */

backup(i)
int i;
{
    register int j;
    for ( j=0; j<i; j++)
        printf("%s", BS );
}
```

EOF



The Macintosh™ Section

Reserved as a

A place for your thoughts

And ours.......

Mac-Watch

Though underpowered, the original 128k Mac bought gasps of amazement because of the new capabilities it brought to to "the rest of us". One of the more exciting of those capabilities was the ability to easily combine text and graphics on the same page. Software producers saw the appeal of text-graphics integration capabilities and reacted with ever improving software. Graphics programs (like SuperPaint) now have better text handling capabilities, and word processor programs (like WriteNow) contain better graphics handling capabilities.

Ready, Set, Go! 3™ and its competitors carry this trend a step further by providing a package of powerful tools for easily integrating text and imported graphics into an attractive page design for newsletters, bulletins, advertisements, and similar publications. The task performed by such software is labeled "desktop publishing". Ready, Set, Go! is one of the two best sellers in this field, along with PageMaker™; and for the moment at least, has more power than its closest competition.

RSG3 provides for the design of pages to include designated text and graphics areas, flow of text from column to column in any order throughout the document, importing and precisely placing graphics with automatic text flow-around, text editing, and control of text aesthetics.

Page Layout

In beginning the design process, the user designates the number of rectangular grids to be shown to aid in page layout. For example, a 3 x 3 grid might be used for a simple 3-column page of text, while an 8 x 8 grid might be used for a more complex page with headings and graphics.
Blocks are then placed on this grid and designated to receive text or graphics, or to remain as white space. An optional snap-to-grid feature allows layout blocks to be placed quickly and very precisely. The spacing between text columns or text and graphics is automatically set.

If the grid arrangement is not precise enough for some users, the Specification option may be used to set the exact location of text or graphics blocks in points, inches, or centimeters.
As originally set up, the various text block are totally independent, but RSG3 provides for linking then so that text flows from column to column, around pictures and from page-to-page in any desired order. Text can even be made to skip columns and/or pages or flow to earlier pages. All this is accomplished merelyby clickingon each text block with a "linker tool" in the order that text flow is to occur.

RSG3 documents may be viewed in actual size, half size, 75% size, double size or may be sized to fit the screen. Editing and rearranging of page layout may be accomplished in any view with the results if the change beinginstantlyvisible. Additionally, facing pages may be viewed together but changes are not possible while in this view.

Filling in the Blocks...

Having designed the basic layout of the document, it's time to add text agraphics. Text can be generated using RSG3's built-in word processing capabilities, or prepared externally. RSG3 reads documents created by Microsoft Word™, McWrite™, or any document saved in text (ASCII) format. This is accomplished by selecting Get Text from the menu, then double clicking the chosen text file. The text then flows throughout the RSG3 document around graphics windows, between columns, and between pages in accordance with the predetermined linkages.

Importing graphics is just as easy. The graphics window is clicked to make it active, then Get Picture is chosen from the menu. The desired graphics file, when clicked, instantly appears in position on the RSG page. The cropping tool can be used to reposition the graphics in the graphics window. Also, the window can be resized to display more or less of the chosen graphics. To add a finishing touch, several styles and weights of lines may be used to separate headers, footers, and columns. Also, graphics or text areas can be enclosed in circles, boxes, or boxes with rounded corners in various line widths. Various patterns are provided for the lines, circles, and boxes and for filling in the interior of graphic objects.

The real surprise is how little time this process takes. If you know in advance how a four-page newsletter is to be formatted and have text and graphics files ready, you can easily set up the formulated RSG pages, import text and graphics, and add finishing touches in much less than 10 minutes.

Word Processing

RSG3 includes fully competent word processing capabilities. While not as full-featured as some dedicated word processing programs like Word, RSG3 can easily handle generation and editing of small to moderate documents. Features like use of clipboard and scrapbook, selecting text, deleting text, moving text, copying text, find and replace, and font and style selection work essentially as they do in McWrite.

In addition to above basic word processing features, RSG3 includes a 65,000 word spelling checker and a glossary function. Where this program really shines, however, is the near-complete control over text aesthetics. RSG has a real-time hyphenation function which when combined with the "justify" option can result in very smooth text margins. While hyphenation in accordance with a table of rules occasionally results in errors, RSG compensates by providing a hyphenation exception dictionary. The user may put as many words in this dictionary as he likes.

RSG provides for right, left, center, or justified format. The spacing between lines and between paragraphs can be precisely set. The tightening of spacing between letterskerning) can be accomplished automatically or under user control. The cursor keys on the MacPlus keyboard can be used to adjust kerning a point at a time. The kerning function may be used on an entire heading or on as little as two selected letters. Letters may be spaced apart (the opposite of kerning) in essentially the same manner and with the same level of control as for kerning. Selected text can be shifted up or down from the baseline position by a chosen number of points. This may be accomplished through use of a dialogue box or through use of a MacPlus keyboard. RSG provides for three styles of paragraph indentation. Each style can be set as to the number of points the affected line is offset.

Some roughness was encountered in the use of the word processing features. On a number of occasions difficulty was experienced in inserting the text tool into existing text for editing, particularly at the beginning of lines. Also, in selecting text which is partially off-screen, the scroll function would sometimes work only in scrolling from right to left. If an insertion point were chosen to the left of the target text, that portion of the text that is off-screen would not always be selected. On occasion the scroll function seemed slow and jerky compared to McWrite. These kinks will not present any major hinderence to most users since text will usually be prepared in other dedicated word processing programs with only light editing being done in RSG3.

Revisions

Rare is the composition that will not get revised a number of times before printing. RSG allows the user to make layout changes and instantly see the results. If you do not like the position of a graphics box, simply select it and move it. Not enough white space around the picture? Easy to fix! Just select the graphics block handle and resize it. Note that as the change occurs in the above cases, the text automatically flows around the graphics block in accordance with its new position or size. Pages can be inserted or deleted. The linkage of text flow between columns and pages can be changed. If that 2,000-word article lacks an inch filling the designated text block, the text block can be appropriately resized and other filler material put in the blank space.

The Manual

The RSG3's manual is a slick publication which is more like a magazine than a typical software users' guide. The bare basics are adequately covered through a tutorial and various reference sections. The RSG3 disk contains text and graphics files to support the user in setting up a the sample document for practice. After presenting the introductory material in stylish manner, the utility of the manual drops off significantly, and the discussions of more advanced program capabilities are quite sketchy. The 4-page chapter on Layout Basics is especially lacking. Most users will be diaapointed with this manual.

Printing

RSG3 supports printing on the Imagewriter, the Laserwriter, or any other typesetter or printer that understands PostScript. Programming directly in PostScript is also supported.

Will It Fit Your Mac?

RSG3 will run on any Mac with at least 512k. With difficulty, it can be made to work on systems with only one disk drive. The spelling checker cannot be used with single disk systems. RSG3 is not copy protected.

Should You Buy It?

RSG3 is a highly competent desktop publishing program with enough power to fully meet most users' needs. The power is complemented nicely by its ease of use. For those needing a desktop publishing program which they can learn in a weekend and use to turn out truly professional results right away, RSG3 will be a good choice.

James Law
1806 Rock Bluff Rd.
Hixson, TN 37343
(615) 842-5471

FORTH

A Tutorial Series

By: R. D. Lurie
9 Linda Street
Leominister, MA 01543

A PROGRAMMING PHILOSOPHY

I used to wonder why I never heard much about long and complex programs being written in FORTH. Well, I think that I now know the reason. Of course, large application programs have been, and are being written in FORTH, but it is usually not necessary for the average task to be such a big deal.

In contrast to most other languages, FORTH, by its very nature, encourages you to write short, concise programs which you can later chain together to accomplish some major undertaking. I guess that C is the nearest conventional language to FORTH in making it easy to chain short programs by the use of pipes, etc. FORTH doesn't have the pipes of C, but it doesn't really need them.

As an example of what I mean, consider the following case history. I have been rewriting my spelling checker in FORTH, converting it from assembly language. The dictionary consists of 14 lists of words in strict alphabetical order. Each list holds words of only one length, such that "RAM" and "ROM" are in the same list, but "key" and "keys" are not in the same list. I found that I could search the fastest in the least amount of RAM by using this dictionary structure.

In making this conversion, I have modified the dictionary structure by removing all of the "white space" from within each list of words. Sometimes, the white space was a blank and sometimes it was a carriage return. I could have removed any of these with an editor, but the obvious way to do it was write a program to do so.

I also needed to count how many words there were in each list beginning with each letter of the alphabet. After trying to do it by reading the screen and getting a different count each time, I wrote another program strictly for this purpose.

I needed to make several other changes and notations about each list, so I wrote a little program to do each one separately, as I needed the information.

All of this could have been combined into one program which was much shorter than the sum of the lengths of each of these little "utility" programs; but it would have taken me much longer to write and debug it than all of the short ones did. Besides, I now have several additions to my toolbox ready the next time I need to do some odd-ball job.

In summation, my FORTH programming philosophy is to write several short programs, rather than one long program, whenever possible. Furthermore, I never throw away any debugged definition, since I will surely find more use for it in the future.

TWO USEFUL UTILITIES

I would like to share with you a couple of one-screen utilities that I find myself using quite a lot.

QX

The first one is called QX and is shown in screen #62. This was inspired by the QX utility supplied by LMI in their Z-80 FORTH; however I have made a couple of changes in it which make it fit more comfortably into my programming habits. QX stands for Quick indeX, and produces a screen index in a compressed form. Figure 1 is an example of its use with the program NO-<SP> , which I will describe later.

One of my main uses for QX is in finding the first empty screen in a block of screens that I am working on. If I am writing a long program, I make a habit of reserving blocks of screens for specific job catagories, such as file I/O and help screens. I need to know the next available screen in a block when I am ready to add another help screen, etc. QX is very quick at doing that and usually does not cause something useful to scroll off the screen while it is working. Of course, QX does the traditional jobs that INDEX does, just in less space.

QX is called in the traditional FORTH manner.

```
<first-screen#> <last-screen#> QX
```

Since the algorithm may not be immediately obvious from the listing (screen #62), I will spend some time explaining it. The initial CR simply forces printing to begin on a fresh line. This forces everything to line up in neat columns, which makes the display easier to read.

Line #2 may be the most confusing part of the definition. The initial 0 serves to initialize the column counter, which never leaves the Data Stack. The definition probably would have been easier to follow if a VARIABLE had been assigned for the counter, but I wrote the definition long before I ever thought that it might become the subject of a description in 68' MJ. Besides, this is a good place to illustrate a Data Stack position being used to hold a counter.

The ROT ROT in the rest of line #2 simply shifts the various numbers so that the 0 (counter) winds up at the bottom of the Data Stack.

The phrase 1+ SWAP adds on to the value of the last screen number and then exchanges the two numbers so that they are in the proper order for the DO ... LOOP .

By setting the DO ... LOOP limits in this way, the loop index, I , assumes the number value of the desired screen. Therefore, I can be printed with a formatting statement 4 .R so that it always occupies 4 spaces, right justified. SPACE skips one space as a further aid in formatting.

The real work is done in line #5. I BLOCK loads the proper screen into RAM and leaves its starting address on the Data Stack. This address is used with the 14 (as the count) as the parameters for the TYPE command. The count of 14 is not truly arbitrary, since it produces the most infonnation in four columns on and 80- character screen. If you change it, be sure to adjust the number of columns printed; otherwise, you can get quite a jumbled mess for you display which you will not be able to read.

Lines #6-#9 act as a column counter. I needed this because one of my printers cannot be made to do a carriage-return without an explicit instruction. If you use an 80-column display and a printer which automatically returns after printing 80 columns, then you do not need these lines.

DUP makes a copy of the column counter, which is now at the top of the Data Stack. 4 MOD divides the count by 4 and saves the remainder on the top of the stack. The remainder is incremented by 1 and compared to 4, the desired number of columns. Remember, the results of 4 MOD must range 0-3. I put in the 1+ so that I could make the comparison directly to the desired number of columns, so that I would understand the algorithm several months later (like now, for instance).

I any case, if the incremented counter does equal 4, the IF portion of the conditional phrase is executed. This consists of sending a CR to the display, dropping the old value for the counter, and starting a new count at 0.

Otherwise, the ELSE part of the conditional phrase is executed. In this case, the column count is incremented by 1, and a SPACE is sent to the display for formatting purposes.

The ?BRK and ?LEAVE are somewhat implementation dependent, in that they are not common to all versions of FORTH. ?BRK tests the keyboard for a key-press, and ?LEAVE exits the DO ... LOOP if one is detected. These two are not necessary, but do provide a convenient panic-button.

DROP clears the no longer needed counter from the Data Stack and CR forces a new line, again in the interest of neatness.

QL

This utility was originally written as a means for getting the maximum number of programming lines displayed on the screen or printed on a page without having to make a specially edited copy of consecutive screens. QL (shown in screen #61) produces the type of output shown in Figure 2. Essentially, QL strips all blank lines from a screen. This makes it harder to read and understand, but it cuts way down on the number of lines required to show a group of screens. Normally, I prefer to have only one definition on a screen, and this results in a lot of empty lines. QL gets rid of these and gives me a nicely condensed Quick List.

As with QL, QX uses the traditional FORTH command line:

```
<first-screen#> <last-screen#> QL
```

Line #1 serves to print a heading which contains the current date. .SYS-DATE is similar enough to .DATE (68' MJ, 11/86) that I will not take the space to describe it here.

Line #2 sets up the "outside" DO ... LOOP which has the screen numbers as its parameters. Some of you may find it confusing that the same loop index is called I in line #3 and J in line #5. This is the FORTH convention that the index for the currently executing DO ... LOOP is called I , and the index for the next outer DO ... LOOP is called J . Some versions of FORTH do not have access to this outer loop counter, so you will have to keep it in a separate variable. On the other hand, some FORTHs, such as FF9 allow one more level of nesting availability by having a K . By the way, this has nothing to do with the actual number of DO ... LOOP 's which can be nested; it just refers to how many indices that can be called from within a given DO ... LOOP .

The screen number is printed in line #3 in the purely conventional manner.

The DO ... LOOP in line #4 is set up to print a maximum of 16 lines. At least, it will certainly look at 16 lines, even if none are printed.

J BLOCK loads the current screen into RAM buffer from disk and leaves the starting address of line #0 on top of the Data Stack. I is the current line number, so I 64 * multiplies the line number by the length of a line, and + adds this number to the buffer address already on the Data Stack. In this way, each iteration of the inner DO ... LOOP places the starting address of the current line onto the top of the Data Stack.

The 64 -TRAILING tells the computer to start counting back from the address on the stack plus 64 until the first non-blank is encountered. When this happens, leave the number of remaining characters in the line as a count on the top of the Data Stack. It is a peculiarity of -TRAILING that the address which was already on the Data Stack is not lost, which is an exception to the normal activity of a FORTH word.

DUP makes a copy of the count left by -TRAILING to be used as a flag for the following IF ... ELSE ... THEN conditional. Since 0 is identical to FALSE, and non-0 is identical to TRUE, the count, itself, can be used as the flag. This is considered good FORTH practice, but can be confusing to beginners. We needed to duplicate this number, since it is to be used as the count for a TYPE command later on.

If the count was greater than 0, line #7 prints a formatted line number as three spaces, right justified. A SPACE is then printed as part of the output formatting. TYPE then prints the characters which were previously found. A CR finishes the output for this line of text.

Had the count been 0, the ELSE branch of the conditional would have used 2DROP to clear the address and count for this line from the Data Stack.

Remember that two DO ... LOOP 's were nested, so we need to end the definition with LOOP LOOP ;.

NO-<SP>

I won't spend a lot of time with a detailed description of NO- <SP>, since virtually all of the definitions have been described in a previous column. However, I do want to point out the major features of the program and give a short description of the algorithm.

As I mentioned previously, the purpose of NO-<SP> was to remove all "white-space" from a list of words, but, of course, it would work with any text file.

The program reads the file into RAM for processing. This is easy, because I have plenty of unused RAM with this short program.

I need two pointers and two counters. R-POINTER is the read pointer and W-POINTER is the write-pointer. TALLY counts the total number of characters and DROP-TALLY counts the white-space characters as they are skipped. A character is read at R- POINTER and R-POINTER is incremented. The character is checked for "white-space" and ignored if it is a <SP> or a <CR>, and DROP-TALLY is incremented.

If the character is not a <SP> or a <CR>, it is stored at W- POINTER and W-POINTER is incremented.

Notice that R-POINTER gradually drifts away from W-POINTER by the count stored in DROP-TALLY. This difference is eventually used by CLEAR-SURPLUS to erase the surplus characters left after NO-<SP> has done the first part of its job.

Finally, the finished file is written back to disk.

```
SCR # 61
  0 : QL      ( n1 n2 -- )                               \ RDL 03/15/87
  1      CR      55 SPACES     .STD-DATE      CR
  2      ( n1 )   ( n2 )   1+      SWAP      DO
  3        CR      ." SCR #"      I      .      CR
  4        16   0   DO
  5           J      BLOCK      I      64      *      +
  6           64      -TRAILING      DUP
  7             IF      I      3   .R
  8                 SPACE
  9                 TYPE      CR
 10             ELSE 2DROP
 11             THEN
 12         LOOP
 13      LOOP ;
 14
 15


SCR # 62
  0 : QX      ( low-SCR# high-SCR# -- )                  \ RDL  02/16/87
  1           CR
  2           0 ROT ROT
  3           1+ SWAP DO                                  \ loop limits
  4             I 4 .R SPACE                              \ screen#
  5             I BLOCK 14 TYPE
  6             DUP 4 MOD 1+ 4 =
  7               IF CR DROP 0
  8               ELSE 1+ SPACE
  9               THEN
 10             [illegible]     ?LEAVE
 11           LOOP
 12           DROP
 13           CR ;
 14
 15 \ A condensed screen index.

Figs

10 17 QX
  10 \ NO-<SP>         11 VARIABLE R-POI   12 : INITIALIZE      13 : FILE-NAME
  14 : OPEN-FLEX-FO    15 : WRITE-FLX-F    16 : DO-BYTE         17 : FLEX-TEXT

 ok
```

Figure 1. A demonstration of the QX utility.

```
10 17 QL
                                                      06/08/87

SCR #10
  0 \ NO-<SP>                                           \ RDL 05/26/87
  1 \      Remove all of the "white-space" from a text file.
  4 25000 CONSTANT FILE-SIZE
  6 CREATE FILE FILE-SIZE ALLOT
```

```
SCR #11
  VARIABLE R-POINTER
  VARIABLE W-POINTER
  VARIABLE TALLY
  VARIABLE DROP-TALLY
  : ?YES/NO      ( -- ? )                    \ RDL  01/11/86
       ." (Y-yes/N-no) "
     KEY DUP DUP EMIT                      ( fetch input           )
        ASCII Y =
           IF DROP TRUE
           ELSE DUP ASCII N =
              IF DROP FALSE
           ELSE DROP CR RECURSE THEN THEN ;  ( a recursive loop )

SCR #12
  : INITIALIZE  ( -- )                          \ RDL 05/26/87
      FILE FILE-SIZE BLANK      \ clear FILE
      FILE R-POINTER !
      FILE W-POINTER !
      0 TALLY !
      0 DROP-TALLY ! ;
  : CLEAR-SURPLUS    ( -- )                     \ RDL 05/26/87
      TALLY  @    DROP-TALLY  @    -    1+     DUP
      FILE    +
      FILE-SIZE    ROT    -
      ERASE ;

SCR #13
  : @FLEX-NAME  ( -- )                          \ RDL  01/02/87
    ." FLEX file name: "                     \ prompt
      PAD 15 EXPECT                            \ load FORTH input buffer
      PAD SPAN @                               \ "adr" and "count"
      FCB FILENAME                             \ move name to FCB
         ?ERR                                  \ report a error
      1 FCB SETEXT ;                           \ default to TXT
  : OPEN-FLEX-FOR-READ     ( -- )                 \ RDL  01/02/87
      @FLEX-NAME                               \ input FLEX file name
      1 FCB C!                                 \ open for read
      FCB FMS                                  \ call FLEX FMS
         ?ERR ;                                \ report an error

SCR #14
  : OPEN-FLEX-FOR-WRITE    ( -- )                \ RDL  01/03/87
      @FLEX-NAME                             \ input FLEX file name
      2 FCB C!                               \ open for write
      FCB FMS                                \ call FLEX FMS
        ?ERR ;                               \ report an error
  : READ-FLEX-FILE    ( -- )                     \ RDL  01/02/87
    FILE FILE-SIZE +               \ use addresses for limits of
        FILE                       \    DO ... LOOP
      DO FCB FMS>                  \ read a byte from the FCB
        ?DUP                       \ an error signal?
           IF 8 - ?ERR THEN        \ process error (halts program)
           ?EOF ?LEAVE             \ exit loop gracefully at EOF
        I C!                       \ store byte in RAM
        1 TALLY +!                 \ count the byte
      LOOP ;

SCR #15
  : WRITE-FLEX-FILE  ( -- )                      \ RDL  01/03/86
      FILE TALLY @ + FILE DO   \ set writing loop
        I C@                   \ fetch byte
        WRITE-BYTE             \ write it to the FCB
      LOOP ;
  : @FILE-BYTE  ( -- c )                          \ RDL 05/26/87
      R-POINTER  @  DUP        \ fetch pointer to byte
      1+  R-POINTER  !    C@ ;
  : DO-CHAR      ( c -- )                         \ RDL 05/26/87
      W-POINTER  @             \ fetch pointer to byte
      DUP
      1+                       \ increment pointer to next byte
        W-POINTER  !           \ store pointer
      C! ;                     \ store char at unincremented adr
```

```
SCR #16
  : DO-BYTE       ( -- )                             \ RDL 05/27/87
      @FILE-BYTE                  \ fetch byte from RAM
  DUP
           CASE
    13  OF     DROP    1  DROP-TALLY  +!                       ENDOF
    BL  OF     DROP    1  DROP-TALLY  +!                       ENDOF
         DO-CHAR

           ENDCASE ;
  : @FLEX-TEXT  ( -- )                             \ RDL  01/03/87
      CR
       OPEN-FLEX-FOR-READ
       READ-FLEX-FILE
       CLOSE-FILE ;

SCR #17
  : !FLEX-TEXT  ( -- )                             \ RDL  01/03/87
      CR
       OPEN-FLEX-FOR-WRITE
       WRITE-FLEX-FILE
       CLOSE-FILE ;
  : NO-CRS       ( -- )                              \ RDL 05/27/87
       INITIALIZE
       CLS ." SOURCE FILE"  CR
       @FLEX-TEXT                 \ fetch text from FLEX disk file
       FILE DUP TALLY @ + 1+ SWAP  \ set DO ... LOOP limits
         DO DO-BYTE LOOP          \ process text
       CLEAR-SURPLUS
       CR  CR  ." OBJECT FILE"  CR
       !FLEX-TEXT                 \ store text into FLEX disk file
       CR  ." FINISHED" ;
 ok
```

Figure 2. A demonstration of the CRL utility.

Pascal A Tutorial

By: Robert D. Reimiller
Certified Software Corp
616 Camino Caballo
Nipomo, CA 93444
805 929-1359

This month we will enter the example program given last month and debug it, followed by making a binary file that can be run as a command from the shell.

Change to a directory that you wish to use for this example and enter the editor by typing :

```
$ se
```

This will start the editor and provide you with a blank screen, enter :

