COMPUTER LANGUAGE™

$2.95

VOLUME 2, NUMBER 1

JANUARY 1985

PROGRAM IN STYLE

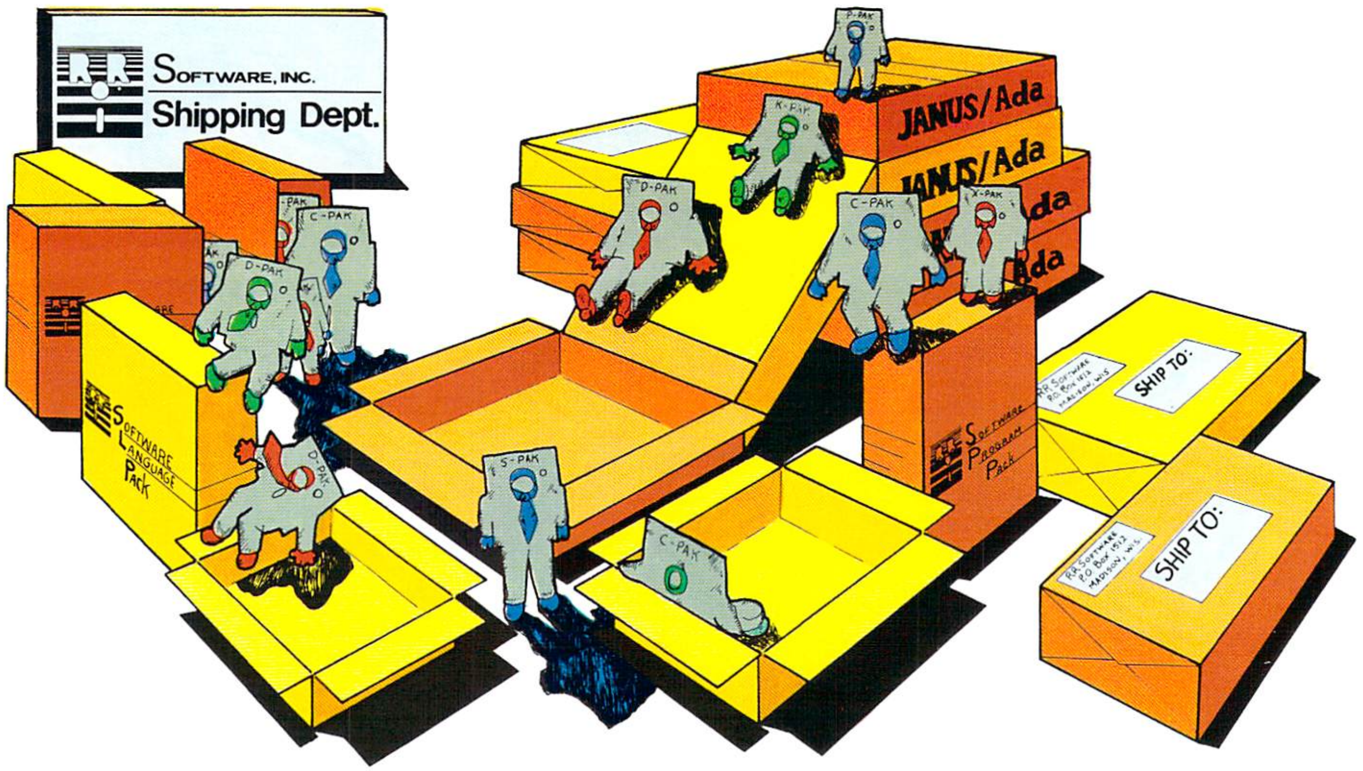
THE dBASE III LANGUAGE

EXTENSIBILITY IN FORTH

LIBES AFTER MICROSYSTEMS

MACROS AND PROCEDURES

WE'VE GOT YOUR PACKAGE!!



We offer you the most flexible, cost efficient means of introducing your programming staff to the Ada Language. **You** can choose the level of Support **you** need, when you need it! These Janus/Ada packages are customer-tested and available now...

(C-Pak) Introductory Janus/Ada Compilers
(D-Pak) Intermediate Janus/Ada Systems
(S-Pak) Advanced Janus/Ada Systems
(P-Pak) Janus/Ada Language Translators

Janus/Ada "Site" Licenses
Janus/Ada Source Code Licenses
Janus/Ada Cross Compilers
Janus/Ada Maintenance Agreements

Coming Soon: New Computer and Operating Systems Coverage

Selected Janus/Ada packages are available from the following:

National Distributors

Westico, Inc.
25 Van Zant St.
Norwalk, CT 06855
(203) 853-6880

Soft-Net
5177 Richard, Suite 635
Houston, TX 77056
(713) 933-1828

AOK Computers
816 Easley St., Suite 615
Silver Springs, MD 20910
(310) 588-8446

Trinity Solutions
5340 Thornwood Dr., Suite 102
San Jose, CA 95123
(408) 226-0170

Compuview Products, Inc.
1955 Pauline Blvd., Suite 200
Ann Arbor, MI 48103
(313) 996-1299