```
new ex1
```

You will now given a line number of one (in half brightness if your terminal supports this) and the cursor will be at the first position of the line. Start typing the first line of the program, terminated by a carriage return. If you have ever used Stylograph then you should pick up the use of this editor quite quickly, as they share many of the same features. You can refer to the reference sheet for the editor to learn the editing commands, you can also use the built in help facility if you forget what control key to use.

You can indent the program as shown, or use any other method you feel comfortable with. Many people put reserved words in upper case, but it does not matter to the compiler.

When you have finished typing in the program, go back to command mode. A smart first step is to save the program, simply enter :

```
s
```

Which will save the program under the default name of "ex1". To check for syntax errors (such as typos), enter :

```
$pc -@
```

This will compile the program in the edit buffer (thats what the "@" means) but will not generate object code or a listing. If the compiler finds any errors, it will show where the error is (pointing to it's best guess of the item in error) and tell you what the error is. You can continue with the compilation by entering any character other than "Q" (or "q"). After the end of the compilation, or if you enter Q after any error, you will be returned to the editor with the cursor pointing to the location of the last error. You can now make the necessary changes and then enter control N to move the cursor to the previous error in the file until there are no more errors in the stack. Both the compiler and the assembler can stack up to 20 error locations in this manner.

When you can compile the program with no errors, you are ready to move to the testing stage. On your successful compilation, note the value it shows for "stack" at the end of the compilation. Save the corrected version of the program and exit the editor by entering save and quit :

```
sq
```

In order to take advantage of the integrated package it is necessary to setup a control file that contains all the information needed to compile, assemble, debug, and link your program. The linkage creator is the program that produces this control file. Enter the linkage creator by entering :

```
$ lc ex1
```

The linkage creator will try to open the compiled relocatable object code for the program "ex1" to try to determine the stack size required. Since we haven't actually produced this file, it will inform you that it can't determine the stack size. No problem, since you remembered the value from 2 paragraphs above, right?

There are two main modes, automatic setup, which is designed for programs that will run under OS-9, and the second mode, which is designed for stand-alone programs. This program is to run under OS-9, so we select automatic mode. In the example program there are no buffers that might need to be expanded, so we can set a fixed stack size. Other programs, especially those that use the heap, need to have a way to specify the stack/heap size when they are run. This is done using the "Z" command line option, as is done for many of the programs in the pascal package. Since we do not need this flexibility, we will answer NO when asked if we want to handle the Z option. We will next be prompted for the stack, heap, and varib size to allocate, since the compiler said that we needed $2FC bytes, lets specify $400 bytes to leave room for operating system calls and such (do not enter the '$', it is assumed).

We are next prompted for the library directory to use, if installed as suggested, we would specify "/dd/". We are next prompted if we want to load other pascal modules, for modular programs. This example is very small and only needs the main program, so we answer with a carriage return. The linkage creator allows up to 40 - 80 character lines to specify modules, which should handle even the largest project. In a similar manner we are prompted for additional assembly language files, we have none, so we answer with a carriage return. Next, additional libraries can be specified, this would include such items as graphics or floating point processor libraries. We don't have any, so we answer with a carriage return.

We next are asked for linker command line options, lets say we want the load map to be written to disk, all we need to do is specify the file name as the command line option, answer with "ex1.mp". "Map options" allows us to specify the format of the load map, lets use "F" for a full map. Next we are prompted for include files, we aren't using any, so answer with a carriage return.

Next we have a prompt for debugger command line options, the small example should not require any special memory handling, so we answer with a carriage return. The next prompt is for the target debugger, which we will not be using, so just answer with a carriage return.

The next two prompts allow multiple options, one option set per line. The first is for the compiler, this is used to provide us with a menu of standard options when compiling from the pascal shell. We know we will want to compile the example for use with the debugger, so lets make the first line "-d".

Later, we want relocatable object code so we can generate a loadable OS-9 module, so make the second line "-r". We would also like a listing, so we might have something like "-lp62ftw96 /p", with variances due to your printer specifications, of course. We can combine generating code and listing of course, into something like "-dlp62ftw96 /p". When you are done putting in options (10 maximum), just enter a carriage return. The same procedure is used for the assembler, with likely options of "-o", "-lp62ftw96 /p", and "-olp62ftw96 /p". Last, we are prompted for screen editor options, normally the only time this is done is for large modules that need additional screen editor buffer space, enter a carriage return for this example.

The linkage creator now writes out the control file, called a "shell file", which has a suffix of ".sf" and the same file name as the pascal source "ex1.sf". It also writes an assembly language source file that is used to run the pascal program from OS-9. This file has a suffix of ".ps". If we were writing a stand alone program the code from the compiler is the same, only this assembly language file is changed to operate under the different environment, we will cover that another month. The file "ex1.ps" is then assembled into "ex1.pa", ready for linking when we get to that step.

As a summary of what was entered for the linkage creator :

```
OmegaSoft Linkage Creator Version 1.21
Copyright 1987 by Certified Software Corporation
Global stack size not determined
auto sety ? y
Process Z command line option ? n
Stack, heap, and Varib size : 400
Library directoryname : /dd/
Other pascal files :
Other assembly files :
Other library files :
Linker command line info : ex1.mp
Map options : f
Include files :
Debugger options :
Target debugger options :
Compiler options : -d
Compiler options : -r
Compiler options : -lp62ftw96 /p
Compiler options : -dlp62ftw96 /p
Compiler options :
Assembly options : -o
Assembly options : -lp62ftw96 /p
Assembly options : -olp62ftw96 /p
Assembly options :
Editor options :
```

```
OmegaSoft 68000 Assembler version 1.21
Copyright 1987 by Certified Software Corporation
Errors : 0 Code : 00A8 Data : 0000 Varib : 0000 Table
14 of 944
Code assembled
$
```

The next step is the Pascal Shell program which can be started by entering :

```
$ ps ex1
```

It first reads the "shell file" to build a list of pascal, assembly, and include files. This is a menu driven program with the following options :

```
Edit module
Compile module(s)
Assemble module(s)
Debug
Link
Run Chain
Background chain   idle
Exit Pascal Shell
Target debug
Execute linked program
Beep on termination
Restart
Review
Abort background
Pass command line to shell
```

You use the same keys for cursor movement as you did for the editor (setup during installation). Hit a carriage return to make a selection from the menu. Rather than getting straight down to business, lets spend some time learning what each of the selections will do for us.

Edit module - you will be shown a list of files that can be edited. These will include pascal source, assembly source, and include files, which were specified in the linkage creator. The normal cursor control keys are used to move from one file to another. Enter a carriage return to exit back to the main menu. Enter a "S" to edit the selected file.

Pascal files will have a "P-" in front of them, likewise, "A-" for assembly, and "I-" for include and misc. files. When you are done editing the selected file, you will be back in this menu, you can enter "F" to flag the file as modified, which will change the "x-" to "x*". There is also an automatic mode that will flag files that have been modified since the last "chain", or based on the update time of a file, similar to a "make" utility function.

Compile module - you will be shown a list of pascal source files, for each file you can skip it, or enter "S" to select it for compilation, or "R" to reset it so it will not be compiled. You can also enter "A" which will set all flagged files for compilation. These compilations will be queued and done in the "chain".

Assemble module - same as compile, except you are shown a list of assembly source files.

Debug - Puts the debug command into the queue so you will enter the debugger after the selected files have been compiled/assembled. You can use the default command line options, specify new ones, or cancel queuing of this command.

Link - similar to debug, will run the linking loader after compilation and assembly.

Run Chain - This is a multiple step operation. It first compiles all selected pascal source files, then assembles all selected assembly language files, and then does the link, debug, or target debug if they were selected. If at any step an error occurs, it will stop and return to the main menu. You can then fix the error, and run the chain again. it will restart at the step that caused the error.

Background Chain - This will do compilations and assemblies in background, allowing you to do further editing on other files in foreground. Like the foreground chain, if an error occurs, it will stop and display an "error status". When it is done, you will find on RAM disk (or wherever you want it) a file that contains all of the sign-ons and error messages (if any) generated by the compilations and assemblies. This file name will show up on the edit file menu if you need to inspect it.

Exit Pascal Shell - returns to OS-9.

Target Debug - similar to debug, will run the target debugger after compilation and assembly.

Execute linked program - will execute the linked program, you will be prompted for a command line to use.

Beep on termination - allows enabling of a "reminder beep" at one second intervals once the chain has terminated.

Review - shows what will occur when the chain is run (compilations, assemblies, etc.).

Restart - resets all queues and re-reads the shell file. Useful after you edit the shell file for any reason.

Abort Background - stops the background chain once it has finished the step it is doing.

Pass command line to shell - allows you to run OS9 commands from within the pascal shell (directory listings, etc.)

Our first step will be to compile the example program for use with the debugger. Position the cursor to "compile module" and enter carriage return (CR). You will be shown the default options, move the cursor to "-d" and hit carriage return, this will move you to the file selection menu. The cursor will be positioned to "ex1", enter a "S". Enter a CR to exit back to the main menu.

Position the cursor to "debug" and enter CR. We will use the default command line option, so enter CR. Position the cursor to "Run chain" and enter CR. The compilation will be done, and then the debugger will be started and the file loaded. At this point we will be in the debugger "filer" mode. Enter "P" followed by CR to enter "pascal" mode.

Since this program uses the standard input and output paths only, running it is very simple. Just enter "G" and a carriage return. As a simple test, enter :

```
test: do.b $400-. fill out the buffer
```

It will then print :

```
test ds.b $400-* fill out the buffer
```

Enter :

```
end
```

followed by a control Z and then carriage return. This will generate an EOF which will stop the program. To have it do something useful, we need to redirect the standard input and output paths, this is done with the "set command line" command, enter :

```
P> sc
$> </dd/defs/funcs.a >/dd/pdef/funcs
```

This will convert the funcs.a defs file into a RA compatible include file called funcs in the pascal definition directory "PDEF". Enter "G" again followed by a carriage return. It will then run the program, but you will not see what is going on since the output is redirected to a file.

There are many powerful debugging capabilities available, both at the pascal and assembly language levels, we will go over some of these next month. Next, enter Q with a CR to exit the debugger, which returns us to the Pascal Shell main menu. Go back and select compilation again for file ex1, except select the option "-r", this will generate relocatable object code. Then, instead of selecting debug, select link. Run the chain, it will compile the example program generating relocatable object code, and then link it with the stack setup code (ex1.ps) and the runtime libraries to generate file file "ex1.lo" in the current data directory.

You will then be back in the main pascal shell menu. To try the linked version of our example, select "execute linked program", and when prompted for the command line, enter : "</dd/defs/io.a >/dd/pdef/io" so we will convert the io.a file. After this is done, you might want to add this little utility to your commands directory. Select "pass command ..." and enter :