International Distributors

Micronix
11 Blackmore St.
Windsor 4030
QLD. Australia
(07) 57 9152

Progesco
155, rue du Fauburg
St. Denis
75010 Paris
(1) 205-39-47

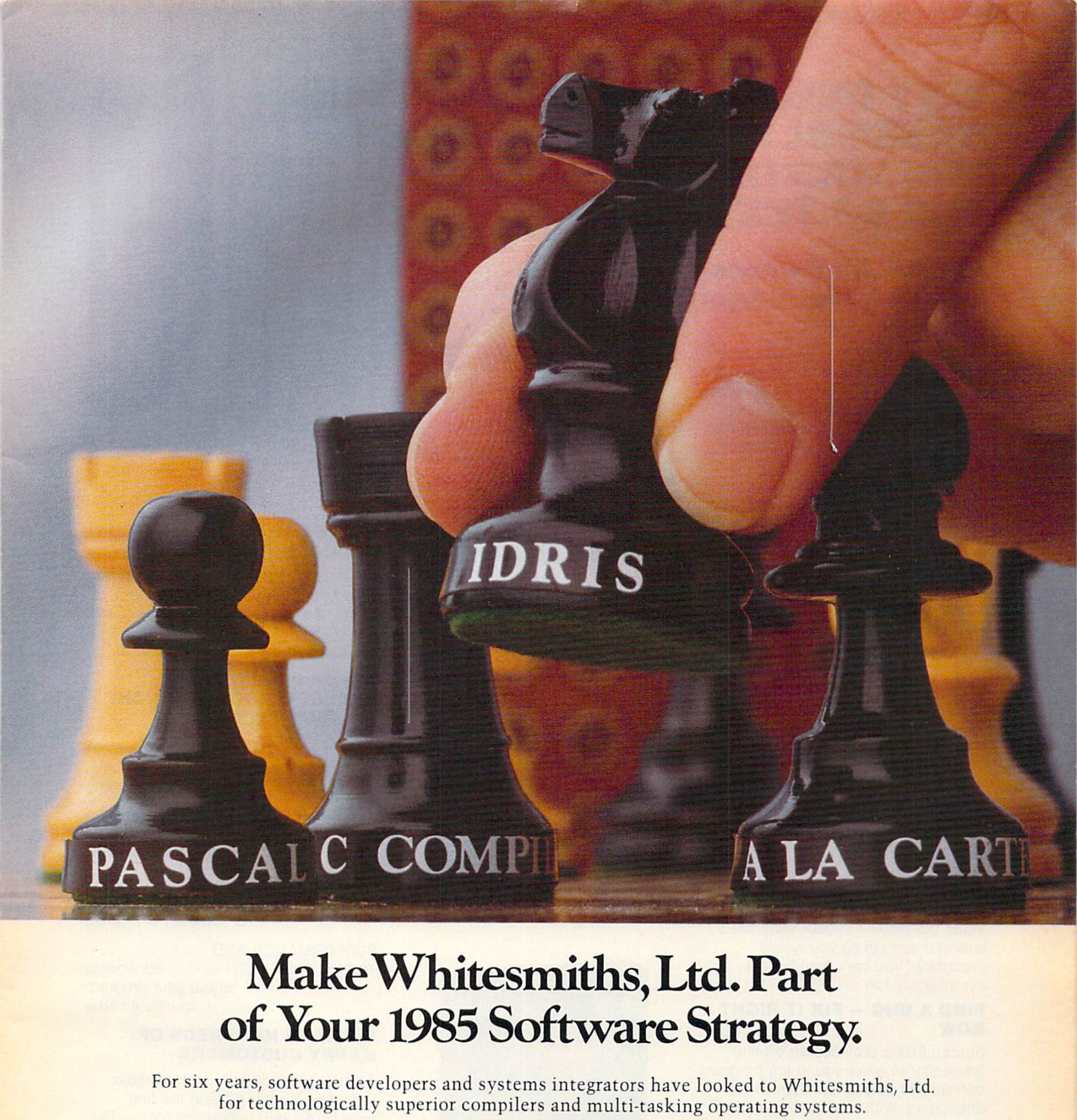
Lifeboat of Japan
S- 13-14, Shiba
Minato-Ku
Tokyo 108 Japan
03-456-4101

SOFTWARE, INC.

specialists in state of the art programming

P.O. Box 1512 Madison, Wisconsin 53701
(608) 244-6436 TELEX 4998168

CIRCLE 58 ON READER SERVICE CARD



Make Whitesmiths, Ltd. Part of Your 1985 Software Strategy.

For six years, software developers and systems integrators have looked to Whitesmiths, Ltd. for technologically superior compilers and multi-tasking operating systems.

Before you make your next move, contact Whitesmiths, Ltd.

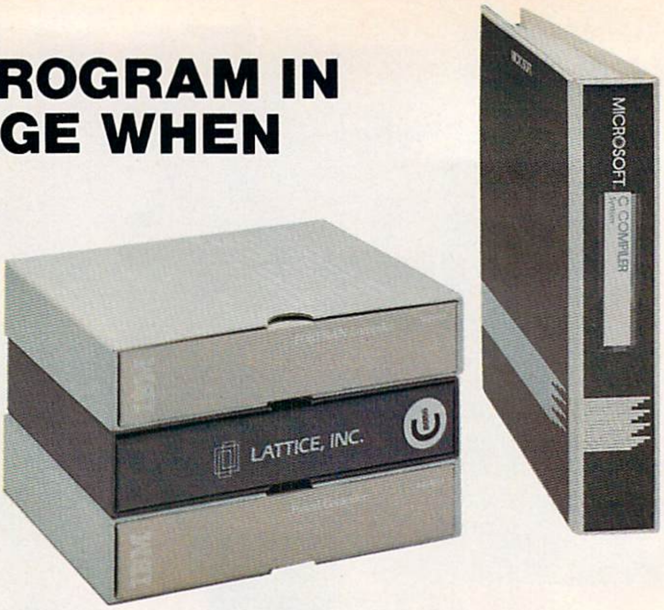
Whitesmiths, Ltd.
97 Lowell Road
Concord, MA 01742
TLX 951708 SOFTWARE CNCM.
(617) 369-8499

Whitesmiths, Ltd.

DISTRIBUTORS: Australia, Fawnray Pty. Ltd., Hurstville, (612) 570-6100; **Japan** Advanced Data Controls Corp., Chiyoda-ku, Tokyo (03) 263-0383; **United Kingdom,** Real Time Systems, Douglas, Isle of Man 0624-26021; **Sweden,** Unisoft A.B., Goteborg, 31-125810.

CIRCLE 85 ON READER SERVICE CARD

WHY DEBUG YOUR PROGRAM IN ASSEMBLY LANGUAGE WHEN YOU WROTE IT IN ONE OF THESE...



ATRON Announces Source Level Software Debugging

Without source level debugging, the programmer must spend time mentally making translations between assembly language and the C, PASCAL, or FORTRAN source code in which the program was written. These tedious translations burn up valuable time which should be spent making critical product schedules. The low level hex and symbolic debuggers available today are superceded by ATRON'S solution — Source Probe.

HOW TO SINGLE STEP YOUR SOURCE CODE AND KEEP CRITICAL DATA IN VIEW

With Source Probe, you can step your program by source code statements. While stepping, a window which you define can display critical high level data structures in your program. The next several source code statements are also displayed to give you a preview of what the program will do

HOW TO DISPLAY DATA IN MEANINGFUL FORMATS

Why look at program data in hex when you defined it to be another data type in your program. Source Probe provides a formated print statement to make the display of your variables look like something you would recognize. You can specify data symbolically too.

FIND A BUG — FIX IT RIGHT NOW

Source Probe provides an on-line text editor to allow you to log program corrections as you find them while debugging. With on-line display and editing of source files, the time lost printing and looking through program listings can be eliminated.

A SNAP SHOT OF REAL TIME PROGRAM EXECUTION — BY SOURCE CODE !

When Source Probe is running on ATRON'S PC PROBE hardware, the real time execution of the program is saved. You can then view your source code as it executed in real time — including all the changes the program made to your data variables.

HOW TO FIND A BUG WHICH OVERWRITES MEMORY

When running on PC PROBE, the Source Probe can trap a bug which overwrites a memory location. Because complex pointers are normally used in high level language programming, this bug occurs frequently and is very difficult to find.

A BULLET PROOF DEBUGGER

What good is a debugger that can be wiped out by an undebugged program? With Source Probe running on PC PROBE, the software is write protected and cannot be changed.

ATRON PROVIDES THE DEBUGGING TOOLS WHICH FIT YOUR PROBLEM

PC PROBE —	A hardware aid to symbolic software debugging
SOFTWARE PROBE —	A symbolic debugger, runs without PC PROBE
SOURCE PROBE —	A source level debugger, versions run with or without PC PROBE
PERFORMANCE AND TIMING ANALYZER —	For finding where your program spends its time

WE HAVE HUNDREDS OF HAPPY CUSTOMERS

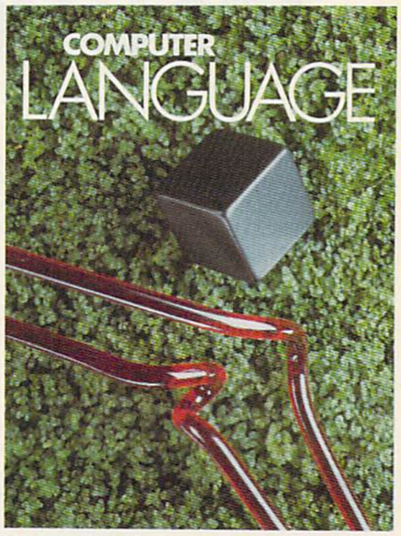
ATRON produced the first symbolic debugger for the PC and the first hardware aided debugging tool — PC PROBE. We have hundreds of happy customers who have made their schedules because of ATRON debugging tools. Why waste more time — call us today!

ATRON
a debugging company

20665 FOURTH STREET • SARATOGA, CA 95070 • (408) 741-5900

CIRCLE 4 ON READER SERVICE CARD

COMPUTER LANGUAGE



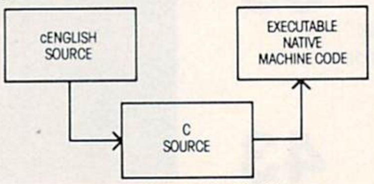
ARTICLES

DEPARTMENTS

c ENGLISH.™ The C Generation Language.

What is cENGLISH? cENGLISH is a comprehensive fourth generation procedural language based on dBASE II™ syntax. It is portable to a wide range of micros and minis. The language features user-transparent interfaces to a wide range of popular C compilers, operating systems, and data base managers.

How is portability achieved? cENGLISH through its compiler interface translates cENGLISH into documented C source and uses a host C compiler to produce native machine code.



C source can be embedded in cENGLISH source.

Differences in the operating system and data base manager are handled by the runtime libraries.

The result is that cENGLISH source can be compiled without modification on any micro or mini configuration supporting cENGLISH.

What about performance? cENGLISH executes FAST, just like any compiled C program.

How easy is cENGLISH to use? While cENGLISH is a powerful high level language that can accommodate complex software development, it remains simple and straightforward to use.

Call or write for availability of cENGLISH for the following configurations—

Compilers:
  Standard O/S compilers: Lattice C™ for MS/DOS™

Operating Systems:
  UNIX,™ UNIX-like, MS/DOS,™ Coherent,™ VMS™

Data Base Managers:
  C-ISAM™ and INFORMIX,™ UNIFY,™ ORACLE,™ PHACT,™ Logix™

Foreign Language Versions:
  German, French, Spanish

Attention MS/DOS users. Demo version and special introductory offer available for IBM PC,™ XT,™ AT,™ and other MS/DOS systems. Requirements: 256K, hard disk or two floppy disk drives, and MS/DOS 2.1 or higher.

Attention dBASE II and dBASE III users. dBASE II to cENGLISH Converter now available; dBASE III Converter available later this quarter. Converted code is portable to micros or minis and executes as fast as original cENGLISH source.