```
$copy ex1.lo /dd/cmds/ex1
```

You can then run this program any time you want by using the name "ex1". You can now exit the pascal shell by positioning the cursor to "Exit Pascal Shell" followed by a carriage return.

Note that the file io.a contains two symbols with the same spelling, PD_NAME and PD_Name, this does not cause a problem in R68 because it follows the "C" convention of treating upper and lower case as different. This will not work for RA because it uses the Pascal convention of treating upper and lower case as the same (except in string literals). I suggest changing PD_Name to PD_NameP.

OmegaSoft is a registered trademark of Certified Software Corporation, OS-9 and OS-9/68000 are trademarks of Microware Systems Corporation.

EOF

FOR THOSE WHO **NEED TO KNOW** 68 MICRO JOURNAL™

Logically Speaking

The following is the beginning of a continuing series. Most of you will remember Bob from his series of letters on XBASIC. If you like it or want more, let Bob or us know. We want to give you - *what you want!*

The Mathematical Design of Digital Control Circuits

By: R. Jones
Micronics Research Corp.
33383 Lynn Ave., Abbotsford, B.C.
Canada V2S 1E2

PREFACE

The major part of this tutorial is based on a series of lectures I began in the early 60s. These 2-hour lectures (covering a period of 25 or 26 weeks) were delivered to groups composed of a mix of engineers, technicians and non-technical persons, some with very little knowledge of mathematics and others with an extensive grasp of the deepest aspects of the subject. And yet, because the "math" of digital logic is so unlike ordinary mathematics, it could truly be said that they all started off on an equal footing.

Be of good Cheer

Therefore to those of you who may be wondering whether you'll be able to master the lessons to come, I would say "Be of good cheer! If you can count to 100 or so, and have a fairly logical mind, you can do it." Just be sure NEVER, EVER, to move ahead until you have mastered each stage, and above all - and I cannot stress this too strongly - DO the exercises and self-tests I shall be providing along the way.

The trouble with "logic" is that very often the subject-matter is so logically apparent that the student tends to say to him/herself "Yes. I see that. It's so obvious!" and to skip giving a point the deep attention it really deserves. Words are treacherous - so examine each statement carefully before accepting it as "logically so".

For example, let's take a look at the following set of logical statements. They also happen to be mathematical, but let's concentrate on the "logic" :

Example

1. Tom has 95 marbles.
2. Dick has 4 more marbles than Tom.
3. Therefore Dick has 99 marbles.

Now, if we know for sure that Statement 1 is TRUE, and that Statement 2 is TRUE, then no-one will deny that Statement 3 MUST also be true. Obvious? Take a second look before saying "Yes". If we're all agreed, let's move on. Most folks would also concede that the whole structure of this set of statements is consistently and logically TRUE, in that we don't necessarily have to relate to Tom and Dick. We could replace 'Tom' with 'Mary', 'Dick' with 'Santa Claus' and 'marbles' with 'toys', and our logical conclusion would still be TRUE. Or how about :

1. Uncle Fred has 95 dollars.
2. Aunt Minnie has 4 more dollars than Uncle Fred.
3. Therefore Aunt Minnie has 99 dollars.

Again, provided that Statements 1 and 2 are TRUE, then it follows that Statement 3 MUST also be true. As long as the substitution of 'Uncle Fred' for 'Tom', 'Aunt Minnie' for 'Dick' and so on, is consistently carried out, the logical conclusion remains TRUE.

Continued on page 37

Telephone: (615) 842-4600 **South East Media** *Telex: 5106006630*

*OS-9, UniFLEX, FLEX, SK*DOS*

SOFTWARE

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

SCULPTOR

Full OEM & Dealer Discounts Available!

THE SCULPTOR SYSTEM
Sculptor combines a powerful fourth generation language with an efficient database management system. Programmers currently using traditional languages such as Basic and Cobol will be amazed at what Sculptor does to their productivity. With Sculptor you'll find that what used to take a week can be achieved in just a few hours.

AN ESTABLISHED LEADER
Sculptor was developed by professionals who needed a software development tool with capabilities that were not available in the software market. It was launched in 1981 and since then, with feedback from an ever-increasing customer base, Sculptor has been refined and enhanced to become one of the most adaptable, fast, and above all reliable systems on the market today.

SYSTEM INDEPENDENCE
Sculptor is available on many different machines and for most operating systems, including MS DOS, Unix/Xenix and VMS. The extensive list of supported hardware ranges from small personal computers, through multi-user micros up to large minis and mainframes. Sculptor is constantly being ported to new systems.

APPLICATION PORTABILITY
Mobility of software between different environments is one of Sculptor's major advantages. You can develop applications on a stand-alone PC and – without any alterations to the programs – run them on a large multi-user system. For software writers this means that their products can reach a wider marketplace than ever before. It is this system portability, together with high-speed development, that makes Sculptor so appealing to value added resellers, hardware manufacturers and software developers of all kinds.

SPEED AND EFFICIENCY
Sculptor uses a fast and proven indexing technique which provides instant retrieval of data from even the largest of files. Sculptor's fourth generation language is compiled to a compact intermediate code which executes with impressive speed.

INTERNATIONALLY ACCEPTED
By using a simple configuration utility, Sculptor can present information in the language and format that you require. This makes it an ideal product for software development almost anywhere in the world. Australasia, the Americas and Europe – Sculptor is already at work in over 20 countries.

THE PACKAGE
With every development system you receive:
- ☐ A manual that makes sense
- ☐ A periodic newsletter
- ☐ Screen form language
- ☐ Report generator
- ☐ Menu system
- ☐ Query facility
- ☐ Set of utility programs
- ☐ Sample programs

For resale products, the run-time system is available at a nominal cost.

Facts **Features**

Sculptor for 68020
OS-9 & UniFLEX
$995

DATA DICTIONARY
Each file may have one or more record types described. Fields may have a name, heading, type, size, format and validation list. Field type may be chosen from:
- ☐ alphanumeric
- ☐ integer
- ☐ floating point
- ☐ money
- ☐ date

DATA FILE STRUCTURE
- ☐ Packed, fixed-length records
- ☐ Money stored in lower currency unit
- ☐ Dates stored as integer day numbers

INDEXING TECHNIQUE
Sculptor maintains a B-tree index for each data file. Program logic allows any numbers of alternative indexes to be coded into one other file.

INPUT DATA VALIDATION
Input data may be validated at three levels:
- ☐ automatic by field type
- ☐ validation list in data dictionary
- ☐ programmer coded logic

ARITHMETIC OPERATORS

–	Unary minus
*	Multiplication
/	Division
%	Remainder
+	Addition
–	Subtraction

RELATIONAL OPERATORS

=	Equal to
<	Less than
>	Greater than
< =	Less than or equal to
> =	Greater than or equal to
< >	Not equal to
and	Logical and
or	Logical or
ct	Contains
bw	Begins with

SPECIAL FEATURES
- ☐ Full date arithmetic
- ☐ Echo suppression for passwords
- ☐ Terminal and printer independence
- ☐ Parameter passing to sub-programs
- ☐ User definable date format

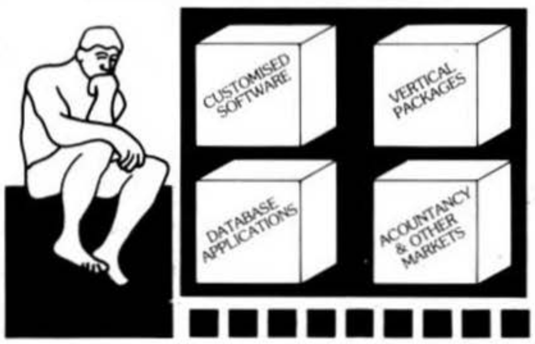
MAXIMA AND MINIMA

Minimum key length	1 byte
Maximum key length	160 bytes
Minimum record length	3 bytes
Maximum record length	32767 bytes
Maximum fields per record	32767
Maximum records per file	16 million
Maximum files per program	16
Maximum open files	Operating system limit

PROGRAMS
- ☐ Define record layout
- ☐ Create new indexed file
- ☐ Generate standard screen-form program
- ☐ Generate standard report program
- ☐ Compile screen-form program
- ☐ Compile report program
- ☐ Screen-form program interpreter
- ☐ Report program interpreter
- ☐ Menu interpreter

- ☐ Query facility
- ☐ Reformat file
- ☐ Check file integrity
- ☐ Rebuild index
- ☐ Alter language and date format
- ☐ Setup terminal characteristics
- ☐ Setup printer characteristics

SCREEN-FORM LANGUAGE
- ☐ Programmer defined options and logic
- ☐ Multiple files open in one program
- ☐ Default or programmer processing of exception conditions
- ☐ Powerful verbs for input, display and file access
- ☐ Simultaneous display of multiple records
- ☐ Facility to call sub-programs and operating system commands
- ☐ Conditional statements
- ☐ Subroutines
- ☐ Independent of terminal type



MUSTANG-020 Users - Ask For Your Special Discount!

* Tandy CoCo III Special - Reg. $595 * Special $389 *

	*	**	***		*	**	***
MUSTANG-020	$995	$199	$595	PC/XT/AT MSDOS	$595	$119	$595
OS/9 UniFLEX 6809	"	"	"	AT&T 3B1 UNIX	"	"	"
IBM Compatibles	"	"	"	SWTPC 68010 UniF	$1595	$319	$797
Tandy CoCo III	Special $389.00			SWTPC 68010 UNIX	$1990	$398	$995

... Sculptor Will Run On Over 100 Other Types of Machines ...
... Call for Pricing ...
!!! Please Specify Your Make of Computer and Operating System !!!

* Full Developement Package
** Run Time Only
*** C Key File Library

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343
Telephone: (615) 842-4600 Telex 5106006630

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

DISASSEMBLERS

SUPER SLEUTH from Computer Systems Consultants Interactive Disassembler; extremely *POWERFUL!* Disk File Binary/ASCII Examine/Change, Absolute or FULL Disassembly. XREF Generator, Label "Name Changer", and Files of "Standard Label Names" for different Operating Systems.
Color Computer SS-50 Bus (all w/ A.L. Source)
CCD (32K Req'd) Obj. Only $49.00
F, S, $99.00 - CCF, Obj. Only $50.00 U, $100.00
CCF, w/Source $99.00 O, $101.00
CCO, Obj. Only $50.00
OS9 68K Obj. $100.00 w/Source $200.00

DYNAMITE+ -- Excellent standard "Batch Mode" Disassembler. Includes XREF Generator and "Standard Label" Files. Special OS-9 options w/ OS-9 Version.
CCF, Obj. Only $100.00 - CCO, Obj. $ 59.95
F, S, " " $100.00 - O, object only $150.00
U, " " $300.00

PROGRAMMING LANGUAGES

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

PASC from S.E. Media - A FLEX9, SK*DOS Compiler with a definite Pascal "flavor". Anyone with a bit of Pascal experience should be able to begin using PASC to good effect in short order. The PASC package comes complete with three sample programs: ED (a syntax or structure editor), EDITOR (a simple, public domain, screen editor) and CHESS (a simple chess program). The PASC package come complete with source (written in PASC) and documentation.
*FLEX, SK*DOS $95.00*

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

KANSAS CITY BASIC from S.E. Media - *Basic for Color Computer OS-9* with many new commands and sub-functions added. A full implementation of the IF-THEN-ELSE logic is included, allowing nesting to 255 levels. Strings are supported and a subset of the usual string functions such as LEFT$, RIGHT$, MID$, STRING$, etc. are included. Variables are dynamically allocated. Also included are additional features such as Peek and Poke. A must for any Color Computer user running OS-9.
CoCo OS-9 $39.95

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

C Compiler from Introl -- Full C except Doubles and Bit Fields, streamlined for the 6809. Reliable Compiler; FAST, efficient Code. More UNIX Compatible than most.
*FLEX, SK*DOS, CCF, OS-9 (Level II ONLY), U - $575.00*

PASCAL Compiler from Lucidata -- ISO Based P-Code Compiler. Designed especially for Microcomputer Systems. Allows linkage to Assembler Code for maximum flexibility.
F, S and CCF 5" - $99.95 F, S 8" - $99.95

PASCAL Compiler from OmegaSoft (now Certified Software) -- For the *PROFESSIONAL;* ISO Based, Native Code Compiler. Primarily for Real-Time and Process Control applications. Powerful; Flexible. Requires a "Motorola Compatible" Relo. Asmb. and Linking Loader.
F, S and CCF - $425.00 - One Year Maint. $100.00
OS-9 68000 Version - $900.00

KBASIC - from S.E. MEDIA -- A "Native Code" BASIC Compiler which is now Fully TSC XBASIC compatible. The compiler compiles to Assembly Language Source Code. A NEW, streamlined, Assembler is now included allowing the assembly of LARGE Compiled K-BASIC Programs. Conditional assembly reduces Run-time package.
*FLEX, SK*DOS, CCF, OS-9 Compiler /Assembler $99.00*

CRUNCH COBOL from S.E. MEDIA -- Supports large subset of ANSII Level 1 *COBOL* with many of the useful Level 2 features. Full FLEX, SK*DOS File Structures, including Random Files and the ability to process Keyed Files. Segment and link large programs at runtime, or implemented as a set of overlays. The System requires 56K and CAN be run with a single Disk System. *A very popular product.*
*FLEX, SK*DOS, CCF - $99.95*

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

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CC0 = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

FORTHBUILDER is a stand-alone target compiler (crosscompiler) for producing custom Forth systems and application programs. All of the 83-standard defining words and control structures are recognized by FORTHBUILDER.
FORTHBUILDER is designed to behave as much as possible like a resident Forth interpreter/compiler, so that most of the established techniques for writing Forth code can be used without change.
Like compilers for other languages, FORTHBUILDER can operate in "batch mode".
The compiler recognizes and emulates target names defined by CONSTANT or VARIABLE and is readily extended with "compile-time" definitions to emulate specific target words.
FORTHBUILDER is supplied as an executable command file configured for a specific host system and target processor. Object code produced from the accompanying model source code is royalty-free to licensed users.
F, CCF, S - $99.95

DATABASE ACCOUNTING

XDMS from Westchester Applied Business Systems
FOR 6809 FLEX-SK*DOS(5/8")
Up to 32 groups/fields per record! Up to 12 character filed name! Up to 1024 byte records! User defined screen and print control! Process files! Form files! Conditional execution! Process chaining! Upward/ Downward file linking! File joining! Random file virtual paging! Built in utilities! Built in text line editor! Fully session oriented! Enhanced forms! Boldface, Double width, Italics and Underline supported! Written in compact structured assembler! Integrated for FAST execution!
XDMS-IV Data Management System
XDMS-IV is a brand new approach to data management. It not only permits users to describe, enter and retrieve data, but also to process entire files producing customized reports, screen displays and file output. Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotaling, and presentation of up to three related files as a "database" on user defined output reports.
POWERFUL COMMANDS!
XDMS-IV combines the functionality of many popular DBMS software systems with a new easy to use command set into a single integrated package. We've included many new features and commands including a set of general file utilities. The processing commands are Input-Process-Output (IPO) oriente which allows almost instant implementation of a process design.

SESSION ORIENTED!
XDMS-IV is session oriented. Enter "XDMS" and you are in instant command of all the features. No more waiting for a command to load in from disk! Many commands are immediate, such as CREATE (file definition), UPDATE (file editor), PURGE and DELETE (utilities). Others are process commands which are used to create a user process which is executed with a RUN command. Either may be entered into a "process" file which is executed by an EXECUTE statement Processes may execute other processes, or themselves, either conditionally or unconditionally. Menus and screen prompts are easily coded, and entire user applications can be run without ever leaving XDMS-IV
IT'S EASY TO USE!
XDMS-IV keeps data management simple! Rather than design a complex DBMS which hides the true nature of the data, we kept XDMS-IV file oriented. The user view of data relationships is presented in reports and screen output, while the actual data resides in easy to maintain files. This aspect permits customized presentation and reports without complex redefinition of the database files and structure. XDMS-IV may be used for a wide range of applications from simple record management systems (addresses, inventory ...) to integrated database systems (order entry, accounting...)
The possibilities are unlimited...
FOR 6809 FLEX-SK*DOS(5/8") **$249.95**

ASSEMBLERS

ASTRUK09 from S.E. Media -- A "Structured Assembler for the 6809" which requires the TSC Macro Assembler.
F, S, CCF - $99.95
Macro Assembler for TSC -- The FLEX, SK*DOS STANDARD Assembler.
Special -- CCF $35.00; F, S $50.00
OSM Extended 6809 Macro Assembler from Lloyd I/O. -- Provides local labels, Motorola S-records, and Intel Hex records; XREF. GeneÒrate OS-9 Memory modules under FLEX, SK*DOS.
*FLEX, SK*DOS, CCF, OS-9 $99.00*
Relocating Assembler/Linking Loader from TSC. -- Use with many of the C and Pascal Compilers.
F, S, CCF $150.00
MACE, by Graham Trott from Windrush Micro Systems -- Co-Resident Editor and Assembler; fast interactive A.L. Programming for small to medium-sized Programs.
F, S, CCF - $75.00
XMACE -- MACE w/Cross Assembler for 6800/1/2/3/8
F, S, CCF - $98.00

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

**** Shipping ****
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

***OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.**

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

UTILITIES

Basic09 XRef from S.E. Media – This Basic09 Cross Reference Utility is a Basic09 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also includes a Program List Utility which outputs a fast "pretty printed" listing with line numbers. Requires Basic09 or RunB.
O & CCO obj. only -- $39.95; w/ Source - $79.95

BTree Routines - Complete set of routines to allow simple implementation of keyed files - *for your programs* - running under Basic09. A real time saver and should be a part of every serious programmers tool-box.
O & CCO obj. only - $89.95

Lucidata PASCAL UTILITIES (Requires Pascal ver 3)

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

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

PROFILER -- provides an Indented, Numbered, "Structogram" of a Pascal Source Text File; view the overall structure of large programs, program integrity, etc. Supplied in Pascal Source Code; requires compilation.
F, S, CCF --- EACH 5" - $40.00, 8" - $50.00

DUB from S.E. Media – A UniFLEX BASIC decompiler Re-Create a Source Listing from UniFLEX Compiled basic Programs. Works w/ ALL Versions of 6809 UniFLEX basic.
U - $219.95

LOW COST PROGRAM KITS from Southeast Media The following kits are available for FLEX, SK*DOS on either 5" or 8" Disk.

1. BASIC TOOL-CHEST $29.95
   BLISTER.CMD: pretty printer
   LINEXREF.BAS: line cross-referencer
   REMPAC.BAS, SPCPAC.BAS, COMPAC.BAS: remove superfluous code
   STRIP.BAS: superfluous line-numbers stripper
2. FLEX, SK*DOS UTILITIES KIT $39.99
   CATS. CMD: alphabetically-sorted directory listing
   CATD.CMD: date-sorted directory listing
   COPYSORT.CMD: file copy, alphabetically
   COPYDATE.CMD: file copy, by date-order
   FILEDATE.CMD: change file creation date
   INFO.CMD (& INFOGMX.CMD): tells disk attributes &contents
   RELINK.CMD (& RELINK82): re-orders fragmented free chain
   RESQ.CMD: undeletes (recovers) a deleted file
   SECTORS.CMD: show sector order in free chain
   XL.CMD: super text lister
3. ASSEMBLERS/DISASSEMBLERS UTILITIES $39.95
   LINEFEED.CMD: 'modularise' disassembler output
   MATH.CMD: decimal, hex, binary, octal conversions & tables
   SKIP.CMD: column stripper
4. WORD - PROCESSOR SUPPORT UTILITIES $49.95
   FULLSTOP.CMD: checks for capitalization
   BSTYCIT.BAS (.BAC): Stylo to dot-matrix printerr
   NECPRINT.CMD: Stylo to dot-matrix printer filter code
5. UTILITIES FOR INDEXING $49.95
   MENU.BAS: selects required program from list below
   INDEX.BAC: word index
   PHRASES.BAC: phrase index
   CONTENT.BAC: table of contents
   INDXSORT.BAC: fast alphabetic sort routine
   FORMATER.BAC: produces a 2-column formatted index
   APPEND.BAC: append any number of files
   CHAR.BIN: line reader

BASIC09 TOOLS consist of 21 subroutines for Basic09. 6 were written in C Language and the remainder in assembly. All the routines are compiled down to native machine code which makes them fast and compact.

1. CFILL -- fills a string with characters
2. DPEEK – Double peek
3. DPOKE – Double poke
4. FPOS -- Current file position
5. FSIZE -- File size
6. FTRIM -- removes leading spaces from a string
7. GETPR – returns the current process ID
8. GETOPT – gets 32 byte option section
9. GETUSR – gets the user ID
10. GTIME -- gets the time
11. INSERT -- insert a string into another
12. LOWER – converts a string into lowercase
13. READY – Checks for available input
14. SETPRIOR -- changes a process priority
15. SETUSR – changes the user ID
16. SETOPT -- set 32 byte option packet
17. STIME -- sets the time
18. SPACE -- adds spaces to a string
19. SWAP – swaps any two variables
20. SYSCALL – system call
21. UPPER -- converts a string to uppercase

For OS-9 - $44.95 - Includes Source Code
See Review in January 1987 issue of 68 Micro Journal

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

•• Shipping ••
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

SOFTOOLS

The following programs are included in object form for immediate application. PL/9 source code available for customization.

READ-ME Complete instructions for initial set-up and operation. Can even be printed out with the included text processor.
CONFIG one time system configuration.
CHANGE changes words, characters, etc. globally to any text type file.
CLEANTXT converts text files to standard FLEX, SK*DOS files.
COMMON compare two text files and reports differences.
COMPARE another check file that reports mis-matched lines.
CONCAT similar to FLEX, SK*DOS append but can also list files to screen.
DOCUMENT for PL/9 source files. Very useful in examining parameter passing aspects of procedures.
ECHO echos to either screen or file.
FIND an improve find command with "pattern" matching and wildcards. Very useful.
HEX dumps files in both hex and ASCII.
INCLUDE a file copy program that will accept "includes" of other disk files.
KWIC allows rotating each word, on each line to the beginning. Very useful in a sort program, etc.
LISTDIR a directory listing program. Not super, but better than CAT.
MEMSORT a high-speed text file sorter. Up to 10 fields may be sorted. Very fast. Very useful.
MULTICOL width of page, number of columns may be specified. A MUST!
PAGE similar to LIST but allows for a page header, page width and depth. Adjust for CRT screen or printer as set up by CONFIG. A very smart print driver. Allows printer control commands.
REMOVE a fast file deleter. Careful, no prompts issued. Zap, and its gone!
SCREEN a screen listing utility. Word wraps text to fit screen. Screen depth may be altered at run time.
SORT a super version of MEMSORT. Ascending/descending order, up to 10 keys, case over-ride, sort on nth word and sort on characters if file is small enough, sorts in RAM. If large file, sort is constrained to size of your largest disk capacity.
TPROC a small but nice text formatter. This is a complete formatter and has functions not found in other formatters.
TRANSLIT sorts a file by x keyfields. Checks for duplications. Up to 10key files may be used.
UNROTATE used with KWIC this program reads an input file and unfolds it a line at a time. If the file has been sorted each word will be presented in sequence.
WC a word count utility. Can count words, characters or lines.

NOTE: this set of utilities consists of 6 5-1/4" disks or 2 8" disks, w/ source (PL9). 3 5-1/4" disks or 1 8" disk w/o source.
Complete set SPECIAL INTRO PRICE:
5-1/4" w/source FLEX - SK*DOS - $129.95
w/o source - $79.95
8" w/source - $79.95 - w/o source $49.95

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

SOLVE from S.E. Media - OS-9 Levels I and II only. A Symbolic Object/Logic Verification & Examine debugger. Including inline debugging, disassemble and assemble. SOLVE IS THE MOST COMPLETE DEBUGGER we have seen for the 6809 OS-9 series! SOLVE does it all! With a rich selection of monitor, assembler, disassembler, environmental, execution and other miscellaneous commands, SOLVE is the MOST POWERFUL tool-kit item you can own! Yet, SOLVE is simple to use! With complete documentation, a snap! Everyone who has ordered this package has raved! See review - 68 Micro Journal - December 1985. No 'blind' debugging here, full screen displays, rich and complete in information presented. Since review in 68 Micro Journal, this is our fastest mover!
Levels I & II only - OS-9 $69.95

DISK UTILITIES

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

O-F from S.E. Media -- Written in BASIC09 (with Source), includes: REFORMAT, a BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX, SK*DOS Format so it can be used normally by FLEX, SK*DOS; and FLEX, a BASIC09 Program that does the actual read or write function to the special O-F Transfer Disk; user-friendly menu driven. Read the FLEX, SK*DOS Directory, Delete FLEX, SK*DOS Files, Copy both directions, etc. FLEX, SK*DOS users use the special disk just like any other FLEX, SK*DOS disk
O - 6809/68000 $79.95

LSORT from S.E. Media - A SORT/MERGE package for OS-9 (Level I & II only). Sorts records with fixed lengths or variable lengths. Allows for either ascending or descending sort. Sorting can be done in either ASCII sequence or alternate collating sequence. Right, left or no justification of data fields available. LSORT includes a full set of comments and errors messages.
OS-9 $85.00

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CC0 = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

**** Shipping ****
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600

South East Media

Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

HIER from S.E. Media - *HIER is a modern hierarchal storage system for users under FLEX, SK*DOS.* It answers the needs of those who have hard disk capabilities on their systems, or many files on one disk - any size. **Using HIER a regular (any) FLEX, SK*DOS disk (8 - 5 - hard disk) can have sub directories.** By this method the problems of assigning unique names to files is less burdensome. Different files with the exact same name may be on the same disk, as long as they are in different directories. For the winchester user this becomes a must. Sub-directories are the modem day solution that all current large systems use. **Each directory looks to FLEX, SK*DOS like a regular file, except they have the extension '.DIR'.** A full set of directory handling programs are included, making the operation of HIER simple and straightforward. A special install package is included to install HIER to your particular version of FLEX, SK*DOS. Some assembly required. Install indicates each byte or reference change needed. Typically - 6 byte changes in source (furnished) and one assembly of HIER is all that is required. No programming required!
*FLEX - SK*DOS $79.95*

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

COPYCAT from Lucidata -- *Pascal NOT required.* Allows reading TSC Mini-FLEX, SK*DOS, SSB DOS68, and Digital Research CP/M Disks while operating under SK*DOS, FLEX1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Also includes some Utilities to help out. Programs supplied in Modular Source Code (Assembly Language) to help solve unusual problems.
F, S and CCF 5" - $50.00   F, S 8" - $65.00

VIRTUAL TERMINAL from S.E. Media - Allows one terminal to do the work of several. The user may start as many as eight task on one terminal, under *VIRTUAL TERMINAL* and switch back and forth between task at will. No need to exit each one; just jump back and forth. Complete with configuration program. The best way to keep up with those background programs.
O & CCO - obj. only - $49.95