SAMPLE cENGLISH PROGRAM

```
IDENTIFICATIONS
  MODULE:  Mininame
  AUTHOR:  bcs
  DATE:    8/29/84
  REMARKS: Sample cENGLISH program that adds first
           names to a file
END IDENTIFICATIONS

GLOBALS
  FIXED LENGTH 1 ans
  FIXED LENGTH 15 Fname
END GLOBALS

MAIN PROGRAM

  BEGIN
    CLEAR SCREEN
    SET ECHO OFF

    USE "NAMES"
    VIEW BY "ID_FNAME" ASCENDING

    AT 23,1 SAY "Add a record? Y or N"
    AT 23,25 ENTER ans USING "!"

    WHILE ans EQ "Y"
      CLEAR GETS
      AT 6,1 SAY "Enter first name"
      AT 6,20 GET Fname
      READ SCREEN

      INSERT
        Fname = Fname
      END INSERT

      AT 12,10 SAY "Welcome to cENGLISH," & Fname
      WAIT
      AT 14,10 SAY "HIT ANY KEY TO CONTINUE"
      STORE "                    " TO Fname
      STORE "    " TO ans
      AT 23,1 SAY "Add another record? Y or N"
      AT 23,30 ENTER ans USING "!"
      CLEAR ROW 1 THRU 23

    END WHILE

    AT 12,10 SAY "That's all for now!"
    UNUSE "NAMES"
    SET ECHO ON

END PROGRAM
```

cLINE™, Inc.

I'd like to know more about cENGLISH. Please send further information.

Your Name ____________ Title ____________

Company ____________ Telephone ____________

Address ____________

City ____________ State ______ Zip ______

Check one: ☐ End User ☐ System House ☐ Dealer ☐ Distributor

Send to: cLINE Inc., 20 West Ontario, Chicago, IL 60610-3809
Telex 516315 Phone (312) 944-4510

In Canada: cLINE Canada, Inc. Complexe La Laurentienne,
425 St. Amable, Suite 165, Quebec, Canada G1R5E4
Phone (418) 524-4641

A4Q84

CIRCLE 26 ON READER SERVICE CARD

Editor's Notes

Microcomputer programming has changed dramatically since the late 1970s, when magazines like *BYTE, Interface Age, Dr. Dobb's Journal,* and *S-100 Microsystems* were required reading for the informed few.

Even today, those who remember the early days speak nostalgically of the way these magazines acted as a kind of technical underground of valuable information. Back then magazines didn't have to be glossy, just good.

For this month's ComputerVisions interview, Carl Landau, publisher of *COMPUTER LANGUAGE*, flew out to New Jersey and spent some time with one of the more influential microcomputer people in the late 1970s—Sol Libes, editor of the now defunct *Microsystems*.

In this special interview, Carl talks with Sol about his memories of the early days of microcomputing and the beginning of *Microsystems*. He also reveals some of the reasons why this important magazine suddenly collapsed.

We have a deep respect for the role that magazines like *Microsystems* have played in the past. We hope *COMPUTER LANGUAGE* will fill some of the gaps caused by the loss of fine technical publications like *Microsystems*.

Now that the new year has just begun, let me briefly outline some special features we have planned for our 1985 Editorial Calendar.

Next month we present our first theme issue: the C programming language. The issue will feature a special comparative review of the 23 C compilers now being sold on the market. Over a two-month period, a *COMPUTER LANGUAGE* team of five C experts developed and used objective criteria to examine each compiler.

Throughout the year you can expect to see similar product comparisons on BASIC, FORTH, Modula-2, COBOL, and expert systems.

The month of May will be the setting for our BASIC theme issue, again with a product comparison of the over 25 BASIC interpreters and compilers on the market. In July we will feature a special artificial intelligence issue with articles on LISP, Prolog, and expert systems.

Many readers have written to me saying that our coverage of exotic languages like SNOBOL and PILOT has been the most creative and enlightening feature of our new magazine. In August we will devote the entire issue to articles on the many obscure, yet fascinating languages being used by small groups of people around the country.

Many people in our industry claim that the key to good programming is a working knowledge of algorithms. This, said Donald Knuth, is the backbone of good programming style. In November we will focus on algorithms from both a theoretical and mathematical sense.

Finally, to complete the 1985 calendar year, our December issue will focus on compiler writing design and techniques.

Over the past few months, *COMPUTER LANGUAGE* has enjoyed a rapid growth in all aspects—from advertising to circulation to editorial manuscript submissions. By providing you with a well-defined editorial focus and fresh material each month, we look forward to 1985 with a positive and confident attitude. Happy New Year!

[illegible]

Craig LaGrow
Editor

Telecommunicate to COMPUTER LANGUAGE

COMPUTER LANGUAGE has established two bulletin board systems for you to upload and download text and binary programs, as well as to leave your own electronic Letter to the Editor. All the program listings referred to in every issue of the magazine will be available here.

In addition, *COMPUTER LANGUAGE* has its own Special Interest Group on CompuServe's national data base. After calling into your local CompuServe node, simply type "GO CLM" at any prompt and you'll be in our SIG.

To access our bulletin board, set your computer or terminal to the following parameters: 8 data bits, no parity, 1 stop bit, full duplex, and either 300 or 1200 baud. The telephone number is (415) 957-9370. After your modem makes the connection, type RETURN several times, and everything else is easy.

Both systems are open 24 hours per day, 7 days per week. Due to the heavy number of callers, please do not log into the system more than one time per day. Messages left on either system will be combined the following day.

COMPUTER LANGUAGE

EDITOR
Craig LaGrow

MANAGING EDITOR
Regina Starr Ridley

TECHNICAL EDITOR
John Halamka

EDITORIAL ASSISTANT
Hugh Byrne, Lorilee Biernacki

CONTRIBUTING EDITORS
Serg Koren, Pierre Provencher, Tim Parker, Ken Takara

INDUSTRY NEWS CONSULTANT
Bruce Lynch

ADVERTISING SALES
Jan Dente

OPERATIONS CONSULTANT
Beatrice C. Blatteis

CIRCULATION COORDINATOR
Renato Sunico

ART DIRECTOR
Jeanne Schacht

COVER PHOTO
Dow Clement Photography

PRODUCTION ARTIST
Anne Doering

PRODUCTION
Barbara Luck, Steve Campbell, Kyle Houbolt

TECHNICAL CONSULTANT
Addison Sims

MARKETING CONSULTANT
Steve Rank

ACCOUNTING MANAGER
Lauren Kalkstein

PUBLISHER
Carl Landau

COMPUTER LANGUAGE *is published monthly by* COMPUTER LANGUAGE *Publishing Ltd., 131 Townsend St., San Francisco, CA 94107. (415) 957-9353.*

Advertising: For information on ad rates, deadlines, and placement, contact Carl Landau or Jan Dente at (415) 957-9353, or write to: COMPUTER LANGUAGE, *131 Townsend St., San Francisco, CA 94107.*

Editorial: Please address all letters and inquiries to: Craig LaGrow, Editor, COMPUTER LANGUAGE, *131 Townsend St., San Francisco, CA 94107.*

Subscriptions: Contact COMPUTER LANGUAGE, *Subscriptions Dept., 2443 Fillmore St., Suite 346, San Francisco, CA 94115. Single copy price: $2.95. Subscription prices: $24.00 per year (U.S.); $30.00 per year (Canada and Mexico). Subscription prices for outside the U.S., Canada, and Mexico: $36.00 (surface mail), $54.00 (air mail)—U.S. currency only. Please allow six weeks for new subscription service to begin.*

Postal information: Second-class postage rate is pending at San Francisco, CA and additional mailing offices.

Change of address: Please allow six weeks for change of address to take effect. POSTMASTER: Send change of address (Form 3579) to COMPUTER LANGUAGE, *131 Townsend St., San Francisco, CA 94107.*

NEW from BORLAND!

TURBO TOOLBOX & TURBO TUTOR

"TURBO is much better than the Pascal IBM sells."
Jerry Pournelle,
Byte, July 1984

"TURBO PASCAL appears to violate the laws of thermodynamics.
You won't find a comparable price/performance package anywhere. It is simply put, the best software deal to come along in a long time. If you have the slightest interest in Pascal . . . buy it."
Bruce Webster,
Softalk IBM: March 1984

BORLAND INTERNATIONAL GIFT PACK

ONLY **$99.95**

A SAVINGS OF $30!

What a gift for you and your friends! The extraordinary TURBO PASCAL compiler, together with the exciting new TURBO TOOLBOX and new TURBO TUTOR. All 3 manuals with disks for $99.95.

TURBO PASCAL Version 2.0 (reg. $49.95). The now classic program development environment still includes the FREE MICROCALC SPREAD SHEET. Commented source code on disk

• Optional 8087 support available for a small additional charge

NEW! TURBO TOOLBOX (reg. $49.95). A set of three fundamental utilities that work in conjunction with TURBO PASCAL. Includes:

• TURBO-ISAM FILES USING B+ TREES. Commented source code on disk
• QUIKSORT ON DISK. Commented source code on disk
• GINST (General Installation Program)

Provides those programs written in TURBO PASCAL with a terminal installation module just like TURBO'S!

• NOW INCLUDES FREE SAMPLE DATABASE . . . right on the disk! Just compile it, and it's ready to go to work for you. It's a great example of how to use TURBO TOOLBOX and, at the same time, it's a working piece of software you can use right away!

NEW! TURBO TUTOR (reg. $29.95). Teaches step by step how to use the TURBO PASCAL development environment—an ideal introduction for basic programmers. Commented source code for all program examples on disk.

30 DAY MONEY BACK GUARANTEE These offers good through Feb. 1, 1985

For VISA and MASTERCARD order call toll free: **1-(800)-255-8008 1-(800)-742-1133**
(Lines open 24 hrs., 7 days a week) Dealer and Distributor inquiries welcome (408) 438-8400

CHOOSE ONE (please add $5.00 for handling and shipping U.S. orders)

___ All Three-Gift Pack	$ 99.95 + 5.00 **SPECIAL!**	___ Turbo Toolbox	$49.95 + 5.00
___ All Three & 8087	139.95 + 5.00 **SPECIAL!**	___ Turbo Tutor	29.95 + 5.00
___ Turbo Pascal 2.0	49.95 + 5.00	___ Turbo 8087	89.95 + 5.00

Check__________ Money Order__________ VISA__________ MasterCard__________
Card #: ______________________________ Exp. date: ______________ Shipped UPS
My system is: 8 bit ___ 16 bit ___
Operating System: CP/M 80 ___ CP/M 86 ___ MS DOS ___ PC DOS ___
Computer: ______________________ Disk Format: ______________________
Please be sure model number & format are correct.
NAME: ____
ADDRESS: ____
CITY/STATE/ZIP: ____
TELEPHONE: ____

California residents add 6% sales tax. Outside U.S.A. add $15.00 (if outside of U.S.A. payment must be by bank draft payable in the U.S. and in U.S. dollars). Sorry, no C.O.D. or Purchase Orders. 20

BORLAND INTERNATIONAL
4113 Scotts Valley Drive
Scotts Valley, California 95066
TELEX: 172373

CIRCLE 6 ON READER SERVICE CARD

The first compiler for dBASE II®

SPEED

dB Compiler™ produces applications which execute substantially faster than under dBASE II® in 16-bit environments. Some operations are even faster than under dBASE III®!