FLEX, SK*DOS DISK UTILITIES from Computer Systems Consultants -- Eight (8) different Assembly Language (w/ Source Code) FLEX, SK*DOS Utilities for every FLEX, SK*DOS Users Toolbox: Copy a File with CRC Errors; Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sectors; Linearize Free-Chain on the Disk; print Disk Identification; and Sort and Replace the Disk Directory (in sorted order). -- PLUS -- Ten XBASIC Programs including: A BASIC Resequencer with EXTRAs over "RENUM" like check for missing label definitions, processes Disk to Disk instead of in Memory, etc. Other programs Compare, Merge, or Generate Updates between two BASIC Programs, check BASIC Sequence Numbers, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files. A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, XBASIC, and PRECOMPILER BASIC Programs.
ALL Utilities include Source≥ (either BASIC or A.L. Source Code).
F, S and CCF - $50.00
BASIC Utilities ONLY for UniFLEX -- $30.00

COMMUNICATIONS

CMODEM Telecommunications Program from Computer Systems Consultants, Inc. -- Menu-Driven; supports Dumb-Terminal Mode, Upload and Download in non-protocol mode, and the CP/M "Modem7" Christensen protocol mode to enable communication capabilities for almost any requirement. Written in "C".
*FLEX, SK*DOS, CCF, OS-9, UniFLEX, 68000 & 6809A*
Source $100.00 - without Source $50.00

X-TALK from S.E. Media - X-TALK consists of two disks and a special cable, the hookup enables a 6809 SWTPC computer to dump UniFLEX files directly to the UniFLEX MUSTANG-020. This is the ONLY currently available method to transfer SWTPC 6809 UniFLEX files to a 68000 UniFLEX system. Gimix 6809 users may dump a 6809 UniFLEX file to a 6809 UniFLEX five inch disk and it is readable by the MUSTANG-020. The cable is specially prepared with internal connections to match the non-standard SWTPC SO/9 I/O Db25 connectors. A special SWTPC S+ cable set is also available. Users should specify which SWTPC system he/she wishes to communicate with the MUSTANG-020. The X-TALK software is furnished on two disks. One eight inch disk contains S.E. Media modem program C-MODEM (6809) and the other disk is a MUSTANG-020 five inch disk with C-MODEM (68020). Text and binary files may be directly transferred between the two systems. The C-MODEM programs are unaltered and perform as excellent modem programs also. X-TALK can be purchased with or without the special cables, but this special price is available to registered MUSTANG-020 users only.
X-TALK Complete (cable, 2 disks) $99.95
X-TALK Software (2 disks only) $69.95
X-TALK with CMODEM Source $149.95

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

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

EDITORS & WORD PROCESSING

JUST from S.E. Media – Text Formatter developed by Ron Anderson; for Dot Matrix Printers, provides many unique features. Output "Formatted" Text to the Display. Use the FPRINT.CMD supplied for producing multiple copies of the "Formatted" Text on the Printer INCLUDING EMBEDDED PRINTER COMMANDS (very useful at other times also, and worth the price of the program by itself). "User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); up to ten (10) imbedded "Printer Control Commands". Compensates for a "Double Width" printed line. Includes the nonnal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc. Use with PAT or any other editor.
• Now supplied as a two disk set:
Disk #1: JUST2.CMD object file,
*JUST2.TXT PL9 source:FLEX, SK*DOS - CC*
Disk #2: JUSTSC object and source in C:
*FLEX, SK*DOS - OS9 - CC*
The JTSC and regular JUST C source are two separate programs. JTSC compiles to a version that expects TSC Word Processor type commands, (.pp .sp .ce etc.) Great for your older text files. The C source compiles to a standard syntax JUST.CMD object file. Using JUST syntax (.p .u .y etc.) With all JUST functions plus several additional printer formatting functions. Reference the JUSTSC C source. For those wanting an excellent BUDGET PRICED word processor, with features none of the others have. This is it!
Disk (1) - PL9 FLEX only- F, S & CCF - $49.95
Disk Set (2) - F, S & CCF & OS9 (C version) - $69.95
OS-9 68K000 complete with Source - $79.95

PAT from S.E. Media - A full feature screen oriented TEXT EDITOR with all the best of "PIE™". For those who swore by and loved only PIE, this is for you! All PIE features and much more! Too many features to list. And if you don't like these, change or add your own. PL-9 source furnished. "C" source available soon. Easily configured to your CRT, with special config section.
*Regular FLEX, SK*DOS $129.50*
• SPECIAL INTRODUCTION OFFER • *$79.95*
SPECIAL PAT/JUST COMBO (w/source)
*FLEX, SK*DOS $99.95*
OS-9 68K Version $229.00
SPECIAL PAT/JUST COMBO 68K $249.00
Note: *JUST in "C" source available for OS-9*

CEDRIC from S.E. Media - A screen oriented TEXT EDITOR with availability of 'MENU' aid. Macro definitions, configurable 'permanent definable MACROS' - all standard features and the fastest 'global' functions in the west. A simple, automatic terminal config program makes this a real 'no hassel' product. Only 6K in size, leaving the average system over 165 sectors for text buffer - appx. 14,000 plus of free memory! Extra fine for programming as well as text.
*FLEX, SK*DOS $69.95*

BAS-EDIT from S.E. Media - A TSC BASIC or XBASIC screen editor. Appended to BASIC or XBASIC, BAS-EDIT is transparent to normal BASIC/XBASIC operation. Allows editing while in BASIC/XBASIC. Supports the following functions: OVERLAY, INSERT and DUP LINE. Make editing BASIC/XBASIC programs SIMPLE! A GREAT time and effort saver. Programmers love it! NO more retyping entires lines, etc. Complete with over 25 different CRT terminal configuration overlays.
*FLEX, CCF, SK*DOS $39.95*

SCREDITOR III from Windrush Micro Systems – Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; over 300 pages of Documentation with Tutorial. Features Multi-Column display and editing, "decimal align" columns (AND add them up automatically), multiple keystroke macros, even/odd page headers and footers, imbedded printer control codes, all justifications, "help" support, store common command series on disk, etc. Use supplied "set-ups", or remap the keyboard to your needs. Except for proportional printing, this package will DO IT ALL!
*6800 or 6809 FLEX, SK*DOS or SSB DOS, OS-9 - $175.00*

SPELLB "Computer Dictionary" from S.E. Media – OVER 150,000 words! *Look up a word from within your Editor or Word Processor (with the SPH.CMD Utility which operates in the FLEX, SK*DOS UCS).* Or check and update the Text after entry; ADD WORDS to the Dictionary, "Flag" questionable words in the Text, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of Small Disk Storage systems.
F, S and CCF - $129.95

STYLO-GRAPH from Great Plains Computer Co. – A full-screen oriented WORD PROCESSOR – (uses the 51 x 24 Display Screens on CoCo FLEX/SK*DOS, or PBJ Wordpak). Full screen display and editing; supports the Daisy Wheel proportional printers.
NEW PRICES 6809 CCF and CCO - $99.95,
F, S or O - $179.95, U - $299.95

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

mastercard

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600 **South East Media** Telex: 5106006630

OS-9, UniFLEX, FLEX, SK*DOS

STYLO-SPELL from Great Plains Computer Co. -- Fast Computer Dictionary. Complements Stylograph.
NEW PRICES 6809 CCF and CCO - $69.95,
F, S or O - $99.95, U - $149.95

STYLO-MERGE from Great Plains Computer Co. -- Merge Mailing List to "Form" Letters, Print multiple Files, etc., through Stylo.
NEW PRICES 6809 CCF and CCO - $59.95,
F, S or O - $79.95, U - $129.95

STYLO-PAK --- Graph + Spell + Merge Package Deal!!!
F, S or O - $329.95, U - $549.95
O, 68000 $695.00

MISCELLANEOUS

TABULA RASA SPREADSHEET from Computer Systems Consultants -- TABULA RASA is similar to DESKTOP/PLAN; provides use of tabular computation schemes used for analysis of business, sales, and economic conditions. Menu-driven; extensive report-generation capabilities. Requires TSC's Extended BASIC.
F, S and CCF, U - $50.00, w/ Source - $100.00

DYNACALC -- Electronic Spread Sheet for the 6809 and 68000.
F, S, OS-9 and SPECIAL CCF - $200.00, U - $395.00
OS-9 68K - $595.00

FULL SCREEN INVENTORY/MRP from Computer Systems Consultants -- Use the Full Screen Inventory System/Materials Requirement Planning for maintaining inventories. Keeps item field file in alphabetical order for easier inquiry. Locate and/or print records matching partial or complete item, description, vendor, or attributes; find backorder or below stock levels. Print-outs in item or vendor order. MRP capability for the maintenance and analysis of Hierarchical assemblies of items in the inventory file. Requires TSC's Extended BASIC.
F, S and CCF, U - $50.00, w/ Source - $100.00

FULL SCREEN MAILING LIST from Computer Systems Consultants -- The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Locate all records matching on partial or complete name, city, state, zip, or attributes for Listings or Labels, etc. Requires TSC's Extended BASIC.
F, S and CCF, U - $50.00, w/ Source - $100.00

DIET-TRAC Forecaster from S.E. Media -- 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 Carbohydrates, Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual. Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. Provides number of days and daily calendar after weight goal and calorie plan is determined.
F, S - $59.95, U - $89.95

CROSS ASSEMBLERS

TRUE CROSS ASSEMBLERS from Computer Systems Consultants -- Supports 1802/5, Z-80, 6800/1/2/3/8/11/HC11, 6804, 6805/HC05/146805, 6809/00/01, 6502 family, 8080/5, 8020/1/2/35/C35/39/40/48/C48/49/C49/50/8748/49, 8031/51/8751, and 68000 Systems. Assembler and Listing formats same as target CPU's format. Produces machine independent Motorola S-Text.
*68000 or 6809, FLEX, SK*DOS, CCF, OS-9, UniFLEX*
any object or source each - $50.00
any 3 object or source each - $100.00
Set of ALL object $200.00 - w/source $500.00

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

CRASMB from LLOYD I/O -- Supports Motorola's, Intel's, Zilog's, and other's CPU syntax for these 8-Bit microprocessors: 6800, 6801, 6303, 6804, 6805, 6809, 6811 (all varieties); 6502, 1802/5, 8048 family, 8051 family, 8080/85, Z8, Z80, and TMS-7000 family. Has MACROS, Local Labels, Label X-REF, Label Length to 30 Chars. Object code formats: Motorola S-Records (text), Intel HEX-Records (text), OS9 (binary), and FLEX, SK*DOS (binary). Written in Assembler ... e.g. Very Fast.

CPU TYPE - Price each:

For:	MOTOROLA	INTEL	OTHER	COMPLETE SET
FLEX9	$150	$150	$150	$399
SK*DOS	$150	$150	$150	$399
OS9/6809	$150	$150	$150	$399
OS9/68K	------	------	------	$432

CRASMB 16.32 from LLOYD I/O -- Supports Motorola's 68000, and has same features as the 8 bit version. OS9/68K Object code Format allows this cross assembler to be used in developing your programs for OS9/68K on your OS9/6809 computer.
*FLEX, SK*DOS, CCF, OS-9/6809 $249.00*

GAMES

RAPIER - 6809 Chess Program from S.E. Media -- Requires FLEX, SK*DOS 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, S and CCF - $79.95

Availability Legends
O = OS-9, S = SK*DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CCF = Color Computer FLEX

VISA

South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343

master card

** Shipping **
Add 2% U.S.A. (min. $2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola-*FLEX and UniFLEX are Trademarks of Technical Systems Consultants-*SK*DOS is a Trademark of Star-K Software Systems Corp.

All agreed? Wow!! I almost got deafened by the chorus of 'Yes, Yes' from all you readers out there! Well, how about substituting 'no cat' for 'Tom', 'one cat' for 'Dick', and 'legs' for 'marbles'? Now we have :

1. No cat has 95 legs.
2. One cat has 4 more legs than no cat.
3. Therefore one cat has 99 legs.

I think EVERYONE will agree that Statement 1 is TRUE, as is Statement 2. Let's ignore the odd cat which has had an accident and only has 3 legs!! Therefore, Statement 3 must also be TRUE. Agreed? Sorry, but I didn't hear a chorus of 'Yes' this time! What's the problem? I warned you that words can be treacherous, didn't I? I'd further venture to say that if 'cat' had been replaced by 'xzirdl' (a silicate life-form found only on the outermost moon of the planet Jupiter), you'd have been more prepared to accept the "logic" of these statements. It's only because you know from experience that cats don't have 99 legs that you disagree, NOT because you detect a flaw in the "logic". Why then is Statement 3 logically TRUE for Aunt Minnie and Uncle Fred, but not for cats???

You may reasonably ask what all this has to do with designing digital control circuits. My main purpose is to demonstrate why we convert words to a mathematical form when we design control circuits. 9 times out of 10, a set of written or verbal instructions spelling out the way in which a machine has to behave can be deceptively "logical" (as we'll see later on in this series), sometimes to the point where it's actually IMPOSSIBLE to design the specified machine or circuit!

I'll let you ponder the cat problem until you've got a bit more "logic" under your belts, but in the meantime just imagine - - if it had just ONE more leg our poor pussy-cat would be a centipede!!

Now we're ready

OK, I think we're more or less ready now to begin exploration of a strange new world, where 1 + 1 is sometimes equal to 2, at other times equal to 10, and on occasion even equal to 1. But never fear, your trusty guide, Bob, (that's me) has been this way many times before, and will surely help you over any rough patches we come across during our travels.

I'll try to keep the subject-matter as light-hearted as possible, but don't be misled by my method — this is serious stuff, and I'd like you to make every effort to really grasp what I'm talking about.

Our equipment is no more than a pad of squared paper (1/4 inch squares will do admirably) and a sharpened pencil (maybe with an eraser at the other end). So .... if we're all set, let's be on our way.

INTRODUCTION

The aim of this series is to provide you with an understanding of the mathematical basis of control-circuit design, to the extent that you should be able to commence with an initial set of specifications (written or verbal) setting out the intended function of the required machine. Then you'll convert these specs to a mathematical form which you'll manipulate according to certain rules, and from the resulting expressions construct the circuit diagram of the machine, using either relays or transistor logic-circuits as the case may be.

Initially, we'll restrict our attention to control-circuits using relays only, until we've become quite familiar with the various techniques, and then, and then only, shall we consider using transistor logic. It's MUCH easier to see whether a network of relay-contacts will conduct an electrical current than it is to decide whether the equivalent "logic-circuit" network will output a 'high' or 'low' voltage for a given set of input conditions.

By the way, in case you've never thought about it before, they are called "logic" circuits because their design is based on the use of mathematical logic.

I must emphasize that the methods to be taught are, by and large, tools only, and it is entirely up to you how proficient you become in their use. It still requires skill and experience to become a good designer!

The beginning of mathematical designs of digital circuits - George Boole.........

The story of the mathematical method of designing digital control circuits really begins in 1847, when George Boole published his book "The Mathematical Analysis of Logic". Up to this time, the study of logic (word-logic, that is) tended to bog down in confusing masses of words, which, with their attendant emotional content, only served to conceal the underlying logic of what was being said. All of which, translated, means that you are less likely to have a logical hang-up over 'xzlrdls' than you are over 'cats' (with which you are quite familiar), and even less likely to get yourself confused with 'X' or 'Y'. No-one could possibly get emotionally involved with simple letters of the alphabet!

Several attempts had been made to develop a set of symbols for logical expression, together with the rules for manipulating these symbols, but his was the first really workable method. It became known as Boolean algebra. However, it was not until 1938 that C. E. Shannon observed that electrical-relay networks could also be expressed in Boolean algebra, and since that time its study has proceeded at an accelerating pace, producing a variety of techniques for designing control circuits, as well as for the analysis of already existing circuits.

Before we proceed further though, let's explain the word "digital". Consider a volume-control on a radio - it can be turned down so low that we can hardly hear it, or adjusted to any intermediate setting to give any desired volume, right up to the maximum of which the receiver is capable. This is an "analog" control, as it can be adjusted through a continuous, infinitely-close range of adjustments. But if our volume-control could only be "clicked" up in ten equal steps, increasing the volume by 10% at each step, it would be "digital". There would be no possibility of obtaining a sound-level of 15% for example, as this would be in between the 10% and the 20% "step". An ordinary room-light is digital (either ON or OFF) but the accelerator-pedal in a car is analog.

Mile "0" coming up

So here we are at Mile 0!

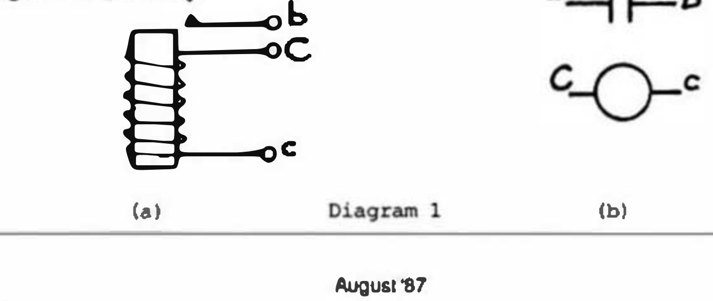
CONTROL CIRCUITS

Control circuits fall into two distinct classes - "combinational" and "sequential". A combinational circuit (which is the type we'll be examining for the first few miles of our journey) always produces the same effect for the same combination of controls. Our simple room-light is such an example, because whenever the switch is OFF the light is also OFF, and whenever the switch is ON the light is also ON - unless, of course, it has broken down.

In the case of a sequential circuit, however, a control may produce different effects, depending on the past history of the device. A good example of this is a reading-lamp with a PUSH-ON/PUSH-OFF button-switch. Here, whether the light is ON or OFF does not directly depend on whether the push-button is pressed or released. If the light is OFF right now, then pressing and releasing the button will turn it ON. But if the light is presently ON, then EXACTLY the same action will turn it OFF.

RELAYS - ELECTRICAL SYMBOLS AND CIRCUIT DIAGRAMS

I mentioned earlier that we're going to learn a lot about relay control-circuits before we tackle transistors, so what better place to start than to take a look at the little monster right here?



(a) Diagram 1 (b)

Diagram 1a shows the construction of a typical small relay, while Diagram 1b gives the standard symbols for its various parts. We have a "normally-closed" (NC) contact between A and a, through which current can flow when the relay coil is not energised, and a "normally-open" (NO) contact between B and b, through which no current can flow because of the gap between the contacts. If the coil of the relay is energised (that is, if current is caused to flow between C and c) a magnetic-field is created in the core of the coil, attracting towards itself the moveable contacts joined by the dotted line. This causes the NC-contact to open, and the NO-contact to close.

System Control

To the newcomer to this field of study it seems almost incredible that a network of such relays, connected together in a specific way, can control a machine through the most intricate processes, even doing its "thinking" for it. Some of the early computers were built up from such networks, and it's our purpose to find out how to do this to achieve any system of control that we desire.

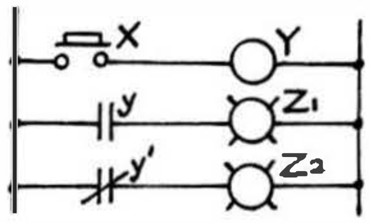


Diagram 2

Let's take a look at Diagram 2 for a moment! We see two vertical lines, and strung out between them like the rungs of a ladder are three horizontal lines with some new symbols. In order to read and understand this diagram, we must imagine electrical current coming in down the left vertical rail, and trying to find a path across to the right-hand vertical rail. In row-1 it is unable to reach the coil of relay-Y because the contact of push-button-X is open. In row-2 the current is unable to flow through light-Z1 because of the NO-contact y of relay Y, but in row-3 current can flow through the NC-contact y' of relay Y. Note that the contacts bear the same designation as the relay to which they belong, but in lower-case, y being a NO-contact and y' being a NC-contact. The NC-contact would normally be indicated by a short bar over the contact-name, but as this could very easily be confused with an underline in the line of text above, we will use the alternative form, and indicate it with a prime, thus y', y1', y2' and so on.

The fact that light-Z2 is ON tells us that relay-Y is NOT energised, or NOT operated, and so the name of the NC-contact, that is y', is read as "not-y". One could argue that the fact that light-Z1 is OFF also tells us that relay-Y is NOT energised, but this ain't necessarily so. The relay COULD be energised but the lamp itself be a dud, so from now on we'll not regard the ABSENCE of a signal as giving us reliable information.

We can readily see from our little circuit that if push-button X is depressed, current can now flow through the coil of relay-Y, as a result of which both of its contacts will operate, thus turning ON light Z1 and turning OFF light Z2. Aha, I almost forgot to mention that controls operated by humans (called primary controls) are usually given the name X, or X1, X2, etc; devices such as relays, timers and so on (called secondary controls) are named Y; and the output devices being controlled, such as lights, heaters or motors are named Z.

Truth-Tables sets the conditions

It's possible to draw up a table, known as a Truth-Table, setting out the conditions of this circuit, as shown in Diagram 3 :

Push-button X	Relay Y	y	y'	Z1	Z2
0	0	0	1	0	1
1	1	1	0	1	0

Diagram 3

Here is our first encounter with the 1s and 0s which are going to become our friends along our journey. '0' ALWAYS implies a negative, or NOT condition, such as 'NOT energised', 'NOT lit', or even a simple 'NO', while '1' ALWAYS implies the affirmative, such as 'IS energised', 'IS lit', 'IS so' or just plain 'YES'. Thus row 1 of Diagram 3 summarises the situation if X is 'NOT pressed' - Y will NOT be energised, y will NOT be conducting current while y' will be so conducting, Z1 will be OFF and Z2 will be ON. Conversely for row 2.

● ●

Now we're for more complicated circuits

Now we're ready to look at a slightly more complicated circuit in Diagram 4. Here it's quite obvious that no current can flow through any of the rows, so nothing is going to happen until a primary control is operated. If push-button X1 is operated, we can see that relay Y1 will become energised, both of its contacts y1 will close, and light Z1 alone will come on. Light Z2 will stay OFF because there is still an open y2 contact "in series with" the closed y1 contact.

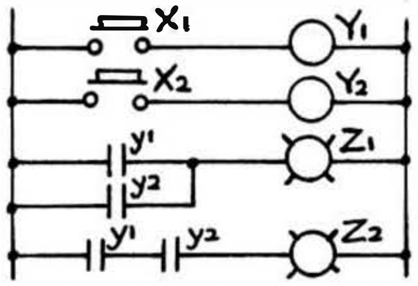


Diagram 4

(Note that this circuit uses relays which have 2 normally-open contacts each). If push-button X2 alone is operated, relay Y2 alone will be energised, the y2 contacts will close, and again only Light Z1 will come on. The contacts y1 and y2 in series with Light Z1 are said to be "in parallel" with one another.

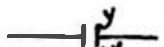
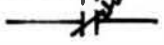
From the foregoing we can very easily see that if X1 and X2 are operated together then both relays Y1 and Y2 will become energised and both lights Z1 and Z2 will light. We would say that the control expressions for Lights Z1 and Z2 are :

Z1 = y1 OR y2 OR both

Z2 = y1 AND y2

BASIC SYMBOLS OF BOOLEAN ALGEBRA

This little section will involve very little that you don't know already, because whether you've been aware of it or not, you've learned most of them in the previous section. These are not ALL of the symbols of Boolean algebra, but they are all we need to know at this stage of our adventure, which, by the way, comes under the general heading of SWITCHING THEORY. This particular name needs no explanation from me, I'm sure. Anyway, let's take a look at the six symbols we're going to learn.

y	=		* Transmits current if Relay Y energised
y'	= *		* Does so when Relay Y is NOT energised
1	=	IS,	eg IS transmitting, IS operated, IS lit, etc
0	=	IS NOT,	eg IS NOT energised, IS NOT operated, IS NOT lit
+	=	OR (ie, in parallel with)	eg y1+y2 = y1 OR y2 OR both
.	=	AND (ie, in series with)	eg y1.y2 = y1 AND y2

This is usually shortened to y1y2 = y1 AND y2

And that's it! The first four symbols you've already met, and the last two are needed so we can describe a circuit in a completely mathematical form. For example, to describe the control network of Diagram 4, we would write :

Z1 = y +y2   Z2 = y1y2   Y1 = X1   Y2 = X2

Given these 4 expressions, and the information that the Xs are pushbuttons, it should be a VERY simple matter for us to reconstruct the original circuit-diagram. Let's pause a while here to check whether this is in fact true by trying our hand at the following :

TEST ONE

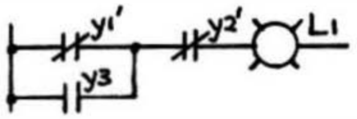
1. Draw the following circuit-diagrams, where Zs are lights -

(i)	Z1 = y1y2y3	(ii)	Y1 = y2+y3'
(iii)	Z3 = X1+y1'	(iv)	Y2 = (y1+y2)y3
(v)	Y4 = y1+y2y3	(vi)	Z2 = (y1+y2')(y3'+y4)

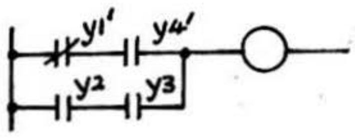
HINT :

In (iv) we understand that a y1 contact is in parallel with a y2 contact, and this little block is then in series with a y3 contact. Distinguish carefully between (iv) and (v), and watch out for those NOT signs, such as y1' etc.

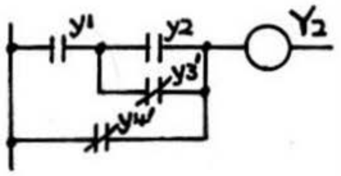
2. Write expressions for the following networks -



(i)



(ii)



(iii)

Well, here we are at Mile 1, and really, it feels *so good* resting here while you're all working away at these problems that I think I'll just relax a while before we travel on to Mile 2 of our trip. Hope you've enjoyed the scenery along the way so far. It won't always be open country like this - sometimes we'll have to hack a path through a little heavy undergrowth before we see daylight again. But keep in mind always that others have been this way before you, and to date I haven't lost a single soul. Granted, some have ended up a little worse for wear, but all have somehow managed to survive. And so shall you!!

I almost forgot to mention that answers to all problems encountered on our journey will always be supplied before taking off along the next Mile.

"Mile 2 to be continued"

Bit-Bucket

By: All of us

"Contribute Nothing · Expect Nothing", DMW '86

Continued From Last Month

XBASIC Xplained

or

Things you won't find in the documentation

Copyright 1986 by
R. Jones, Micronics Research Corp.,
33383 Lynn Avenue, Abbotsford,
British Columbia, CANADA V2S 1E2
& Computer Publishing, Inc. (CPI) © 1987

LOGIC OPERATIONS - AND, OR & NOT

Editor's Note: Due to typographical necessity the "overscore" is represented here as an "underscore" "_". We apologize for any inconvenience this may cause.

My friend Denis phoned me some time ago with a problem in one of his programs where he was experiencing difficulties in manipulating logic operators. XBASIC has three of them - AND, OR and NOT - which constitute a logically complete set, in that any logical statement can be expressed with them. In actual fact, though, because AND and OR are logically complementary, the set can be reduced to 2 operators, by using $\underline{\text{AND}}$ to represent OR, or, of course, $\underline{\text{OR}}$ to represent AND. Here the bar over AND or OR is read as 'NOT' - thus NOT AND or NOT OR. Readers already versed in the mathematics of digital logic will recognise NOT-AND as NAND, and NOT-OR as NOR, and also be aware that one of these operators alone is capable of expressing any Boolean logic function. However, let's take things one step at a time, and for the moment get back to our basic AND, OR and NOT! For the benefit of readers new to the game, perhaps I should mention that Boolean algebra is named after one George Boole, who is credited with being the first person to develop a truly consistent technique for manipulating English-language sentences in a mathematical form.

To begin with, I'll reproduce the essence of my friend's problem, and we'll just take it from there. Consider the following:

```
10 INPUT "Do you need instructions (Y or N)",Q$
20 IF Q$ = "N" OR Q$ = "n" GOTO 50
30 IF Q$ <> "Y" OR Q$ <> "y" GOTO 10
40 EXEC, "LIST GAME.INF"
50 rest of program ....
```

His intention is quite clear, namely that a response of 'N' (upper or lower-case) should bypass the LISTing of instructions by branching to Line 50. This leaves only "Y" or "y" as an alternative acceptable response, so at Line 30 the intent is that if the response is neither of these the program should loop back and repeat the request for input, otherwise it should fall through and LIST the instructions at Line 40 and then carry on with the rest of the program at Line 50.

To his surprise, he found that a negative response worked OK, branching him to Line 50, as did an invalid entry, which correctly returned him to Line 10. But, and a big but it was, a response of "Y" or "y" also behaved as an invalid response and bounced him back to Line 10! There was just no way to get the Instructions to LIST. I recall having seen similar puzzling occurrences with other people, so perhaps some explanations are in order.

OK then, let's begin by pointing out that 'AND' is mathematically equivalent to '*' (multiply) and 'OR' to '+' (add), although Boolean logic expressions usually use '.' for AND, or omit it altogether. This is also true of normal algebraic operations, where a*b*c (or a.b.c) is written as abc, or 2*x as 2x. A further point to keep in mind is that, just as in ordinary arithmetic, where 'multiply' has priority over 'add', so too does 'AND' have priority over 'OR'. Thus 2 + 3 * 4 is not simply executed from left to right to form 5 * 4 and thence 20 - - the part 3 * 4 must be executed first, producing 2 + 12 and thence 14 as the correct result. Another way of looking at this is to imagine a set of parens around the high-priority part, thus 2 + (3 * 4). Obviously, if we wished the '+' to be executed first we would write our expression as (2 + 3) * 4, as parens have a priority even higher than that of '*'.

So too with AND and OR. The expression A AND B OR C is expressed in Boolean algebra as AB + C, which is quite different from A(B + C). Let's consider AB + C a little further, perhaps how to express its inverse, that is $\underline{AB}$ + $\underline{C}$. This is performed by 'complementing' (please, not 'complimenting' as occurs all too often) everything under the NOT bar, including the implied AND between A and B! It might seem that the result should be $\underline{5}$ OR $\underline{B}$ AND $\underline{C}$, or $\underline{A}$ + $\underline{B}$. $\underline{C}$ in mathematical form, but this is not so, as we would be overlooking the fact that AB form a 'tied' pair in the original expression (remember the imaginary parens around them, as in our example 2 + 3 * 4), and must retain that relationship in the transformed expression. Thus ($\underline{A}$ OR $\underline{B}$) AND $\underline{C}$, or ($\underline{A}$ + $\underline{B}$) $\underline{C}$ is the correct answer.

One small final step, and we are ready to go back and re-examine the little program above. In spite of the fact that XBASIC (and several other BASICs too) have chosen the value '-1' to represent TRUE, Boolean algebra uses the value '1'. Everyone seems to have agreed on '0' to represent FALSE, however. Before going back to our initial problem, let's take one last look at the expression AB + C. Suppose A were TRUE, B were FALSE and C TRUE, would the complete expression evaluate as TRUE or FALSE? This now becomes very easy to decide by merely replacing the individual terms with their corresponding TRUTH values. So AB + C becomes 1.0 + 1, reducing to 0 + 1 (1 * 0 = 0) and finally 1 (0 + 1 = 1), or to perform the whole operation in one step we would say (1 * 0) + 1 = 1, giving a truth-value of TRUE. In similar vein if A were FALSE, B were TRUE and C FALSE, we would say (0 * 1) + 0 = 0, thus producing a result of FALSE for the complete expression. How much easier can this thing get?

One more important feature of Boolean algebra to remember, and that is that 1 + 1 = 1. I won't go into the reasons for that right here, as it is really outside the scope of this article. Maybe in some future series, who knows?

Now to get back on line once more. First, let's use our new knowledge to check out Line 20 of our little program. If we assume a response of 'N' then 'Q$="N"' would be TRUE and 'Q$="n"' would have to be FALSE. Thus the complete logic expression becomes 'TRUE OR FALSE', that is 1 + 0, which evaluates to 1 (TRUE), and so the program would branch to Line 50. The converse, ie an entry of "n", produces 0 + 1 = 1, which would also cause a branch to Line 50.

How about Line 30? Suppose an entry of "Y" occurred. Why does the program not fall through to line 40 and produce a LISTing of the instructions? Let's examine it and find out! An entry of "Y" makes 'Q$<>"Y"' FALSE, and makes 'Q$<>"y"' TRUE, thus the complete expression equates to 'FALSE OR TRUE', or 0 + 1 = 1. As the whole logic expression is TRUE the instruction to GOTO 10 would be carried out. If "y" were entered we would have the logic equation 1 + 0 = 1, and if it were some other entry such as "X", we would have 1 + 1 = 1. Thus, any response (other than "N" or "n" in Line 20 which would bypass the problem line altogether) results in a truth-value of '1' and a return to Line 10. Problem now is, how should Line 30 have been written?

Let's take another look at the intent of Line 30, and I'd suggest that this would be a good policy to adopt wherever 'negatives' are involved, and I classify '<>' as a negative (as it involves the use of the word NOT). What was intended was that if Q$="Y" OR Q$="y" then Line 40 should be executed, otherwise back to Line 10, ie back to line 10 if NOT(Q$="Y" OR Q$="y"). In mathematical terms this means complementing the stuff inside the parens, to produce 'Q$<>"Y" AND Q$<>"y"' ('<>' being the complement of '=', and 'AND' the complement of 'OR'). Line 30 should therefore read :

```
30 IF Q$ <> "Y" AND Q$ <> "y" GOTO 10
```

Now an entry of "Y" will produce the Boolean algebra equivalent 0 * 1 = 0, and an entry of "y" the Boolean equivalent 1 * 0 = 0. Each expression therefore evaluates to FALSE and would cause a fall-through to Line 40, the desired response. On the other hand, an unacceptable entry of say 'X' would produce 1 * 1 = 1, evaluating to TRUE and causing a branch back to Line 10. Note that I have here used the symbol '*' to represent 'AND' simply to make my meaning clearer.

Summarising, the technique is to regard all ANDs as equivalent to '*' with an imaginary set of parens enclosing the terms on either side, and all ORs as equivalent to '+'. The truth-value of each term is then evaluated, from which the truth-value of an entire logic expression can be deduced. So:

Condition-A AND Condition-B OR Condition-C AND Condition-D

is first transformed to (A * B) + (C * D), or more correctly AB + CD. Now, by replacing A, B, C and D with any given set of truth-values composed of '1's or '0's, the truth-value of the complete expression can be very quickly arrived at. For instance if both Condition-A and Condition-B were TRUE and the others FALSE, we would have 1*1 + 0*0, reducing to 1 + 0, or 1. Thus the complete expression would evaluate as TRUE, and the program of which it forms part would behave accordingly.

In similar vein, where A, B, C etc represent conditional-expressions of the form 'Q$="Y"', 'A%<=B%', 'A$>B$' and so on, A AND B AND C OR D would be transformed into the equivalent Boolean expression ABC + D, or, for the purpose of calculating, as (A * B * C) + D. This expression is then evaluated by substituting '1' or '0' for A, B, C and D according to the truth-value of the individual conditional-expressions which they represent. Note that a single '0' appearing in a chain of ANDs causes the whole chain to evaluate to '0', so no time need be wasted on the remainder of the AND chain. Conversely, a single '1' in a chain of ORs causes the whole chain to evaluate to 1.

Thus if 'A' were equivalent to '0' in our example expression we would immediately equate 'ABC' to '0' and move on to 'D', as '0 * anything' evaluates to '0'. Or, if the expression were reversed to D + ABC, and 'D' were equivalent to '1', we wouldn't waste our time evaluating 'ABC', as '1 + anything' evaluates to '1'. Simple, isn't it?

Just be careful when complementing logic expressions, as I did earlier, to remember that the complement of '>' is '<=' and not simply '<'. That is, if 'A' is NOT GREATER THAN it is LESS THAN OR EQUAL TO 'B'. The complement of '<>' is '='. And don't forget those 'implied ANDs' when complementing from the algebraic form - they should be complemented to 'OR's *and the implied parentheses around a chain of them retained.*

I hope this little discussion has helped make it easier to figure out which way your program is going to respond to logic expressions, without having to try out all combinations in actual test RUNs of your program.

ANOTHER WAY TO SHORTEN & SPEED UP YOUR PROGRAMS

Some considerable time ago I suggested converting FP variables to Integers, SPLITting off instructions and also COMPILing programs as a means of opening up more memory for program space, in addition to speeding up program execution. Here is another technique for use in programs which have a lot of DATA lines which the program at some stage READs and assigns to variables and/or arrays. When this situation occurs we would end up with *two* copies of all this stuff residing in memory at one and the same time. One set in the original DATA portion of the program itself, and another in the Stack-Registers which keep track of their ongoing values. Quite apart from the fact that, when popping in and out of subroutines or executing a GOTO, the program wastes time scanning through all these lines in order to locate its destination-line.

In such cases, just as we did with Game-Instructions, we would set about SPLITting off these DATA lines, removing the line-numbers and the actual word DATA, and renaming the resultant file to "GAME.DAT".

Now, somewhere near the beginning of our program we would add a line to the effect :

```
50 OPEN OLD "GAME" AS 1
```

Of course, if Channel 1 is already pre-empted for some other file, we would use the next available Channel-Number instead. Note that the extension '.DAT' is not necessary as this is XBASIC's default extension when opening up a Channel.

Now we would scan our program for all READ statements referencing DATA in our file, and replace the word 'READ' with 'INPUT #1," - thus :

```
1000 INPUT #1, A%(1),A%(2),A%(3),X$
```

But be careful to SPLIT off only those DATA items which are actually copied into an array, or assigned as initial values to selected variables, and be warned that you cannot RESTORE to a specific DATA item in the file. Though you can RESTORE to the beginning of the file by CLOSing Channel 1 and then re-OPENing it. Also don't forget to CLOSE the file once you've read in all the available DATA items, so :

```
2000 CLOSE 1
```

Where a scratch variable is used repeatedly, such as :

```
100 READ M$: PRINT M$: PRINT: READ M$: PRINT M$
```

it would not be feasible to SPLIT off the corresponding DATA messages, as they are brought into memory just one at a time for assignment to M$, so these occurrences should be left untouched. The example Line 100 would, of course, read into M$ the DATA item currently pointed to by the internal DATA-pointer, PRINT it, then read the next DATA item into M$ before PRINTing it too. For a fuller explanation of how to use these DATA files to your best advantage you can do no better than to read the appropriate section in the XBASIC article.

To be Continued Next Month

Corrections: June issue - page 42

```
42  41  26  08  A6  0E  81  43  10  27  D9  SF
B   A                       C

page 42

line 40 CHR$  >  CHR$
also all references to CHRS$  >  CHR$
```

```
page 43

center of page:
20 IF Z <= 9 THEN X = X * 11 ELSE X = X * 15

PAGE 44

31  > 3  ALSO  31 32 33 SHOULD BE EMPHASISED AS 1  2  3

PAGE 46 mimi-field > mine-field
```

A PL/9 interface for ISAM

by Martin C. Gregorie
10 Sadlers Mead
Harlow, Essex U.K.

As soon as Joe Condon's ISAM package appeared in the Micro Journal I had to get a copy. A club mailing list I look after needed a rewrite and I wanted to use some form of keyed access instead of the serial file dumps of an in-memory table I'd been using. Well the package duly arrived and was installed and checked out. No problems showed up except that it is really designed for use with BASIC, which I never use unless forced to. In any case the mailing list was written in PL/9 and I wanted to reuse as much of it as possible. Only one answer - write an interface for PL/9 - the library presented here.

The library is designed to provide a set of mainframe database-like file handling 'verbs'. I have given it the flavour of a CODASYL database such as IDMSX because that is what I use on mainframes. The aim was to provide a set of record-level procedures whose names would not clash with file handling procedure names within my normal libraries, hence READY/FINISH in place of OPEN/CLOSE. I also wanted to create a set of procedures that would require only the minimum number of parameters for each, and would allow several ISAM files to be open at once. This required the introduction of an 'ISAM Control Block' to retain the values of other parameters as well as the current error code for the file. The ICB contains a structure equivalent to the BASIC string array except that the record buffer is not included in the ICB, though its length and a pointer to it is. I could have produced a much simpler interface consisting of just one procedure but, although it would have been much smaller, it would have also been harder to use.

Following sections of this article contain full documentation for ISAM.LIB, the interface library, the library code itself, an exact PL/9 equivalent of Joe Condon's original demo program (LABEL.PL9), and an interface test program written in assembler (ISAMSIM.TXT). ISAMSIM is called just like ISAM, but all it does is to display the calling parameters and return error code 123. You may want to use it for initial program checkout.

ISAM is used from PL/9 just as normal. Install ISAM and then run programs that call it. You can also compile and test programs under the debugger with ISAM installed but beware that this will corrupt the last 4 bytes of the program being tested due to the necessity of planting link addresses at [MEMEND]-4. This has not yet crashed my system but does cause strange effects in the last statement of the test program. The actual call to ISAM is in the ASMPROC ISAM. This must be an ASMPROC as ISAM corrupts the Y-register which is also used by PL/9 as a global variables pointer. The actual call must preserve the Y-register - something which cannot be done within PL/9. But for this no GEN statements would be necessary.

A tip for ISAM users. I have heard statements to the effect that ISAM is slow - I dont find it any slower than I would expect a keyed file access system to be on a FLEX disk, but one sure way to DESTROY its performance is to let an ISAM file get fragmented. The easy way to fragment the file is to create it on an old, fragmented disk. The solution is to create ISAM files on newly formatted disks, and always back up disks by using the sequence:

a format a new disk
b copy the old disk onto the new one
c file the old disk as your backup
d use the new disk as the working copy

This procedure will have the advantage of speeding up access to all files on the disk by ensuring that all files are unfragmented.

IDMSX is a trademark of International Computers Limited

```
PL/9 ISAM interface Definition
---------------------------------

Introduction
------------
This document describes the procedures in the ISAM.LIB procedure
library.  The library has no dependencies on other procedure
libraries. It contains a set of procedures that provide an
interface to ISAM.CMD with a flavour resembling that of CODASYL
databases.

Data space.
-----------
An ISAM Control Block (icb) is needed for each ISAM file to be
accessed by the program and a record buffer to contain records
read from or written to the file.  The record may consist of any
combination of bytes, strings, integers, and reals.  The
following example will be used in the following examples:

          procedure test:
             byte icb(50),          /* the icb */
                  name(30),office(30),on_leave:
                  integer age;

This declares a control block and a record of 63 bytes length
consisting of:

          name       a string of up to 29 characters
                     (the key field)
          office     a data string of up to 29 characters
          on_leave   a single byte flag
          age        an integer.

Using the interface.
--------------------
The ISAM package must be loaded before the calling program is
run. Alternatively the ISAMSIM program may be loaded in its
place - this program displays ISAM call parameters as a
debugging aid, allowing initial testing to be carried out
without risk of damaging the disk by using ISAM.CMD.

The file is created by the call:

          format(.icb,1,"testfile.isa",1000,63,30)

to create TESTFILE.ISA with 1000 records, each 63 bytes long
with a 30 byte key.

The file can have data loaded by the sequence:

          ready(.icb,1,"testfile.isa",63,30);
          if dbstatus(.icb) then
             begin
                ...... /* set up the record */
                store(.icb,.name);
                if dbstatus(.icb) then /* error action */
             end;
          .....
          finish(.icb);
```

An entire file may be read with the sequence:

```
        ready(.icb,1,"testfile.isa",63,30);
        if dbstatus(.icb)=0 then
           begin
              obtain(.icb,.first,.name);
              while dbstatus(.icb)=0
                 begin
                    /* display the record */
                    obtain(.icb,.next,.name);
                 end;
           end;
        finish(.icb);
```

You may position to any point in the file:

```
        position(.icb,.first);        /* start of file                 */
        position(.icb,"Martin");      /* before record with specified key */
        position(.icb,.last);         /* end of file                   */
                                      /* position may return 201 if any */
                                      /* other value is supplied for the */
                                      /* second parameter, e.g. .next   */
```

You may read any record in the file:

```
        obtain(.icb,.first,.name);
        obtain(.icb,"Martin",.name);    /* this returns -100 if the record */
                                        /* found does not match the key    */
                                        /* but has a higher key instead,   */
                                        /* it is still retrieved though    */
                                        /* if the reply is >0 it is an     */
                                        /* ISAM error                      */
        obtain(.icb,.last,.name);
        obtain(.icb,.next,.name);       /* reads next record in file       */
        obtain(.icb,.current,.name);    /* rereads the current record      */
        obtain(.icb,.previous,.name);   /* reads previous record in file   */
```

Records can be added:

```
        store(.icb,.name);
```

The last record accessed in the file may be modified:

```
        modify(.icb,.name);
```

Or it may be deleted:

```
        erase(.icb);
```

Modify is in fact a convenience function that calls store after erase.
Remember that the reply code should be checked after every ISAM operation. There are two possible methods possible as all ISAM procedures return reply values. They are:

```
        1          obtain(.icb,.next,.name);
                   if dbstatus(.icb) then
                        /* error action */

        2          if obtain(.icb,.next,.name)>0 then
                        /* error action */
```

Dbstatus may be called several times - the reply value is kept til the next ISAM action is called. If errors do occur there are two procedures to help fix the problem; they could be used as follows:

```
        if dbstatus(.icb)>0 then
           begin
              finish(.icb);              /* attempt to close the file */
              if dbstatus(.icb) then     /* fix failed */
                 begin
                    abort(.icb);         /* force file closed */
                    reorganise(.icb);    /* forceably fix the file */
                 end;
           end;
```

This is just an example, not a recommendation!

Procedure definitions

Procedures map onto ISAM verbs as follows:

procedure	ISAM verb	comments
ready	OPEN	opens an ISAM file
finish	CLOSE	closes an ISAM file
abort	INITIALISE	forced ISAM file close
position		sets file position without reading any records
	FIRST	(.cmd=.first)
	LAST	(.cmd=.last)
	START	(.cmd=.key value)
obtain		reads the nominated record
	FIRST + NEXT	(when .cmd=.first)
	LAST + PREVIOUS	(when .cmd=.last)
	START + NEXT	(.cmd=.key value)
	NEXT	(.cmd=.next
	CURRENT	(.cmd=.current)
	PREVIOUS	(.cmd=.previous)
store	ADD	stores the record in the file
erase	DELETE	erases the current record
modify	DELETE + ADD	replaces the current record
reorganise	REORGANISE	file recovery after ABORT
format	CREATE	new ISAM file setup
dbstatus	-	returns the current reply value for the file

All procedures return a reply value.

Parameters are:

Parameter	Description
.cmd	file operation for position and obtain: one of: .first .last .next .current .previous or .key where the byte area 'key' holds the record key to be found
.icb	ISAM file control block reference
id	ISAM file number
klen	key length (first klen chars in the file)
.name	string containing the file name (standard file spec, e.g. MYFILE.ISA)
.rec	reference to the first field in the record buffer
rcnt	number of records in the file
rlen	record length

Procedures return standard ISAM reply codes plus:

Value	Meaning
-100	the record returned by OBTAIN doesnt match the given key
201	attempt to use POSITION in such a way that a record would be read; i.e. not for ISAM verbs FIRST, LAST, or START.

The procedures are:

```
        abort(integer .icb);
             Forcible close the file (may leave corrupt
             flag set)

        dbstatus(integer .icb);
             Return last reply code from a file

        erase(integer .icb);
             Delete the current record

        finish(integer .icb);
             Close the file

        format(integer .icb, id; byte .name;
               integer rcnt,rlen,klen);
             Create a new file as id (named .name containing
             rcnt records each rlen bytes long, keyed on the
             first klen bytes)

        modify(integer .icb; byte .rec);
             Change the contents of the current record, moving
             it in the file as necessary

        obtain(integer .icb; byte .cmd,.rec);
             Retrieve a record from the file. The record is
             selected by the value of .cmd.

        position(integer .icb; byte .cmd);
             Position the record pointer within the file. The
             position is set by the value of .cmd

        ready(integer .icb, id; byte .name; integer rlen,klen);
             Open the file named in .name as ISAM identier id.
             The record length is supplied in rlen and the key
             length in klen.

        reorganise(integer .icb);
             Reorganise the file.

        store(integer .icb; byte .rec);
             Store a new record in the file.
```

PL/9 ISAM interface control block definition.

The ISAM Control Block is declared as a 50 byte area:

```
        byte icb(50);
```

It must remain in scope throughout the portion of program using the file; this means that it should be declares as Global data or in the outermost procedure to use ISAM file handling.

The block is regarded internally as an array of 25 integers:

Location	Contents
+0	Pointer to +13 (ISAM file number)
+1	2
+2	Pointer to ISAM command string
+3	command length (2)
+4	Pointer to +14 (ISAM file name)
+5	Length of file name (up to 14
+6	Pointer to +24 (Record size)
+7	2
+8	Pointer to +23 (ISAM file size in records)
+9	2

To be Continued Next Month

WINDRUSH
Micro Systems Ltd.

Worstead Laboratories (Reg. Office)
North Walsham, Norfolk NR28 9SA
Tel: (0692) 404086
Telex: 975548 WMICRO G
MAPCON approved consultants

DUAL PROCESSOR MDS RUNS FOUR OPERATING SYSTEMS

Windrush Micro Systems Limited are pleased to announce the immediate availability of their Model 8000/009, 8000/010 and 8000/020 MDS systems which can also serve as OEM target hardware.

The MDS system is based on a standard 6U, 19" rack mounting system fitted into a desktop enclosure. In this configuration unrestricted access to all system modules is via the rear of the case. The computer system is also available in a 19" rack mounting version with full access from the front for OEM applications.

The user can choose between 6809, 68000, 68010 and 68020 processor cards and between the FLEX, OS-9/Level I, OS-9/Level II, and the OS-9/68K 'Professional' operating systems. In all systems the rack, disk storage and I/O hardware are identical. This enables the MDS user to configure the system by simply changing processor cards. The 6809 processor card supports the FLEX, OS-9/Level I and OS-9/Level II operating systems. The 68000, 68010 and 68020 processor cards support the OS-9/68K operating system. The 68020 processor card includes the MC68881 floating point co-processor which is fully supported by OS-9/68K.

System features include a choice of processors, 100% fully static zero wait-state static RAM, battery backed clock calendar, four RS-232C serial ports, two centronics parallel printer ports, a 25 Mb (unformatted) Winchester and a choice of 5.25" or 3.5" 1 Mb (unformatted) floppy disk drive. The modular nature of the system enables it to be customised to individual requirements at very low cost.

Typical system costs are as follows:

Model 8000/009/F	68B09 processor, 64K memory, FLEX O/S	£2895.00
Model 8000/009/02	68B09 processor, 1 Mb memory, OS-9/Level II O/S	£4250.00
Model 8000/010	12.5 Mhz 68010 processor, 256 Kb memory, OS-9/68K	£3495.00
Model 8000/010/X	12.5 Mhz 68010 processor, 1280 Kb memory, OS-9/68K	£4064.00
Model 8000/020	16.67 Mhz 68020/68881, 512 Kb memory, OS-9/68K	£4095.00
Model 8000/020/X	16.67 Mhz 68020/68881, 1536 Mb memory, OS-9/68K	£4667.00

MOTOROLA INC.

MOS Memory Products Group
P.O. Box 6000
Austin, Texas 78762

For further information contact

EDITORIAL CONTACT:
Mark Vercruysse
512/928-6804

READER CONTACT:
Memory Marketing
512/928-6700

INQUIRY RESPONSE:
R.Q. Green
P.O. Box 52073
Phoenix, AZ 85072

MOTOROLA INTRODUCES 4K X 4 STATIC RANDOM ACCESS MEMORY,
THE MCM6269

Austin, Texas, May 11, 1987... Motorola's MOS Memory Products Division introduces a new 4K x 4 Static RAM, the MCM6269. Fabricated using Motorola's second-generation high-performance silicon-gate CMOS (HCMOS III) technology, this fully static design eliminates the need for external clocks or timing strobes, while CMOS circuitry reduces power consumption and provides better reliability. With a fast access time of 35ns, this device is suitable for cache and other sub-50ns applications.

Similar in design to the Motorola MCM6268, the MCM6269 is a fast chip select version featuring access to data in 15ns Maximum.

The MCM6269 is organized 4K x 4 and operates under a single 5V supply. The device has low Active AC power operation of 90mA Maximum and 40mA Maximum under DC conditions.

The 35ns MCM6269 conforms to the JEDEC standard pinout. It is available in a 20 lead plastic dual-in-line package and is priced at $5.78 in 100 piece quantities. Order this device by specifying MCM6269P35.

For more information contact your local Motorola Sales Office or authorized Motorola Distributor.

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

UNATTENDED COMPUTER PROTECTION

Electronic Specialists announces expansion of their Unattended System Power Interrupter Line. Designed to protect Unattended computer and other systems from violent power fluctuations or outages, the Interrupters disconnect AC power when irregularities occur.

Automatic and Manual Reset models are now available. In addition, straight blade and Twist configurations for 15, 20 and 30 Amp circuits are offered.

Electronic Specialists, Inc. P.O. Box 389, 171 S. Main St.
Natick, Massachusetts 01760 Phone: 800-225-4876

NEW PRODUCT ANNOUNCEMENT

The PADC-DAC-8 µLAB™

URDA is pleased to announce the availability of the PADC-DAC-8 µLAB™, an Analog to Digital and Digital to Analog Converter expansion board for the P68000µLAB™, or for any device with two 8 bit parallel ports. As a Notebook Computer™ expansion, the A/D-D/A converter comes with a 3 ring mounting to fit conveniently into the P68000 µLAB™.

The µLAB™ series is centered around popular microprocessors, e.g., 68000, 8086, etc., and support chips with operating system software. This combination facilitates programming the microprocessor in its native (machine) language while providing easy access to the development system hardware so that its signals can be interfaced to the rest of the world and the microprocessor system can be used to observe and control parts of the real world.

As an example, the P68000 µLAB™ is a 68000 microprocessor with keypad, LED display, 4 K bytes SRAM, 8 K bytes of EPROM, cassette interface, software operating system, power supply, instruction manual and Programmer's Reference Manual completely packaged in a 2 inch 3 ring binder, i.e., a Notebook Computer™. With an educational and student price of $197.50, the cost barrier to providing individualized instruction on a microprocessor development system is broken.

Much of the world operates with continuous analog signals. To simplify laboratory experiments and computer control, an optional accessory, the PADC-DAC-8 µLAB™, provides 8 channels of analog to digital conversion and 8 channels of digital to analog conversion both with 8 bits per channel. The PADC-DAC-8 µLAB™, complete with power supply, instruction manual and notebook also sells for $197.50.

The PADC-DAC-8 µLAB™, provides a convenient interface between the microprocessor and real world signals for experimentation with the control of a variety of electronic, electro-mechanical, etc., devices and systems. The clock is free running to give the latest available sample thus facilitating multiple sampling frequencies across eight channels without complicated timing schemes. The maximum sampling rate is a function of the number of channels in use with a maximum of 10 K Hertz.

The control word provides for selection of: (1) input/output - 1 bit, (2) channel selection - 3 bits for A/D and 3 bits for D/A, and (3) a single bit for simultaneous D/A conversion.

The double buffered D/A converters use a single convert signal to make it possible to precisely output signals for sampled data control.

The low cost, $197.50, makes it possible to use the PADC-DCA-8 µLAB™, with the P68000 µLAB™, as an educational tool for student projects and one of a kind testing and experimentation.

The PADC-DAC-8 µLAB™ features include:

- 8 channels Analog/Digital Conversion - 8 bits
- 8 channels Digital/Analog Conversion - 8 bits
- Conversion rate is 10 K Hertz for single channel operation. Additional channels will derate according to user software
- Voltage input compliance +,- 12 volts
- Voltage output compliance +,- 12 volts
- All output channels double buffered to allow simultaneous control outputs
- P68000 µLAB™, interface
- Parallel Input/Output Interface
- Individually addressable channels

PADC-DAC-8 µLAB™, Components

- Printed Circuit Board
- 3 Ring Notebook Binder
- User Manual
- 115 VAC 60 Hz Power Adapter

The PADC-DAC-8 µLAB™ lists for $295.00 with a discount price of $197.50 for Educational Institutions, Faculty and Students. To order, contact

Quasitronics, Inc. 1-800-245-4192, in PA 412-745-2663

For technical information, contact URDA, Inc. 412-683-8732

MOTOROLA

Microcomputer Division

2900 South Diablo Way, Tempe, AZ 85282
P.O. Box 20912, Phoenix, AZ 85036

EDITORIAL CONTACT:
Andreas Schreyer
(602) 438-3500

READER CONTACT:
Motorola Marketing
Toll Free Hotline
(800) 521-6274

MOTOROLA ANNOUNCES A NEW VME DELTA SERIES™ SYSTEM, MODEL 2316 TO CHALLENGE DEC™ DESKTOP IN THE OEM TECHNICAL COMPUTER SYSTEMS MARKET

Tempe, AZ, May 1, 1987 . . . Motorola Microcomputer Division announced its newest VME Delta Series system, the Model 2316. The second Delta Series system to be announced, following the Model 2616 earlier this year, addresses the work group segment of the technical computer market.

"The 2316 performs at almost two times the speed of a VAX 11/780, and is being introduced at a price comparable to the new DEC MicroVAX™ 2000 desktop computer," said Andreas Schreyer, Motorola's systems planning manager. "It is clearly faster, more configurable, more flexible and definitely a better choice for OEMs and Systems Integrators. It also is the only system in its class that supports Unix™ release 3. The Unix operating system is the operating system of choice among most technical systems integrators. Stand-alone or combined with other VME Delta Series systems, the 2316 is going to give technical system vendors a run for the money. Motorola is determined to capture a major share of the technical systems market and the VME Delta Series Model 2316 is just the product to do it."

VMEbus, MC68020 32-bit performance and Unix Release 3 are all strong features of the 2316. The system has a six-slot VMEbus chassis which is configurable by the OEM, offering unlimited application possibilities. With up to four Mbytes of memory and up to 160 Mbytes of formatted ESDI Winchester disk, the 2316 can effectively support up to 8 users or devices, making it a powerful system for the work group class of technical computer systems. This provides 32-bit performance at a cost of less than $2,000 per served connection.

Also, since the VME Delta Series uses the Motorola 68020 microprocessor and SYSTEM V/68™ Release 3 operating system, remote peripheral and file sharing is easily accommodated by the Model 2316, via local area network interconnections. And, as with all VME Delta Series systems, the Model 2316 comes with many communications packages including Ethernet protocols and technical system-oriented local area networks. Numerous mass storage devices, local and remote diagnostic software packages, languages, development tools and numerous third-party software packages are also available.

Editor
68 MICRO JOURNAL
Computer Publishing Center
5900 Cassandra Smith Road
P.O. Box 849
Hixson, TN 37343

Code 4631
Naval Research Laboratory
Washington, D.C. 20375

Dear Sir,

I recently sent this letter to Dr. E.M. Pass who writes the C column for your journal. However, he had little familiarity with the Macintosh and, therefore, I thought I should try the Macintosh section of 68' Micro Journal.

My questions center around the use of the Macintosh Plus as a small number cruncher doing scientific calculations on my desktop (eg. numerical solutions to low dimensional ordinary differential equations systems), but I think many of them would apply to C packages on 68000 computers in general.

I have tried using Fortran (Microsoft-Absoft), but I have found that it is clunky and buggy. Program development is painful, compared to say Fortran on the VAX. Also, in general, I am not fond of Fortran. I have recently begun to program in C (using Lightspeed C, a very nice package) and, although I'm still a novice, I have found C to be more "natural", even for many scientific computing applications. It's a great language with a lot of potential. But there are some problems.

On the Mac all C packages apparently use the SANE (Standard Apple Numeric Environment) for their floating point operations. These are very accurate (80 bit, I think), but they are slow. This is a curse for scientific computing where floating point operations are the name of the game. Microsoft Fortran, on the other hand, for all its other faults, has it's own floating point routines built in. This makes Fortran programs with lots of floating point operations run 2 to 4 (maybe more) times faster than their C counterparts (depending on the precision one is using).

Another, albeit smaller, problem is that C lacks some of the niceties of Fortran which us scientists have come to like: array bounds checking, overflow alerts, passing variable dimensioned arrays to subprograms, etc. Work arounds can be found at times, but they detract from what is essentially a nice programming language and environment.

My questions are these. I have heard it rumored (on a local BBS) that there exits a Motorola set of floating point routines which are fast. Do they or any others exist? Can C packages like mine make use of them? If so, how hard is it? Who would I contact? I recently saw a book titled C++ about an (AT&T) extension to C which appeared to support many of the things I like about Fortran –things like, vectors, matrices, type checking and apparently lots more. Does anything like this exist for the Macintosh or is anything in the works for it? Are there C libraries of scientific routines availalble for us small time programmers? Is there a version of "lint" (type checking) for micro's, the Macintosh, in particular?

Thanks for taking the time to read this. I appreciate any answers to any of these questions. Many of the scientists I know have Macintoshes and would be very interested in the answers also. Some may be able to afford the MacII when it becomes available (which will solve much of the speed problem), but many of us will have our Mac Pluses for some time to come. Even so, we feel that that little 68000 could do a lot of scientific calculating for us in the next few years, if only we could work around the above problems.

Sincerely yours,

[illegible]

Louis M. Pecora

Editor's Note: Louis, thanks for the letter. Your observations are essentially correct, as concerns the Macintosh SANE routines.
I think your best bet is to look into the Macintosh II, or one of the 68020/68881 upgrades. The Mac II should be available sometime this summer. It has the math crunching 68881 and is driven by a 68020. On modified Macs to that configuration the speed up has been dramatic.
My personal opinion is that any HLL will be slower than assembler (even with a 68020/68881 combo) so the use of C, or any other HLL is going to be degraded, in speed, to some degree or another. Even Fortran. To that end, assembler is still "King".
The "Lint" program is available from several C vendor for the Mac. Usually it is named something different (maybe for copyright reasons), you will have to check with vendors on that accouut.
We use the Apple MPW set of HLLs. This includes a very good C (no Lint), straight Pascal and Object Pascal as well as all the other programming aids available to the Apple developer. This total package is about $600.00. This price covers about everything available from Apple. If interested, contact the Apple Developers Group - A.P.D.A. - 206 251-6548.
Maybe some of our readers will have more info for you. Good Luck.

DMW

MARION E. WOLFE, COMPUTER CONSULTANT
1828 BEN LOMOND DR.
GLENDALE, CALIFORNIA 91202
(818) 241-0813
June 12, 1984

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

Dr. E. M. Pass's article "C User Notes" in the June 87 issue is very interesting, but there is an error on page 10. He says that binary fractions cannot always be represented exactly as a decimal fraction. This is not correct. There is a clue in the fact that 2, the only factor of the base of binary representation, is also a factor of the base of the decimal system. All binary fractions can be represented exactly as a decimal fraction using the same number of digits to the right of the point.

Binary 0.1 is 1/2 or decimal 0.5, binary 0.01 is decimal 0.25, and so on.

Because 5, a factor of the base of the decimal system, is not a factor of binary base, most decimal fractions cannot be represented exactly in binary.

I have a problem with which I hope someone can help me:

I purchased a CoCo II, hoping to use it as a terminal on my Smoke Signal Chieftain, running OS9, level 2. Now I find that the serial port on the CoCo is set at 600 bps. The SSB serial port has many speeds but not 600. There are 300 and 1200. I use 9600 with my other terminal and printer. I want to retain that speed. How can I get either 600 on the SSB or another speed on the CoCo?

Thank you.

Sincerely

[illegible]

Marion E. Wolfe

SCHOOL OF ENGINEERING
UNIVERSITY OF PITTSBURGH
PITTSBURGH, PENNSYLVANIA 15261

MAILING ADDRESS:
WILLIAM G. VOGT OR MARLIN H. MICKLE
MODELING AND SIMULATION CONFERENCE
348 BENEDUM ENGINEERING HALL
UNIVERSITY OF PITTSBURGH
PITTSBURGH, PENNSYLVANIA 15261

MODELING AND SIMULATION CONFERENCE

Gentlemen:

The University of Pittsburgh Conference on Modeling and Simulation is pleased to announce that we wil be sponsoring two short courses:

COURSE I - PROGRAMMING AND INTERFACING THE 68000 MICROPROCESSOR

COURSE 2 - PROGRAMMING AND INTERFACING THE 8086 MICROPROCESSOR

ANNOUNCING TWO SHORT COURSES

COURSE 1 - PROGRAMMING AND INTERFACING THE 68000 MICROPROCESSOR

AND

COURSE 2 - PROGRAMMING AND INTERFACING THE 8086 MICROPROCESSOR

DESCRIPTION OF BOTH COURSES

Learn how to program either the Motorola 68000 or the Intel 8086 microprocessor and interface it to the real world at 2 1/2 day continuing education courses at the University of Pittsburgh. Course schedules are given below. These short courses are sponsored by the Modeling and Simulation Conference.

The topics covered in each short course include 68000 (or 8086) architecture, addressing, instruction set, programming, peripherals, support, communication, interfacing, closed loop computer control, etc.

Each short course includes 5 hours of lecture and 12 hours of hands-on laboratory experience. Each student will individually perform experiments and get programming practice with his/her own individual P68000 µLAB™ (or P8086 µLAB™) microprocessor development system and his/her own individual PWWEB µLAB™ wire-wrap expansion kit.

BEST OF ALL -- the P68000 µLAB™ (or P8086 µLAB™) microprocessor development system and the PWWEB µLAB™ (total list price $360) on which you learn is YOURS TO KEEP and take home with you!!!

INTENDED AUDIENCE

Each short course is suitable for faculty members in computer science, computer engineering and electrical engineering departments, practicing engineers, scientists, technicians, high school students, college students and anyone else who would like to learn about the 68000 (or 8086) processor and interfacing it to the real world.

INSTRUCTOR

The instructor for the short course is Dr. Marlin H. Mickle, Professor of Electrical Engineering, University of Pittsburgh. Several graduate assistants will also be available to assist in the laboratory sessions.

SCHEDULE OF ACTIVITIES - BOTH COURSES

FIRST DAY

8:30 Coffee and Donuts

9:00 Lecture #1: Overview, Basic Architecture, Number and Information Representation, Simple Programming Instructions

10:00 LABORATORY #1: Visual and Audio Response with the P68000 µLAB™ (or P8086 µLAB™)

12:00 LUNCHEON BUFFET (No Charge) -- Optional continuing practice

1:00 LECTURE #2: Addressing, Instructions, and 68000 (or 8086) Programming in Assembly Language and Native Machine Code

2:00 LABORATORY #2: Reinforcement of Lecture #1 and #2 Concepts

4:30 QUESTION AND ANSWER PERIOD (Optional)

6:00 DINNER (Individual Initiative)

7:00-9:00 SELF STUDY

SECOND DAY

8:30 Coffee and Donuts

9:00 Lecture #3: Interfacing to the 68000 (or 8086) microprocessor and support chips.

10:00 LABORATORY #3: Experiments involving the turning on of lights, running motors, and reading switch inputs.

12:00 LUNCHEON BUFFET (No Charge) -- Optional continuing practice

1:00 LECTURE #4: Data communication and external control of the microprocessor (Interrupts)

2:00 LABORATORY #4: Serial Communication from one 68000 (or 8086) to another 68000 (or 8086) using the RS232C interface.

4:30 QUESTION AND ANSWER PERIOD (Optional)

6:00 DINNER (Individual Initiative)

7:00-9:00 SELF STUDY

THIRD DAY

8:30 Coffee and Donuts

9:00 Lecture #5: Multiple processors, coprocessors and process control.

10:00 Demonstrations using support equipment for the P68000 µLAB™ (or P8086 µLAB™), Assemblers, Closed Loop Control, Signal Processing

12:00 Course Completed

REGISTRATION INFORMATION

Dates: As Listed in the Table Below

Go/No-Go Date: As Listed in the Table Below

PLACE: BENEDUM ENGINEERING HALL, University of Pittsburgh, Pittsburgh, PA 15261

REGISTRATION FEE: $600 (includes $360 in equipment you take home with you)

DEPOSIT: $200 (Nonrefundable unless course does not go)

The Registration Fee of $600 includes all notes, a P68000 (or P8086) µLAB™ microprocessor development system and a PWWEB µLAB™ wire-wrap expansion kit including cables, connector- chips and tools (total list price of $360), attendance at all lectures, laboratory sessions, discussions and demonstrations and two luncheon buffets. Registration will be limited to the capacity of our lecture hall. Therefore, your deposit of $200 is nonrefundable unless a No-Go decision is made in which case your deposit will be returned. We must receive your registration and your deposit for the course by the Go/No-Go date listed below. The balance of $400 will be due on the first day of the course. Please bring a check or cash

TO REGISTER FOR THE SHORT COURSE, SEND YOUR NAME, MAILING ADDRESS AND A CHECK (MADE OUT TO UNIVERSITY OF PITTSBURGH) FOR A $200 DEPOSIT TO:

Marlin H. Mickle or William G. Vogt, Modeling and Simulation Conference, 348 Benedum, Engineering Hall, University of Pittsburgh, Pittsburgh, PA 15261, Tel Nos. 412-624-9682 or 412-624-9686

SCHEDULE OF COURSE OFFERINGS

COURSE NO. 1 - PROGRAMMING AND INTERFACING THE 68000 MICROPROCESSOR

No.	Dates	Go/No-Go Date
1.1	September 12, 19, 26, 1987 (Saturdays)	August 21, 1987
1.2	January 21-23, 1988	January 1, 1988
1.3	May 2, 3, 4, 1988	April 15,1988

COURSE NO. 2 - PROGRAMMING AND INTERFACING THE 8086 MICROPROCESSOR

No.	Dates	Go/No-Go Date
2.1	October 22,23,24, 1987	October 1, 1987
2.2	February 13,20,27, 1988 (Saturdays)	January 29, 1988
2.3	May 2, 3, 4, 1988	April 15,1988

68 Micro Journal
CPI
PO BOX 849
Hixson Tn 37343

I was premature in furnishing you my Shell sort. I have checked out looping the 'lookback', and find that the sort time (68000, 8Mhz) for 20,000 items drops from 12 to 5.5 seconds, a worthwhile gain since it costs nothing in code size. Because all items reach their spot in one pass there is no swapflag, no repeats.

Is it still a Shell sort? I don't know, but it works better.

While I was at it I did a Quicksort. It is not a full implementation because of the stack overflow problems that come with unrestricted recursive calls (each stack frame requires 24 bytes). Just a driver for the Shell sort. The timing fluctuates, probably because of swings in partition sizes, sorting 20,000 items in about 2.3 seconds. It may not be worth the potential problems for such a small gain.

Jim Wilson
1815 N Harvard A-212
Los Angeles, CA 90027
213-465-2391

```
                      *
                      * shsrt.s  a modified Shell sort
                      *
                      * VOID _shsrt(&list,count)
                      *       long  &list,count
                      *
                      * Features an extended look back
                      * and wobble; there are no repeats,
                      * no swapflag.
                      *
                      * Uses d0/a0-a1 as scratch registers
                      *
                      * define frame pointer/offsets
                      fp              equ     a6
                      count           equ     12
                      list            equ     8
                      * register equates
                      bak1            equ     a0
                      bak2            equ     a1
                      bas_lst         equ     a3
                      indx            equ     d2
                      counter         equ     d3
                      intvl           equ     d4
                      source          equ     a4
                      store           equ     d0
                      target          equ     a5
                      *
                      _shsrt
                      *
00000000 4E560000     link     fp,#0
00000004 48E7383C     movem.l  d2-d4/a2-a5,-(sp)
                      * load caller's parameters
00000008 282E000C     move.l   count(fp),intvl
0000000C 266E0008     move.l   list(fp),bas_lst
                      *
                      * repeat while interval > 0
                      *
                      nxt_int:
00000010 E28C         lsr.l    intvl              / 2
00000012 673E         beq      exit               finished
00000014 7001         moveq    #1,d0              avoid..
00000016 B084         cmp.l    intvl,d0           endless loop
00000018 6702         beq      no_wob
0000001A D880         add.l    d0,intvl           the wobble
                      no_wob:
                      * calculate index
0000001C 2404         move.l   intvl,indx         copy and..
0000001E E58A         lsl.l    #2,indx            upscale
                      * set counter for this pass
00000020 262E000C     move.l   count(fp),counter
00000024 9684         sub.l    intvl,counter
                      * initialize pointers
00000026 2A4B         move.l   bas_lst,target
00000028 49F52800     lea      (target,indx.l),source
0000002C              page
                      * compare/swap until count expires
                      *
                      dec_cnt:
0000002C 5383         subq.l   #1,counter         adjust counter
0000002E 6BE0         bmi      nxt_int
00000030 BB8C         cmpm.l   (source)+,(target)+
00000032 6FF8         ble      dec_cnt
                      * save the source
00000034 2024         move.l   -(source),store
                      * load lookback pointers
00000036 41EDFFFC     lea      -4(target),bak1
0000003A 2249         move.l   bak1,bak2
                      * rotate (target) forward
0000003C 2800         move.l   (bak1),(source)+
                      *
                      * boost (source) as high as it will go
                      *
                      boost:
0000003E 93C2         sub.l    indx,bak2          update bak2
                      * 2 reasons not to rotate (bak2)
00000040 B3CB         cmp.l    bas_lst,bak2
00000042 650A         blo      restart            out of bounds
00000044 B091         cmp.l    (bak2),store
00000046 6C06         bge      restart            wrong size
00000048 2091         move.l   (bak2),(bak1)      rotate..
0000004A 2049         move.l   bak2,bak1          update bak1
0000004C 60F0         bra      boost              repeat
                      restart:
0000004E 2080         move.l   store,(bak1)
00000050 600A         bra      dec_cnt
                      exit:
00000052 4CDF3C1C     movem.l  (sp)+,d2-d4/a2-a5
00000056 4E5E         unlk     fp
00000058 4E75         rts
                      *
0000005A              end
```

```
                      *
                      * qksrt.s
                      *
                      * a recursive partition sort
                      *
                      * If 'list' is from 2 to 127 items,
                      * Shell sort it; if not, partition
                      * it and do recursive calls.
                      *
                      * takes pointer and count from stack;
                      *
                      * VOID _qksrt(*base,count)
                      *       long  *base,count
                      *
                      * d0/d1, a0/a1 scratch registers
                      *
                      * define frame pointer/offsets
                      fp       equ     a6
                      base     equ     8
                      count    equ     12
                      *
                      balance  equ     d0
                      head     equ     a3
                      tail     equ     a4
                      *
                      _qksrt
                      *
00000000 4E560000     link     fp,#0
                      * save working registers
00000004 48E70018     movem.l  a3-a4,-(sp)
                      * load caller's parameters
00000008 266E0008     move.l   base(fp),head
0000000C 222E000C     move.l   count(fp),d1
                      * how long is list?
00000010 7002         moveq    #2,d0
00000012 B280         cmp.l    d0,d1              < 2?
00000014 6568         blo      exit               nothing to do
00000016 707F         moveq    #127,d0
00000018 B280         cmp.l    d0,d1              > 127?
0000001A 620C         bhi      no_shell           too big
0000001C 2F01         move.l   d1,-(sp)           count
0000001E 2F0B         move.l   head,-(sp)         base
00000020 487A005A     pea      done(pc)           set up exit..
00000024 60000000     bra      _shsrt             and jump
                      *
                      no_shell:
                      * calculate EOF
00000028 E589         lsl.l    #2,d1
0000002A 49F31800     lea      (head,d1.l),tail
0000002E              page
                      * disrupt ordered lists
0000002E E289         lsr.l    d1                 halve list
00000030 08810001     bclr     #1,d1              odd length?
                      * take balance from middle
00000034 20331800     move.l   (head,d1.l),balance
00000038 27931800     move.l   (head),(head,d1.l)
0000003C 2680         move.l   balance,(head)
                      *
                      * 'tail' searches back for elements
                      * less than/equal to the balance
                      *
                      ptr_rgt:
                      * backup pointer, compare
0000003E B0A4         cmp.l    -(tail),balance
00000040 6DFC         blt      ptr_rgt            too big
00000042 6604         bne      pr2                just right
                      * it may be the collision
00000044 B7CC         cmp.l    head,tail
00000046 6314         bls      collide            ..it is
                      pr2:
                      * swap, advance head pointer
00000048 26D4         move.l   (tail),(head)+
0000004A 2880         move.l   balance,(tail)
                      *
                      * 'head' searches forward for values
                      * greater than/equal to the balance
                      *
                      ptr_lft:
                      * compare, advance pointer
0000004C B09B         cmp.l    (head)+,balance
0000004E 6EFC         bgt      ptr_lft            too small
00000050 6706         beq      pl2                collision?
                      * backup pointer, swap
00000052 28A3         move.l   -(head),(tail)
00000054 2680         move.l   balance,(head)
                      * wake up the other pointer
00000056 60E6         bra      ptr_rgt
                      *
                      pl2:
00000058 B9CB         cmp.l    head,tail          collision?
0000005A 62F0         bhi      ptr_lft            a duplicate!
                      *
                      collide:
                      * balance belongs in low partition,
                      * so move tail pointer
0000005C 584C         add.w    #4,tail
0000005E              page
                      * recursive calls;
                      * sort the low partition
0000005E 200C         move.l   tail,d0            get count..
00000060 90AE0008     sub.l    base(fp),d0        in bytes
00000064 E488         lsr.l    #2,d0              downscale
                      * also get count for high partition
00000066 91AE000C     sub.l    d0,count(fp)
0000006A 2F00         move.l   d0,-(sp)
0000006C 2F2E0008     move.l   base(fp),-(sp)
00000070 618E         bsr      _qksrt
00000072 504F         add.w    #8,sp
                      * now sort the other one
```