INDEPENDENCE

Buy dB Compiler™ once and compile and distribute as many applications as necessary with no additional cost. WordTech imposes no licensing fees, and a compiled application will execute without dBASE II or RunTime®.

SECURITY

Compilation is far better than encryption for protecting programming insights and procedures.

PORTABILITY

dB Compiler's™ cross-environment linkers make it easy to generate executable code for several operating systems. For CP/M-80®, CP/M-86®, PC-DOS®, and MS-DOS®.

Suggested retail price: $750; Cross-environment linkers: $350.
Corporate/Multi-user licenses available.

dB COMPILER™

WORDTECH SYSTEMS, INC. P.O. Box 1747 Orinda, CA 94563 (415) 254-0900

CP/M-80, CP/M-86®, DRI PC-DOS®, IBM MS-DOS®, Micro-Soft Corp., dBASE II, RunTime, dBASE III®, Ashton-Tate, Inc.

CIRCLE 41 ON READER SERVICE CARD

FEEDBACK

COBOL author rebuttal

Dear Editor:

I think David Soderberg (Feedback letter in November issue) was upset by my discussion of left vs. right brain in the premier issue's "COBOL: Pride and Prejudice." I must have touched a nerve. He regards left-brain dominant thinking as "technical reason" while dismissing the right side as "strongly opinionated on non-substantive issues appealing to sensation."

Since I'm evidently pro-COBOL, what is his point about "60-70% of applications written is also in the realm of data processing. And *that's* the point which is overlooked . . . " Emphasizing *that* doesn't add meaning. Good writing would have.

Is COBOL functionally different from Pascal and C? I don't think so. They're all general purpose programming languages that can do anything. Try writing UNIX in COBOL. How about writing a compiler in COBOL. How about writing the best compiler written—any language, any machine. One that generates code that will run any benchmark twice as fast as TURBO Pascal, Lattice C or BASCOM. I'm referring to the Realia COBOL compiler. In my article I asked why COBOL compilers are never written in COBOL. I should have said seldom. Realia proves that the best compilers are written in their target language and COBOL is inherently faster than the rest.

The November issue also had a review of mbp COBOL. I feel you owe your readers a description of Realia. Not only does the compiler generate outstanding code, the indexed IO is even better. I've been running benchmarks against a mainframe (2 mips) running VSAM on 3370s (6,000 rpm) through a high-speed channel (1.75 mbs). The PC beats it.

I talked to a developer whose 3,000-line program takes 2.5 hours to compile under mbp. We haven't tried it yet, but Realia will do it in less than 2 minutes. In case anyone is hung up on the mbp screen manager, I wrote an interface allowing you to compile the same program under Realia and call the mbp library routines.

Robert Wagner
Lubbock, Texas

General criticisms

Dear Editor:

I'd like to offer some brief comments on your new magazine.

■ You have a good idea and so far a reasonable start, but I'm not nearly as enthusiastic as some of your letter writers.

■ "Batch—A powerful IBM 'language'", in the October issue, for instance, is overblown. It says nothing about the slowness of the facility or about the need to have a copy of COMMAND.COM in the A drive. I never did find the "greatest secret—The Undocumented Feature." Nothing was said about the limited environment space or the need to clean it up at the end of a file. As an example of a useful Batch file I offer the one presented in Listing 1, called MC for multi-copy. Note that the last line "set to=" has no blank after the equal sign. This removes the temporary variables from the environment.

■ Your Bulletin Board Service is an OK idea if one has a modem. I don't and find less pleasure in being told that all the good stuff is unavailable to me. *Dr. Dobb's* practice of printing code is much preferred by me, at least.

■ The Code Swap Shop, for instance, could include non-modem ways to get code. I wrote to Bruce Tonkin but can't write to Michael O'Quin as you gave no address. (Can you give me an address for him?)

■ I found the piece on Donald Knuth to be pure fluff. There must be some way to avoid Sunday Supplement level articles.

■ October's "MNSNUS (or, Using Mnemonic Atoms in Symbolic Naming)" was good as was "The Evolution of ZCPR" although the latter again suffers the "get it from the BBS" problem.

Samuel Green, Ph.D.
Stoddard, N.H.

Batch author Darryl Rubin responds: Batch's undocumented feature certainly isn't the greatest PC-DOS has to offer, but

```
USAGE:  MC file1 dest.dr file2 file3 . . .

EXAMPLE:  MC a:file1 b:  c:file2 file3

CODE:
                echo off
                set to=%2
                echo copy %1 %2 /v
                copy %1 %2 /v
                shift
                :loop
                shift
                if "%1"=="" goto out
                echo copy %1 %to% /v
                copy %1 %to% /v
                goto loop
                :out
                set to=
```

Listing 1.

I did explain it (page 34): Batch files can refer to environment variables with the %name% construction. It is true that environment space is limited, so you might want to use the SET test I described (page 35) for checking free space. For more speed, try running your batch files from a RAM disk. Va-voom!

Michael O'Quin's address is P.O. Box 4462, Medford, Ore. 97501. —Ed.

FORTRAN 77 musings

Dear Editor:

I appreciated Bruce Hunter's review of DR FORTRAN 77 in the November issue. I haven't used the implementation he refers to but have done some programming in Fortran 77 on a DEC-20 and can attest to his conclusion that much-maligned FORTRAN has grown into a powerful and flexible programming language.

I think people need to hear his point that FORTRAN 77 includes powerful number-crunching facilities and also facilitates the writing of well-structured programs. Unfortunately, the example given in his review demonstrates neither of these virtues.