```
00000074 2F2E000C                 move.l  count(fp),-(sp)
00000078 2F0C                     move.l  tail,-(sp)
0000007A 6184                     bsr     _qksrt
                                  *
                                  done:
                                  * two exits; first adjusts stack
0000007C 504F                     add.w   #8,sp
                                  exit:
0000007E 4CDF1800                 movem.l (sp)+,a3-a4
00000082 4E5E                     unlk    fp
00000084 4E75                     rts
                                  *
00000086                          end
```

SARDIS
ST-2900

Dear Mr. Williams:

I just wanted to add something to the Bit-Bucket so that other owners of Sardis ST-2900 compact cpu's won't have as much trouble as I did getting on-line to BBS.

I am using Xcom9 Modem Program version 10 for OS-9 level 1. Nothing fancy, but since in assembly language, very fast. It was written by one of our Canadian neighbors, Greg Morse. Just write, SASE please, and I'll give you the details and address.

Thanks, [illegible]

Phil Chadwick
Route 2 Box 102
New Hope, Pa. 18938

CONTACT: E. Kyle Tyner 1-800-622-7627

DOVE
COMPUTER CORPORATION

1200 North 23rd St. □ Wilmington, N.C. 28405 □ Telephones: In State 919-763-7918, Out of State TOLL FREE 1-800-622-7627

Dove Computer Corporation, manufacturer of productivity enhancement products for the Macintosh and industry leader in 1 megabit CMOS SIMM technology, has announced a special rebate program for dealers and distributors.

DOVE's Memory Products Division will offer trade-in credit to dealers and distributors for the 256K SIMMs which are replaced and returned to DOVE in conjunction with sales of their MacSnap™ 1 megabit SIMM products. Credits will be allowed on a 1:1 basis versus MacSnap purchases.

According to Rick Greene, Vice President of Sales and Marketing, this program is intended to be managed by the dealers and distributors to their desired advantage. "It affords our dealers and distributors the opportunity to offer an additional discount to the end purchaser and to increase their sales margin," Greene said. "And it eliminates the waste of 256K SIMMs which were not being utilized after a Macintosh Plus, SE or Macintosh II had been upgraded with our 1 Megabit CMOS SIMM products."

"We are always looking for ways to improve productivity and to make the most of resources available to us. We're excited about this program because it will benefit everyone--our dealers and distributors, the end users and DOVE COMPUTER."

DOVE COMPUTER CORPORATION, located in Wilmington, North Carolina, designs, manufactures and markets memory, communications and software products for the Apple Macintosh line of computers. The Company's products are used primarily in business applications and are sold through dealers.

SierraSystems

6728 Evergreen Ave • Oakland, California 94611 • 415-339-8200

FOR RELEASE: Immediately

CONTACT: Larry Rosenthal
(415) 339-8200

SIERRA SYSTEMS ANNOUNCES HIGH PERFORMANCE 68020/68881 C CROSS DEVELOPMENT SYSTEM.

OAKLAND, CA -- Sierra Systems today announced the release of Sierra C, a C language compiler and cross development package for the Motorola 68020/68881 embedded systems market.

"Sierra C proves that highly optimized code can be produced without slowing down the development cycle," said Sierra Systems owner Larry Rosenthal. "Compile, assemble, link and download is typically 20 times faster than on comparable cross development packages. On the other hand, the code produced runs faster and is more compact than the code produced by current leaders in the optimized cross compiler market."

Sierra C includes a compiler, assembler, linker, downloaders, complete C run-time library, file system, floating point emulation, librarian, command driver, and object code utilities. Hosts currently include the IBM PC/AT under MS-DOS and the DEC VAX under VMS and UNIX. The user interface is identical accross all hosts.

The package has been designed specifically for embedded systems applications, providing features like ROMable, position independent and re-entrant code, run-time initialization of static data, and support for resident libraries and highly fragmented address spaces.

The PC/AT hosted version of Sierra C runs at speeds comparable to VAX workstations. The AT version is provided with an interface cable (patent pending) which permits parallel downloads to the target system at speeds far greater than traditional serial methods. With the parallel downloader, a 100,000 byte executable object can be downloaded in two seconds. A serial download of the same program using S-records at 9600 baud would take over five minutes.

The compiler is full K & R and supports proposed ANSI extensions such as structure passing and the "void", "enum", "const" and "volatile" types. Compiler optimizations include loop invariant removal, strength reductions, common subexpression removal, folding of both integer and floating point types, global register allocation by coloring, dead code removal, branch siting, and optional frame removal.

"Sierra C runs at interactive speeds, letting programmers make and test many more modifications per day than with other systems," said Rosenthal. "This greatly increases programmer productivity and shortens overall development time."

Prices for Sierra C are $2000 for the PC/AT and $4000 to $10000 for VAX based systems. Scheduled discounts are available for multiple copy purchases. Sierra C is available for shipment today.

Located in Oakland, California, Sierra Systems is a growing company dedicated to providing tools for the embedded systems market. Previously engaged in the design of high speed graphics systems, Sierra Systems developed Sierra C as a result of its disenchantment with existing cross development tools.

Classifieds *As submitted - No Guarantees*

DAISY WHEEL PRINTERS
Qume Sprint 9 - $900 Qume Sprint 5 - $800

HARD DISK 10 Megabyte Drive - Seagate Model #412 $275.

3 - Dual 8" drive enclosure with power supply . New in box . $125 each.

5 - Siemens 8" Disk Drive , $100 each.

Tano Outpost II, 56K, 2 5" DSDD Drives, FLEX, MUMPS, $495.

TELETYPE Model 43 PRINTER - with serial (RS232) interface and full ASCII keyboard . $250 ready to run.

SWTPC S/09 with Motorola 128K RAM, 1-MPS2, 1-Paralell Port, MP-09CPU Card - $900 complete.

(615) 842-4600 M-F 9AM to 5PM EST

•••

SWTPC 6809 system 5709 , dual 8" drives, 8212 terminal, 128K RAM, Okidata wide carriage with tractor feed . $2,000 or best offer.
462-0551 or 536-0018.

•••

Have many 6800 and 6809 bare and working boards and manuals . Send SASE for list.
Ron Mauceri, 2037 Dewberry Ct.,Westlake Village, CA 91361

•••

GMX 68020 Computer. Never used. 16.7 MHz. 2 Meg Ram. floppy. 25 Meg HD. 4 serial. 1 par. With C. Uniflex OS. et. al.
$5500 new. Only $3995. Marc Talisman.
(714) 582-9100. (714) 532-3466 eves.

SK★DOS™

The Generic DOS™ for 68000 applications in

- ★ Industrial Control
- ★ Business Use
- ★ Educational Computing
- ★ Scientific Computing
- ★ Number Crunching
- ★ Dedicated Systems
- ★ Turnkey Systems
- ★ Data Collection
- ★ Single-board Computers
- ★ Bus-oriented Computers
- ★ Graphics Workstations
- ★ One-of-a-kind Systems
- ★ Advanced Hobbyist Use

SK★DOS is a single-user disk operating system for computers using Motorola 32-bit CPUs such as the 68008, 68000, 68010, and 68020. It provides the power of a full DOS, yet is simple and easy to use, and will run on systems from 32K to 16 megabytes. Because SK★DOS is easily implemented on a new system, we call it "The Generic DOS" which allows programs written for one system to be run on many others.

SK★DOS comes with over 40 commands and system programs, including a 6809 emulator which allows 68K SK★DOS to run application programs and languages developed for 6809 SK★DOS and other systems. Assemblers, editors, and higher level language support are available from third party software vendors and through public domain software.

SK★DOS is available for single-copy or dealer sales, as well as OEM licensing. Single copies cost $125 (inquire as to available systems). Extremely attractive OEM licensing terms are also available. An optional Configuration Kit contains a detailed Configuration Manual and two disks of source code for system adaptation, including source code for a system monitor/debug ROM and other programs useful for adapting SK★DOS to new systems.

SK★DOS *is available from*

STAR-K
SOFTWARE SYSTEMS CORPORATION
P.O. BOX 209 · MT. KISCO, NY 10549 · 914/241-0287
TELEX 61060:6774

INDUSTRIAL PASCAL FOR 68000 AND 6809

PCSK is a package that generates code for a 68000 series processor running on a 68000 development system. It includes the compiler, assembler, linker, host debugger, target debugger, and screen editor, all integrated together and controlled by a enu driven shell program. Source code is included for the runtime library and many of the utilities. Host operating systems supported are OS-9/68000 (Microware), PDOS (Eyring Research), and VERSAdos (Motorola).

PXK9 is a package that generates code for a 6809 processor running on a 68000 development system. Includes all of the features of the PCSK package above, except for the host debugger. Host operating system is OS-9/68000.

I WANT IT, WHERE DO I GET IT?

For more information on either of these two products please contact Certified Software, South East Media, or one of our European Licensees.

OEM LICENSEES

Gespac sa. 3, chemin des Aulx, CH-1228 Geneva/Plan-les-Ouates, Switz. TEL (022) 713400, TLX 429989

PEP Elektronik Systeme GmbH, Am Klosterwald 4, D-8950 Kaufbeuren, West Germany. TEL (08341) 8974, TLX 541233.

Ellec Elektronik GmbH, Galileo-Galilei-Strasse, 6500 Mainz 42, Postfach 65, West Germany. TEL (06131) 50031, TLX 4187273.

DISTRIBUTORS

R.C.S. Microsystems Ltd. 141 Uxbridge Road, Hampton Hill, Middlesex, England. TEL 01-9792204, TLX 8951470.

Dr. Rudolf Keil GmbH, Porphystrasse 15, D-6905 Schriesheim, West Germany. TEL 062 03/6741, TLX 465025.

Elsoft AG, Zelgweg 12, CH-5405 Baden-Daettwil, Switzerland. TEL 056-833377, TLX 828275.

Byte Studio Borken, Butenwall 14, D-4280 Borken, West Germany. TEL 02861-2147, TLX 813343.

CERTIFIED SOFTWARE CORPORATION 616 CAMINO CABALLO, NIPOMO, CA 93444
TEL: (805) 929-1395 TELEX: 467013
FAX: (805) 929-1395 (MID-8AM)

SOFTWARE FOR 680x AND MSDOS

SUPER SLEUTH DISASSEMBLERS
EACH $99-FLEX $101-OS/9 $100-UNIFLEX
OBJECT-ONLY versions: EACH $50-FLEX,OS/9,COCO
interactively generate source on disk with labels, include xref, binary editing
specify 6800,1,2,3,5,8,9/6502 version or Z80/8080,5 version
OS/9 version also processes FLEX format object file under OS/9
COCO DOS available in 6800,1,2,3,5,8,9/6502 version (not Z80/8080,5) only
NEW: 68010 disassembler $100-FLEX,OS/9,UNIFLEX,OS/9-68K,MSDOS

CROSS-ASSEMBLERS WITH MACRO CAPABILITIES
EACH $50-FLEX,OS/9,UNIFLEX,MSDOS,UNIX 3/$100 ALL/$200
specify 180x,6502,6801,6804,6805,6809,Z8,Z80,8048,8051,8085,68010,32000
modular cross-assemblers in C, with load/unload utilities NOW: OS/9-68K
sources for additional $50 each, $100 for 3, $300 for all

DEBUGGING SIMULATORS FOR POPULAR 8-BIT MICROPROCESSORS
EACH $75-FLEX $100-OS/9 $80-UNIFLEX
OBJECT-ONLY versions: EACH $50-COCO FLEX,COCO OS/9
interactively simulate processors, include disassembly formatting, binary editing
specify for 6800/1, (14)6805, 6502, 6809 OS/9, Z80 FLEX

ASSEMBLER CODE TRANSLATORS FOR 6502, 6800/1, 6809

6502 to 6809	$75-FLEX $85-OS/9 $80-UNIFLEX
6800/1 to 6809 & 6809 to position-ind	$50-FLEX $75-OS/9 $60-UNIFLEX

FULL-SCREEN XBASIC PROGRAMS with cursor control
AVAILABLE FOR FLEX, UNIFLEX, AND MSDOS

DISPLAY GENERATOR/DOCUMENTOR	$50 w/source, $25 without
MAILING LIST SYSTEM	$100 w/source, $50 without
INVENTORY WITH MRP	$100 w/source, $50 without
TABULA RASA SPREADSHEET	$100 w/source, $50 without

DISK AND XBASIC UTILITY PROGRAM LIBRARY
$50-FLEX $30-UNIFLEX/MSDOS
edit disk sectors, sort directory, maintain master catalog, do disk sorts,
resequence some or all of BASIC program, xref BASIC program, etc.
non-FLEX versions include sort and resequence only

CMODEM TELECOMMUNICATIONS PROGRAM
$100-FLEX,OS/9,UNIFLEX,MS-DOS,OS/9-68K,UNIX
OBJECT-ONLY versions: EACH $50
menu-driven with terminal mode, file transfer, MODEM7, XON-XOFF, etc.
for COCO and non-COCO; drives internal COCO modem port up to 2400 Baud

DISKETTES & SERVICES

5.25" DISKETTES
EACH 10-PACK $12.50-SSSD/SSDD/DSDD
American-made, guaranteed 100% quality, with Tyvek jackets, hub rings, and labels

ADDITIONAL SERVICES FOR THE COMPUTING COMMUNITY
CUSTOMIZED PROGRAMMING
we will customize any of the programs described in this advertisement or in our brochure for specialized customer use or to cover new processors; the charge for such customization depends upon the marketability of the modifications.

CONTRACT PROGRAMMING
we will create new programs or modify existing programs on a contract basis, a service we have provided for over twenty years. the computers on which we have performed contract programming include most popular models of mainframes, including IBM, Burroughs, Univac, Honeywell, most popular models of minicomputers, including DEC, IBM, DG, HP, AT&T, and most popular brands of microcomputers, including 6800/1, 6809, Z80, 6502, 68000, using most appropriate languages and operating systems, on systems ranging in size from large telecommunications to single board controllers; the charge for contract programming is usually by the hour or by the task.

CONSULTING
we offer a wide range of business and technical consulting services, including seminars, advice, training, and design, on any topic related to computers; the charge for consulting is normally based upon time, travel, and expenses.

Computer Systems Consultants, Inc.
1454 Latta Lane, Conyers, GA 30207
Telephone 404-483-4570 or 1717

We take orders at any time, but plan long discussions after 6, if possible.

Contact us about catalog, dealer, discounts, and services.
Most programs in source: give computer, OS, disk size.
25% off multiple purchases of same program on one order.
VISA and MASTER CARD accepted; US funds only, please.
Add GA sales tax (if in GA) and 5% shipping.
(UNI)FLEX tm Technical Systems Consultants; OS/9 Microware; COCO Tandy; MSDOS Microsoft.

Ωmega . . . The Desktop Mainframe

* 12.5 MHz MC68020 32-bit microprocessor & MC68881 FPCP as standard.
* 1 megabyte of non-volatile, zero wait-state STATIC RAM and 128k/256k bytes ROM.
* 25.5 megabyte (unformatted) Winchester Hard disc as standard.
* 1.0 megabyte (unformatted) DS/DD 80 track floppy disc drive (specify 5¼" or 3½").
* Five RS-232C serial ports (one may be configured as an RS-485 network node).
* One Parallel printer port and one 16-bit bi-directional parallel port.
* Non-volatile clock calendar and power supply supervisor.
* OS-9/68K™ 'Professional' with full MC68881 FPCP support and 'C' compiler.

OMEGA/12.5/OS	OMEGAw/12.5 MHz 020/881 & OS-9/68K™	$4750.00
OMEGA/16.6/OS	OMEGA w/16.67 MHz 020/881 & OS-9/68K™	$4950.00
OMEGA/1MB	1 megabyteadditional STATICRAM	$ 895.00
OMEGA/SER9	9 port RS-232C serial expansion board	$ 645.00
OMEGA/GRF	640 x 480 x 4 bits/pixel ACRTC graphics I/F	$ 925.00

PRICES INDICATED INCLUDE SHIPPING AND APPLY TO U.S. CUSTOMERS ONLY

VISA/MASTER CARD ORDERS ACCEPTED

OS-9 is a trademark of Microware Systems Corporation

NON-U.S. CUSTOMERS SHOULD CONTACT THE NEAREST DEALER FOR PRICE & DELIVERY INFORMATION

NORTH & SOUTH AMERICA	ALL OTHER ENQUIRIES	CONTINENTAL EUROPE
LLOYD I/O INCORPORATED 19535 NE GLISAN P.O. Box 30945 PORTLAND, OR 97230 (USA) TEL: (503) 666-1097 TLX: 9103805448 LLOYD I O	WINDRUSH MICRO SYSTEMS WORSTEAD LABS. N. WALSHAM, NORFOLK NR28 9SA, ENGLAND TEL: (0692) 404086 TLX: 975548 WMICRO-G	SNIJDER MICRO SYSTEMS SCHOOTEINDSEWEG 8a 5756 BD Vlierden The NETHERLANDS TEL: (0)4930-11975-13666

The C Users' Group

Public Domain Software Library Directory

Now Available!

- Complete descriptions and cross-referenced index for thousands of C Source Programs in The C Users' Group Library.
- A quick and handy reference for locating public domain editors, text formatters, compilers, UNIX-like tools, etc., all in C.
- Indexes programs for CP/M, UNIX, and MS-DOS.

TO ORDER: Send $10 (US Dollars) to:

The C Users' Group
PO Box 97
115 E. Kansas Ave. Ste. 206
McPherson, KS
67460 USA
(316) 241-1065

Foreign orders add 30% surcharge
MasterCard & Visa accepted

Name ____________
Address ____________
____
____

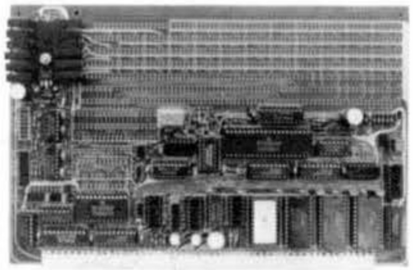
☐ Please send more information about The C Users' Group Membership and Newsletter.

____ M/C ____ Visa # ____________

Exp. date ______ Sign ____________

E-Z On, E-Z Off

The LAB 6809 helps you prototype your design in less time, for less money. Easy to use and easy to modify, the LAB 6809 is the optimal development tool for 6809-based applications with high performance objectives.



☐ Use as stand-alone or system CPU ☐ 68B09 jumper selectable for 1 or 2 MHz ☐ 2 MHz I/O without wait states for on-board devices ☐ 68B50 RS-232C serial port ☐ on-board provisions for additional 6850, 6840, 6821, other I/O devices ☐ PICOBUG monitor provides I/O and debugging routines ☐ 22 sq. in. user wirewrap area ☐ gold-plated SS-50 connectors ☐ comprehensive documentation

Put the LAB 6809 to work on your next project. $395.

Shipping and handling $10 (US), $20 (outside US). Texas residents add sales tax. MasterCard, VISA accepted. Technical bulletin available on request. Dealer inquiries invited.

Barton Laboratories INTELLIGENT TOOLS FOR INTELLIGENT CONTROLS

1606 West 14th Street Austin, TX 78703 **512/477-6809**

Installed Systems World-Wide
OVER 10 YEARS OF DEDICATED QUALITY!

CPI *A Division of Computer Publishing, Inc.*
5900 Cassandra Smith Road
Hixson, Tn 37343
Telephone 615 842-4600
Telex 510 600-6630

DATA-COMP
SPECIAL
Heavy Duty Power Supplies

For A limited time we are offering our HEAVY DUTY SWITCHING POWER SUPPLY. These are BRAND NEW units and will not last long. Also note that these prices are less than 1/4 the normal price for these high quality unit.

Make: Boschert

Size: 10.5 x 5 x 2.5 inches - including heavy mounting bracket and heatsink.

Rating: in 110/220 volts ac (strap change) Out: 130 watts

Output: +5v - 10 amps
+12v - 4.0 amps
+12v - 2.0 amps
-12v - 0.5 amps

Mating Connector: Terminal strip
Load Reaction: Automatic short circuit recovery

Each
SPECIAL: $59.95
2 or more 49.95

Add: $7.50 each S/H

Make: Boschert

Size: 10.75 x 6.2 x 2.25 inches

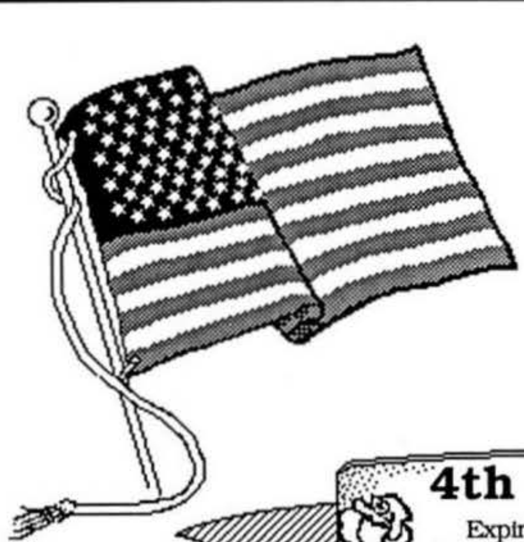
Rating: 110/220 ac (strap change) Out: 81 watts

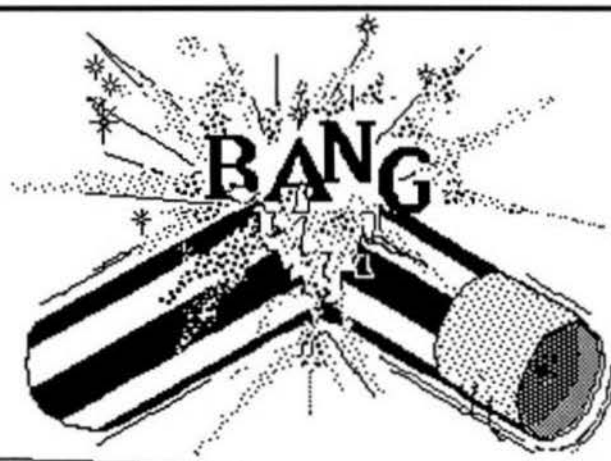
Outputs: +5v - 8.0 amps
+12v - 2.4 amps
+12v - 2.4 amps
+12v - 2.1 amps
-12v - 0.4 amps

Mating Connectors: Molex
Load Reaction: Automatic short circuit recovery

Each
SPECIAL: $49.95
2 OR MORE 39.95

Add: $7.50 S/H each





4th July Sale

Expires July 31, 1987

S. E. Media

Savings to 50%

And More!

All other items **10%** off - Sale ends 7/31/87
Add: $2.50 S/H all software, each item.

ITEM	Prices Reg	Special
CoCo/OS-9 BASIC Xref w/source	79.95	49.95
CoCo/OS-9 BASIC Xref w/o source	39.95	24.95
OSM - 6809 Assembler Lloyd I/O	99.95	49.95
Search & Rescue - Lloyd I/O OS-9	35.00	24.95
C COMPILER - Windrush FLEX	295.00	195.00
BIZPACK Vairginia Co. FLEX		
VIRTUAL TERMINAL CoCo/OS-9	49.95	36.95
CoCo KANSAS CITY BASIC	39.95	24.95
LSORT OS-9 Sort/merge I II	85.00	54.95
BAS-EDIT FLEX (TSC BASIC util)	39.95	19.95
ASSEMBLERS Assorted Loyd I/O	discounted call	
K-BASIC Compiler compiles XBASIC - FLEX - OS-9		
very popular BASIC compiler..	199.95	79.95
DYNACALC Great spreadsheet	200.00	99.95
DYNAMITE FLEX OS-9 CoCo all	varies	49.95
STYLO Word Pro CoCo OS-9	99.95	49.95
PL9 5" FLEX	198.00	129.95
SCREDITOR III 5"	175.00	119.95

AR Great Plains FLEX	295.00	129.50
RRMac Great Plains FLEX	150.00	74.95
INFOMAG Great Plains UniFLEX	395.00	99.50
SPELLING CHECKER FLEX	99.95	49.95
MAIL MERGE FLEX	79.95	39.95
C Compiler Introl FLEX - OS-9	575.00	295.00
REL MACRO ASMB Microware OS-9	200.00	99.50
CHECK & TAX Star-Kits FLEX	50.00	19.95
STAR-DOS CoCo	34.50	19.50
HUMBUG Star-Kits (tape)	40.95	9.95
REMO-TERM CoCo (tape) Remote CRT	?	9.95
SPELL 'N FIX Star-Kits FLEX	89.29	39.95

ASSORTED TSC 6809 FLEX UniFLEX 40-75 % off

Assorted items: (some new - some used working)
SWTPC SO-9 Computers 128K 3000.00 75% off
Assorted boards, etc. up to 90% off - call

NOTE - STAR-DOS-SK-DOS runs same as FLEX

See S.E. Media Catalog - page 29 for descriptions

THE 6800-6809 BOOKS

..HEAR YE.....HEAR

OS-9™ User Notes

By: Peter Dibble

The publishers of 68' Micro Journal are proud to make available the publication of Peter Dibbles **OS9 USER NOTES**

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

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

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

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

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

BOOK $9.95

Typeset -- w/ Source Listings
(3-Hole Punched; 8 x 11)
Deluxe Binder - - - - - - - - - - - $5.50

All Source Listings on Disk

1-8" SS, SD Disk - - - - $14.95
2-5" SS, DD Disks - - - $24.95

FLEX™ USER NOTES

By: Ronald Anderson

The publishers of 68 MICRO JOURNAL are proud to make available 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. 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.

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

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

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

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

Shipping & Handling $3.50 per Book, $2.50 per Disk set
Foreign Orders Add $4.50 Surface Mail
or $7.00 Air Mail
If paying by check - Please allow 4-6 weeks delivery
*** All Currency in U.S. Dollars**

Continually Updated In 68 Micro Journal Monthly

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

VISA

(615) 842-4601
Telex 5106006630

"FLEX is a trademark of Technical Systems Consultants
"OS9 is a trademark of Microware and Motorola
"68' Micro Journal is a trademark of Computer Publishing Inc.

OS-9™ SOFTWARE

L1 UTILITY PAK—Contains all programs formerly in Filter kits 1 & 2, and Hacker's kit 1 plus several additional programs. Complete "wild card" file operations, copies, moves, sorts, del, MACGEN shell command language compiler, Disassembler, Disk sectoredit utility, new and improved editions, approx. 40 programs, increases your productivity. Most programs applicable for both level I & II 6809 OS-9. **$49.95** ($51.95)

Call or send Self Addressed Stamped Envelope for catalog of software for colorComputer OS-9 and other OS-9 systems.

BOLD prices are CoCo OS-9 format disk, other formats (in parenthesis) specify format. All orders prepaid or COD, VISA and MasterCard accepted. Add $1.50 S&H on prepaid, COD actual charges added.

SS-50C
MEMORY LIQUIDATION SALE!
(While Supply Lasts)

1 MEGABYTE RAM BOARD

Full megabyte of ram with disable options to suit any SS-50 6809 system. High reliability, can replace static ram for fraction of the cost. $399 for 2 Mhz or $439 for 2.25 Mhz board assembled, tested and fully populated.

2 MEGABYTE RAM DISK BOARD

RD2 2 megabytes dedicated ram disk board for SS-50 systems. Four layer circuit board socketed for 2 **Megabytes!** Special sale price of **$399.00** includes only 256k of ram installed (you add the rest), includes OS-9 level I and II drivers for Ram disk, (note: you can re-boot your system without losing ram-disk contents). (Add $6 shipping and insurance.)

Please call for answers to your technical questions concerning these products.

D.P. Johnson, 7655 S.W. Cedarcrest St.
Portland, OR 97223, (503) 244-8152
(For best service call between 9-11 am Pacific time.)

OS-9 is a trademark of Microwave and Motorola Inc.
MS-DOS is a trademark of Microsoft Inc.

COMPILER EVALUATION SERVICES

BY: Ron Anderson

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

COMPILER COMPARISON AND EVALUATION REPORT

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

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

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

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

PASCAL 'C' GSPL WHIMSICAL PL/9

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

S.E. MEDIA - C.P.I.
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4601

Call for Software:

68000, C, Basic09
Sculptor

We are receiving calls and letters from numerous sources, including users, business and others looking for OS-9 68000 software; applications, etc.

Many of you have developed software that with little change could be adapted for others. If you are interested in selling it, please let us know. There is a growing market out there now. Get in on the ground floor!

If you can use additional income and have something that might be of interest, call and talk to Larry or Don.

S.E. MEDIA Division - CPI

POB 849
Hixson, TN 37343

Telephone (615) 842-6809
Telex (510) 600-6630

Stop!

Get a 25 Mega-Byte Hard Disk practically FREE only 1¢

***Be Sure to Consider the SPECIAL MUSTANG-08/A 1¢ Sale on page 7**

*This is exactly one cent more than the price of
the same system - with two floppies - for one cent
more you get one floppy and a 25 MegaByte Hard Disk with the faster
CPU board, additional serial ports and improved clock!
Includes Professional OS-9™ Version 2 and the $500.00 C Compiler!

Remember - When it's over, IT'S OVER!

We don't know how long this very, very low price can be maintained, don't miss it!

Data-Comp Div. - CPI

6809<>68XXX UniFLEX

X-TALK

A C-MODEM/Hardware Hookup

Exclusive for the MUSTANG-020 running UniFLEX, is a new transfer program and cable set from DATA-COMP (CPI). X-TALK consist of 2 disks and a special cable, this hook-up enables a 6809 SWTPC UniFLEX computer to port UniFLEX files directly to a 68XXX UniFLEX system.

This is the only currently available method to transfer files, text or otherwise, from a 6809 UniFLEX system to a 68000 UniFLEX system, that we have seen. A must if you want to recompile or cross assemble your old (and valuable) source files to run on a 68000 UniFLEX system. GIMIX users can directly transfer files between a 6809 GIMIX system and our MUSTANG-020 68020 system, or GIMIX 68020 system. All SWTPC users must use some sort of method other than direct disk transfer. The 6809 SWTPC UniFLEX disk format is not readable by most other 68000 type systems.

The cable is specially prepared with internal connections to match the non-standard SWTPC SO/9 DB25 connectors. A special SWPTC+ cable and software is also available, at the same price. Orders must specify which type SWTPC 6809 UniFLEX system they intend to transfer from or to.

The X-TALK software is furnished on two disks. One 8" disk containing the 6809 software and one 5" disk containing the 68XXX software. These programs are also complete MODEM programs and can be used as such, including X-on X-off, and all the other features you would expect from a full modem program.

X-TALK can be purchased with/without the special cables, however, this SPECIAL price is available only to registered MUSTANG-020 owners.

X-TALK, w/cable	**$ 99.95**
X-TALK only	**69.95**
X-TALK w/source	**$149.95**

DATA-COMP
5900 Cassandra Smith Rd.
Hixson, TN 37343

Telephone 615 842-4601
Telex 510 600-6630

Note: Registered MUSTANG-020 owners must furnish system serial number in order to buy at these special low prices.

68 MICRO JOURNAL
Reader Service Disks

Disk- 1 Filesort, Minicat, Minicopy,Minifms, **Lifetime, **Poetry, **Foodlist, **Diet.
Disk- 2 Diskedit w/ inst.& fixes, Prime, *Prnod, **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, *Dskalign, Help, Date.Txt.
Disk-14 *Init, *Test, *Terminal, *Find, *Diskedit, Init.Lib
Disk-15 Modem9 + Updates (Dec. 84 Gilchrist) to Modem9 (April 84 Commo).
Disk-16 Copy.Txt, Copy.Doc, Cat.Txt, Cat.Doc.
Disk-17 Match Utility, RATBAS, A Basic Preprocessor.
Disk-18 Parse.Mod, Size.Cmd (Sept. 85 Armstrong), CMDCODE, CMD.Txt (Sept. 85 Spray).
Disk-19 Clock, Date, Copy, Cat, PDEL.Asm & Doc., Errors.Sys, Do, Log.Asm & Doc.
Disk-20 UNIX Like Tools (July & Sept. 85 Taylor & Gilchrist). Dragon.C, Grep.C, LS.C, FDUMP.C.
Disk-21 Utilities & Games - Date, Life, Madness, Touch, Goblin, Starshot, & 15 more.
Disk-22 Read CPM & Non-FLEX Disks. Fraser May 1984.
Disk-23 ISAM, Indexed Sequential file Accessing Methods, Condon Nov. 1985. Extensible Table Driven. Language Recognition Utility, Anderson March 1986.
Disk-24 68' Micro Journal Index of Articles & Bit Bucket Items from 1979 - 1985, John Current.
Disk-25 KERMIT for FLEX derived from the UNIX ver. Burg Feb. 1986. (2)-5" Disks or (1)-8" Disk.
Disk-26 Compacta UniBoard review, code & diagram, Burlison March '86.
Disk-27 ROTABIT.TXT, SUMSTEST.TXT, CONDATA.TXT, BADMEN.TXT.
Disk-28 CT-82 Emulator, bit mapped.
Disk-29 **Star Trek
Disk-30 Simple Winchester, Dec.'86 Green.
Disk-31 *** Read/Write MS/PC-DOS (SK*DOS)
Disk-32 Hier-UNIX Type upgrade - 68MJ 2/87

NOTE:
This is a reader service ONLY! No Warranty is offered or implied, they are as received by 68' Micro Journal, and are for reader convenience ONLY (some MAY include fixes or patches). Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other. Software is available to cross-assemble all.

* Denotes 6800 - ** Denotes BASIC
*** Denotes 68000 - 6809 no indicator.

Specify 8" disk $19.50
5" disk $16.95

Add: S/H - $3.50
Overseas add: $4.50 surface - $7.00 Air Mail, USA Dollars

68 MICRO JOURNAL
PO Box 849
Hixson, TN 37343
615 842-4600 - Telex 510 600-6630

IF YOU NEED IT, WE'VE GOT IT! (OR WE'LL MAKE IT)

Yes, PERIPHERAL TECHNOLOGY still sells the FD-2 for SS-50 Bus Computers. And, if you don't need it, we sell other products from Single Board Computers to Systems which should fit your requirements. Custom Hardware Design is also available. Here's a small sample of what we offer:

FD-2 FLOPPY DISK CONTROLLER

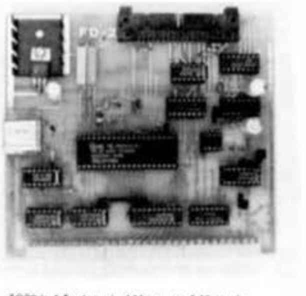
- Controls up to four 5¼" Drives
- Runs in 1 or 2 MHZ Systems
- Can be configured for either 4 or 16
- Addresses per I/O Slot (SS30 or SS30C)
- Uses WD2797 Controller Chip (compatible with 1771/179X Controller Chip)
- Hardware and Software compatible with SWTPC DC-4 controllers
- 6800/6809 Flex Drivers available
- SK*DOS Operating System or OS9/6809 Driver package

PT69-4 SINGLE BOARD COMPUTER

- 6809E Processor/1 MHZ Clock
- Four RS232 Serial Ports using 6850'S
- Two 8-Bit parallel Ports using 6821 PIA
- Time-Of-Day Clock (MC146818)
- 59K of user RAM
- 2K or 4K of EPROM using 2716 or 2732
- Double Sided/Double Density Floppy Controller
- Can Read/Write Radio Shack OS/9 Diskettes
- Board Size 5.6" X 6.2"

SYSTEMS

- Floppy or Winchester Versions Available
- Systems use any 6809 Single Board Computer or 68008 Board
- Will be configured to meet your Requirements
- OS9 & SK*DOS Operating Systems
- Call or write for system configurations



MONTHLY SPECIAL
PT68K-1 10 Mhz $375
Regular Price $495

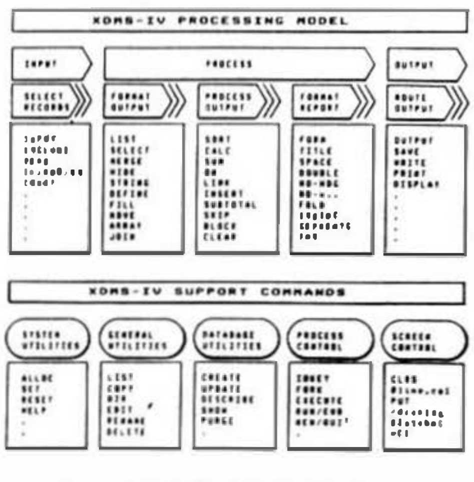
PERIPHERAL TECHNOLOGY
1480 Terrell Mill Road, Suite 870
Marietta, Georgia 30067
(404) 984-0742 Telex # 880584
VISA / MASTERCARD / CHECK / C.O.D.

*OS9 Is A Trademark of Microware & Motorola
*SWTPC Is A Trademark of Southwest Technical Systems

Catalogues Available Upon Request.

XDMS-IV

Data Management System



Save $100.00 - Limited Time
Regular $350.00 - ~~Now~~ Only
$249.95

Technical telephone assistance: Tel 914-941-3552 (Evenings)
FLEX™ Tecnical Systems Consultants, SK*DOS™ STAR-KITS Corp.

FOR 6809 FLEX-SK-DOS(5/8")

Up to 32 groups/fields per record! Up to 12 character filed name! Up to 1024 byte records! User defined screen and print control! Process files! Form files! Conditional execution! Process chaining! Upward/Downward file linking! File joining! Random file virtual paging! Built in utilities! Built in text line editor! Fully session oriented! Enhanced forms! Boldface, Double width, Italics and Underline supported! Written in compact structured assembler! Integrated for FAST execution!

XDMS-IV Data Management System

XDMS-IV is a brand new approach to data management. It not only permits users to describe, enter and retrieve data, but also to process entire files producing customized reports, screen displays and file output. Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotaling, and presentation of up to three related files as a "database" on user defined output reports.

POWERFUL COMMANDS!

XDMS-IV combines the functionality of many popular DBMS software systems with a new easy to use command set into a single integrated package. We've included many new features and commands including a set of general file utilities. The processing commands are Input-Process-Output (IPO) oriented which allows almost instant implementation of a process design.

SESSION ORIENTED!

XDMS-IV is session oriented. Enter "XDMS" and you are in instant command of all the features. No more waiting for a command to load in from disk! Many commands are immediate, such as CREATE (file definition), UPDATE (file editor), PURGE and DELETE (utilities). Others are process commands which are used to create a user process which is executed with a RUN command. Either may be entered into a "process" file which is executed by an EXECUTE statement. Processes may execute other processes, or themselves, either conditionally or unconditionally. Menus and screen prompts are easily coded, and entire user applications can be run without ever leaving XDMS-IV!

IT'S EASY TO USE!

XDMS-IV keeps data management simple! Rather than design a complex DBMS which hides the true nature of the data, we kept XDMS-IV file oriented. The user view of data relationships is presented in reports and screen output, while the actual data resides in easy to maintain files. This aspect permits customized presentation and reports without complex redefinition of the database files and structure. XDMS-IV may be used for a wide range of applications from simple record management systems (addresses, inventory ...) to integrated database systems (order entry, accounting...) The possibilities are unlimited...

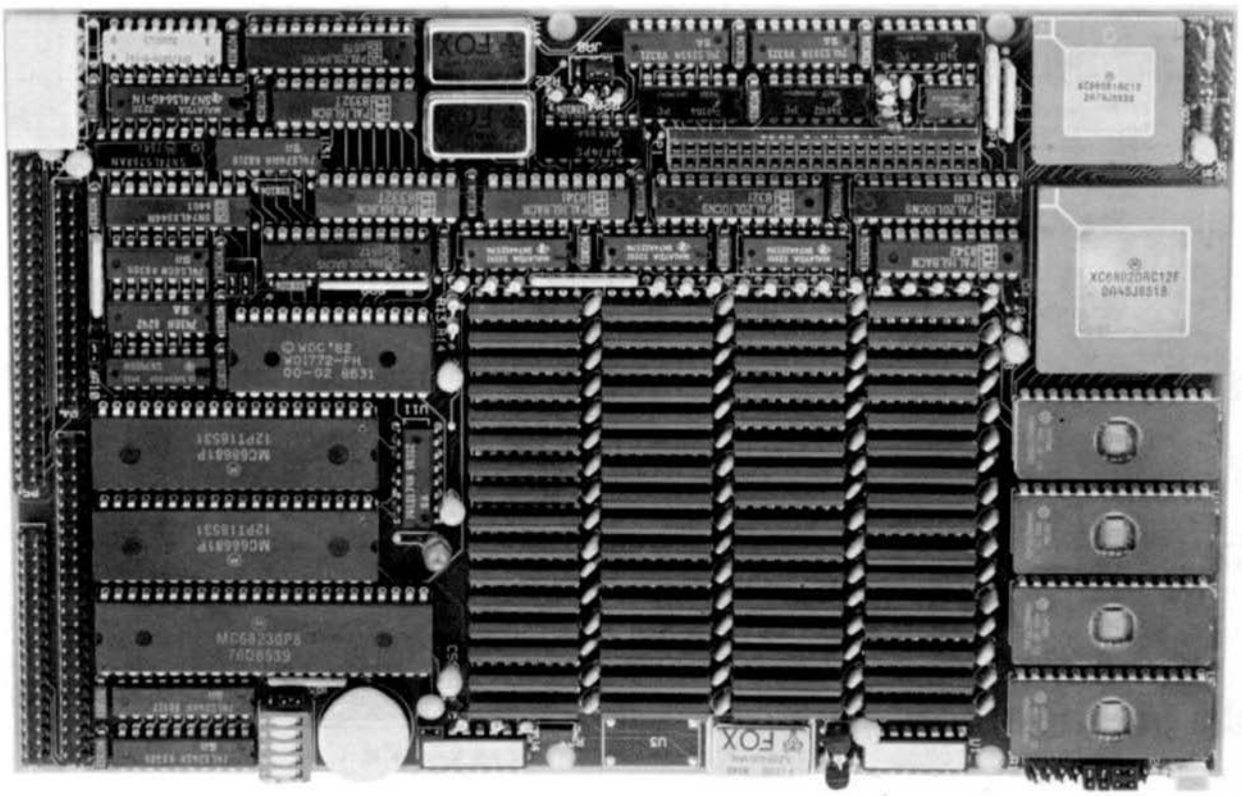
Visa & Master Card Excepted

Telephone: 615-842-4601 or Telex: 510 600-6630
Or Write: S.E. Media, 5900 Cassandra Smith Rd., Hixson, Tenn. 37343

GMX™ Micro-20
68020 SINGLE—BOARD COMPUTER

Mainframe CPU Performance on a 5.75" x 8.8" Board
(benchmark results available on request)

$2565⁰⁰ 12.5 MHz Version
Quantity Discounts Available



Features

- 32-Bit MC68020 Processor (12.5, 16.67, or 20MHZ)
- MC68881 Floating-point coprocessor (optional)
- 2 Megabytes of 32-bit wide, high-speed RAM
- 4 RS-232 Serial I/O Ports (expandable to 36)
- 8-bit Parallel I/O Port ('Centronics' compatible)
- Time-of-Day Clock w/battery backup
- 16-bit I/O Expansion Bus
- Up to 256 Kbytes of 32-bit wide EPROM
- Floppy Disk Controller for two 5¼" drives
- SASI Intelligent Peripheral Interface (SCSI subset)
- Mounts directly on a 5¼" Disk Drive
- Optional Boards include Arcnet, Prototyping, I/O Bus adapter, 60 line Parallel I/O, RS-422/485

Software

Included:
- GMX Version of Motorola's 020Bug Debugger with up/download, breakpoint, trace, single-step, and assembler/disassembler capabilities
- Comprehensive Hardware Diagnostics

Optional:

UNIX™ like Multi-user/Multi-tasking Disk Operating Systems
- OS-9/68000™ (Real-time and PROMable)
- UniFLEX™

Programming Languages and Application Software
- BASIC, C, PASCAL, ABSOFT FORTRAN, COBOL and ASSEMBLER
- Spreadsheet, Data Base Management, and Word Processing

COMPLETE EVALUATION SYSTEMS AVAILABLE

GMX inc 1337 W. 37th Place Chicago, IL 60609

(312) 927-5510 • TWX 910-221-4055
State-of-the-Art Computers Since 1975

For Ordering Call

(615)842-4600

FROM – DATA-COMP, C.P.I.

Now Offering: *FLEX™ (2 Versions)
AND *STAR-DOS PLUS+™

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo

FLEX-CoCo Sr.
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. $250.00
Only $79.95

STAR-DOS PLUS+
- Functions Same as FLEX
- Reads - writes FLEX Disks
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

$34.95

FLEX-CoCo Jr.
without TSC
Editor & Assembler
$49.95

PLUS

ALL VERSIONS OF FLEX & STAR-DOS• INCLUDE

+ Read-Write-Dir RS Disk
+ Run RS Basic from Both
+ More Free Utilities
+ External Terminal Program
+ Test Disk Program
+ Disk Examine & Repair Program
+ Memory Examine Program
+ Many Many More!!!

TSC Editor
Reg $50.00
NOW $35.00

TSC Assembler
Reg $50.00
NOW $35.00

CoCo Disk Drive Systems

2 THINLINE DOUBLE SIDED DOUBLE DENSITY DISK DRIVES SYSTEM WITH POWER SUPPLY,CABINET,DISK DRIVE CABLE,J&M NEW DISK CONTROLLER JFD-CP WITH J-DOS,RS-DOS OPERATING SYSTEMS.	$469.95

* Specify What CONTROLLER You Want J&M, or RADIO SHACK

THINLINE DOUBLE SIDED DOUBLE DENSITY 40 TRACKS	$129.95

Verbatim Diskettes

Single Sided Double Density	$ 24.00
Double Sided Double Density	$ 24.00

Controllers

J&M JFD-CP WITH J-DOS	$139.95
WITH J-DOS, RS-DOS	$159.95
RADIO SHACK 1.1	$134.95
RADIO SHACK Disk CONTROLLER 1.1	$134.95

Disk Drive Cables

Cable for One Drive	$ 19.95
Cable for Two Drives	$ 24.95

MISC

64K UPGRADE FOR C,D,E,F, AND COCO II	$ 29.95
RADIO SHACK BASIC 1.2	$ 24.95
RADIO SHACK DISK BASIC 1.1	$ 24.95
DISK DRIVE CABINET FOR A SINGLE DRIVE	$ 49.95
DISK DRIVE CABINET FOR TWO THINLINE DRIVES	$ 69.95

PRINTERS

EPSON LX-80	$289.95
EPSON MX-70	$125.95
EPSON MX-100	$495.95

ACCESSORIES FOR EPSON

8148 2K SERIAL BOARD	$ 89.95
8149 32K EXPAND TO 128K	$169.95
EPSON MX-RX-80 RIBBONS	$ 7.95
EPSON LX-80 RIBBONS	$ 5.95
TRACTOR UNITS FOR LX-80	$ 39.95
CABLES & OTHER INTERFACES CALL FOR PRICING	

DATA-COMP
5900 Cassandra Smith Rd.
Hixson, TN 37343

SHIPPING
USA ADD 2%
FOREIGN ADD 5%
MIN. $2.50

(615)842-4600
For Ordering
Telex 5106006630

Introducing ....

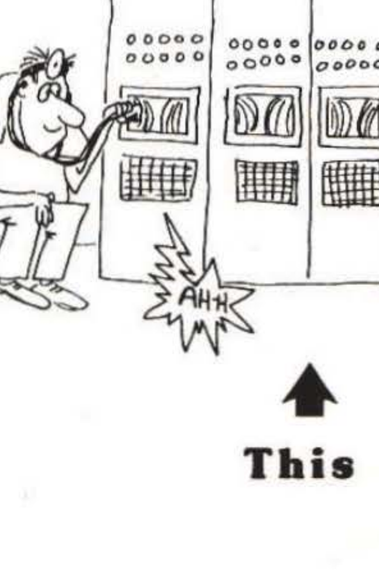
S - 50 BUS / 68XX

Board and/or Computer
Terminals-CRTs-Printers
Disk Drives-etc.

REPAIRS

This

Not This



NOW AVAILABLE TO ALL S50/68XX USERS

The Data-Comp Division of CPI is proud to announce the availability of their service department facilities to **'ALL'** S50 Bus and 68XX users. Including all brands, SWTPC - GIMIX - SSB - HELIX and others, including the single board computers. *Please note that kit-built components are a special case, and will be handled on an individual basis, if accepted.

1. If you require service, the first thing you need to do is call the number below and describe your problem and *confirm a Data-Comp service & shipping number!* This is very important, Data-Comp will not accept or repair items not displaying this number! Also we cannot advise or help you troubleshoot on the telephone, we can give you a shipping number, but **NO** advice! Sorry!

2. All service shipments must include both a minimum $40.00 estimate/repair charge and pre-paid return shipping charges (should be same amount you pay to ship to Data-Comp).

3. If you desire a telephone estimate after your repair item is received, include an additional $5.00 to cover long distance charges. Otherwise an estimate will be mailed to you, if you requested an estimate. Estimates *must be requested.* Mailed estimates slow down the process considerably. However, if repairs are not desired, after the estimate is given, the $40.00 shall constitute the estimate charge, and the item(s) will be returned unrepaired providing sufficient return shipping charges were included with the item to be serviced. Please note that estimates are given in dollar amounts only.

4. Data-Comp service is the oldest and most experienced general S50/68XX service department in the world. We have over $100,000.00 in parts in stock. We have the most complete set of service documents for the various S50/68XX systems of anyone - **YET, WE DO NOT HAVE EVERYTHING!** But we sure have more than anyone else. We repair about 90% of all items we receive. Call for additional information or shipping instructions.

DATA-COMP
5900 Cassandra Smith Rd.
Hixson, TN 37343

(615)842-4607
Telex 5106006630