The number-crunching in his example is thoroughly pedestrian and could be as easily coded in almost any programming language. (Well, maybe not LISP!) And it's not done well, at that. For example, the line

```
k = (4.0 * c – 1)/(4.0 * c – 4) +
      0.613 / c
```

would be more effectively written as

```
k = (c – 0.25) / (c – 1) + 0.613 / c
```

eliminating two floating point multiplications (and their potential for resulting truncation/roundoff errors) and speeding up a statement embedded in a loop that repeats many times.

As for program structures, we find that the main routine is a do loop that runs 65,535 times. Since most algorithms like this are terminated by a specific convergence criterion rather than after an arbitrary number of iterations, we are already puzzled.

Reading on, we learn that in fact there is little or no intention to do the loop 65,535 times. Rather, we will exit the loop and the program in the middle of subroutine calc or subroutine output, and the criterion for exit will be user specification after seeing the results so far. A better structure would use a main-routine loop that terminates when a Boolean variable tells it to.

In fact, since in all cases we want to do at least one iteration, the ideal structure for the program is with a test-at-bottom loop control structure such as Pascal's *repeat-until* or C's *do-while*. To my knowledge, FORTRAN 77 has no analogous control structure. The value of the *quit-or-continue* Boolean variable would be set by a separate subroutine whose sole function is to ask the user whether to continue. The calc and output routines would have nothing to do with this function. If this sounds like nitpicking, think what it would take to modify Hunter's program to display output for each iteration but offer the option of termination only after every 10th cycle!

It worries me that people may look at those listings and conclude that FORTRAN 77 is an inadequate language. An example displaying its full advantages would have been nicer.

Clyde Schechter
New York, N.Y.

Illustration: Anne Doering

Realia COBOL. Numbers speak louder than words.

Compilation Speed (minutes:seconds)

Lines in Program	Realia COBOL	mbp COBOL	Level II COBOL	R-M COBOL	Microsoft COBOL
1,000	:51	8:33	3:42	5:05	5:11
5,000	3:30	48:07	16:58	*	45:26

*Could not successfully compile the program.

Execution Time Ratio (Gibson Mix; calculated S-Profile)

Realia COBOL	mbp COBOL	Level II COBOL	R-M COBOL	Microsoft COBOL
1.0	3.6	14.7	21.6	22.3

Sieve of Eratosthenes

0.818 seconds per iteration

All benchmark tests were performed on an IBM PC-XT with 192KB of memory. IBM PC-XT is a registered trademark of International Business Machines Corporation; mbp COBOL, of mbp Software and System Technology; Level II COBOL, of Micro Focus; R-M COBOL, of Ryan-McFarland; and Microsoft COBOL, of Microsoft.

+ IBM VS COBOL compatibility.

REALIA inc.

$995

10 South Riverside Plaza
Chicago, Illinois 60606
(312) 346-0642

CIRCLE 76 ON READER SERVICE CARD

Pascal points

Dear Editor:

"Pascal: Just a teaching language?" in November's Designers Debate was very interesting. Here are a few related points to consider.

Is Pascal a good teaching language? It has the triad syndrome: three assignment symbols (":=" for values, ":" for types to variables and "=" for types to identifiers) and three kinds of loop constructs. In both cases it requires the students learning, remembering and selecting three items which perform a single basic function. Perhaps the student is being taught who's boss.

We can learn from Pascal one basis for new computer language design: for simplicity and clarity in learning and use, a single construct is to be provided for a single function. In the case of Pascal this would have resulted in a single assignment symbol and a single flexible loop construct.

Starret C. Kennedy
New York, N.Y.

Editorial directions

Dear Editor:

Recent events have caused me to take a heightened interest in the continued success and well-being of your magazine. No sooner had I been accepted for *Microcomputing*'s Software Review Board then I read in *InfoWorld* that *Microcomputing* was folding! I found this distressing, as I have kept every issue (and thoroughly enjoyed most of the earlier ones), and their apparent attempts to rework their format looked promising.

Then I read a couple of days ago in *Dr. Dobb's* that *Microsystems* is giving up the ghost! This is serious. There hasn't been a more hard-core systems programming magazine for microcomputers since Lifeboat Associates' *Lifelines*. It hadn't been exhibiting the usual pre-collapse symptoms (less advertising, thinner, late, etc.) either.

I hope your advertisers and publisher see your magazine as positioned in an unfilled niche in the marketplace rather than a me-too in a dying market segment. I certainly do.

As far as I'm concerned, this leaves you and *Dr. Dobb's* as the only two remaining magazines of real substance at the advanced level. Your editorial directions are sufficiently different that I think you can co-exist without hurting each other.

Charles M. Somerville
Dayton, Ohio

You will probably be interested in our interview with the editor of Microsystems, *Sol Libes, in ComputerVisions on page 23.—Ed.*

OMNI listing available

An illustration of an IRA calculating program written in BASIC and OMNI (the featured language in December's Exotic Language of Month Club) can be downloaded by calling the COMPUTER LANGUAGE *Bulletin Board Service at (415) 957-9370 or CompuServe (type "GO CLM") and reading file OMNI.LTG.—Ed.*

Screen Displays. Fast and Easy!

Blaise Computing presents VIEW MANAGER™—a screen programming system for the IBM personal computer and hardware compatibles. VIEW MANAGER™ speeds the creation, documentation, and incorporation of screens into programs developed in high level languages. Versions for C (Lattice, Microsoft, or Computer Innovations) and Pascal (IBM or Microsoft) are now available.

VIEW MANAGER™ lets you create input/output screens by providing an integrated system of programs carefully constructed to make your screen development easy and fast. It features:

- Quick creation of new screens and editing of existing ones using an interactive painter efficiently storing screens in a screen database;
- Extensive control over the format of data written to and read from data capture fields;
- Ability to create on-line help files as part of the screen system;
- Automatic generation of screen documentation files, including details of formats for data entry fields, screen images, and names and sizes of all existing screens;
- A comprehensive library of routines to include in your programs allowing full manipulation of screens and the data that they display or capture;
- Royalty-free distribution of your commercial or in-house applications developed using VIEW MANAGER™.

All this adds up to a productivity tool no system developer in the C and Pascal environment should be without.

$275 (Source code available for routine library—an additional $150)

VIEW MANAGER™ is part of the Blaise Computing Productivity Series. Other products to speed your development projects in C and Pascal include:

TOOLS™—A library of routines for advanced string handling, forms utilities, screen handling or more $125

TOOLS 2™—A library of routines for access to operating services of DOS 2.0+ from within your program—includes memory allocation, program chaining, file and buffer handling $100

EXEC™—A program chaining dispatcher for all DOS-executable files supporting a common data area $95

BLAISE COMPUTING INC.
2034 Blake Street Berkeley, CA 94704
(415) 540-5441

CIRCLE 11 ON READER SERVICE CARD

WHY WOULD ANY SANE PERSON SPEND $199 FOR A BetterBASIC SYSTEM WHEN DOS's IS FREE?

HERE ARE 10 REASONS:
TEST YOUR SANITY

1. Full support for 640K memory **2.** Structured language with BASIC syntax

3. Separately compiled program modules **4.** Speed: FAST **5.** Extensibility (Make your own BASIC.) **6.** User-defined procedures and functions **7.** Built-in windows support **8.** Interactive programming language based on an incremental compiler

9. 8087 math support **10.** Runs on IBM PC, IBM PC/XT and compatibles

Summit Software Technology, Inc.™
P.O. Box 99, Babson Park
Wellesley, MA 02157

(617) 235-0729

NOW AVAILABLE FOR THE TANDY 2000 & 1200

Better BASIC™

Sane Programmers Order BetterBASIC Now

Price: $199
8087 Math Module: $99
Runtime System: $250
Sample Disk: $10

MasterCard, VISA, P.O. Checks, Money Orders, and C.O.D. accepted.

CIRCLE 83 ON READER SERVICE CARD

By Bruce Lynch

Integrating voice and data communication with computer processing is an important trend. Software is making the use of these capabilities more and more transparent for the end user. All elements—computer, telephone, modem, software, and voice input and output—should appear to the user as a unit. They should be packaged so they all can be used without creating concern over technical details or sequences of operations by the user that are disrupting or frustrating.

Several companies in the last couple of years have attempted to market products—for instance the IBM PC, the Apple, and the DEC Professional 350—that integrate telephones with computers. Recent products have been released for the Macintosh. Costs are now below the threshold needed for widespread adaptation.

However, without proper conceptualization of how an application should flow, marketing efforts will meet with resistance and frustration.

With today's readily available technology it should be practical to approximate an intuitive, unrestrained communication between user and computer through use of a keyboard, screen, and beeper. Add a microphone for the user and the capability could be included to have a speaker play back the message describing telephone call's topic. A widespread market should be practical without the computer understanding how to translate a message from voice recording into data to be played on the screen.

Voice mail systems are becoming fairly popular among very large corporations. They help to do three things:

■ When all that is needed is to convey some quick information to another person, the message can get through even though the target person is not available.

■ When you want a fairly complicated message to be made clear, you no longer have to use an assistant. You record the message for the recipient in the store and forward mode.

■ You also have the capability to branch out of the sequence. You can dial another extension or whatever without hanging up after the tone.

To a person who looks at software as a product category subject to economic forces, it is clear that a lot of software products should be sold at a price based on cost ($8.95 to $50.00). However, other microcomputer software should be sold at a price based on the value it delivers to the user.

Publishers of software above $100 justify their price by pointing to their research, development and marketing costs. Those issues are relevant, but the value delivered to the user is far more important when determining price.

A scenario: imagine a vendor has a magical software package MacTRANS. It will run on the XT. It will translate any commercial software product originally written for the PC or Apple II in any language (BASIC, C, Assembler) to run on the Macintosh. The program produced will fit within 128K RAM and will have full functionality. All of this will take zero programming effort. Of course, such a product isn't really likely to appear.

Should MacTRANS be priced for $30? Of course not—at $30 it might sell 10,000 copies. Remember, we arbitrarily defined this as a product only capable of translating commercial software, not end-user software, silly as that may be. There are definitely companies that would be willing to pay $20,000 for such a product. To balance volume with price, perhaps a price between $250 and $2,000 would be reasonable.

Should the idealist who says software should cost $30 prevent economic forces from encouraging the development of a product that will really only be viable for sophisticated programmer-users? A product that will require customization, be appropriate for only one source language as a starting point, and require a lot of technical support? Such a product would be likely to save six months in lead time to market, a huge amount of uncertainty, and a high cost. Would the natural market for the product really care whether it was $30 or $300?

Should DBMS products TOTAL or ADABASE be sold for $45 a copy to run on IBM mainframes? After all, it only cost $15 or so to manufacture a tape. Maybe $150 would be a better price. That would even cover the cost of documentation and provide a profit on each sale. Isn't it immoral for McCormick & Dodge to charge over $300 for a general ledger module to run on an IBM mainframe?

Certain products should and will quickly become commodities and be sold at commodity (cost plus a markup) prices (like $30). They will become commodity products because so many programmers and so many companies will be able to develop the software, market it, and provide all of the other facilities necessary to build a business around a product concept.

A substantial portion of products currently priced as high as $1,000 in the microcomputer software business are headed toward a $30 retail price. That process will probably take no less than one and a half years and no more than four years for a majority of the product categories that we see today.

Other products—and more are needed—deliver extremely high value to people who require that value and can appreciate it. They deliver value to small markets, markets so small that other companies who are interested in developing software will ignore them. Many of you who are creative will invest substantial effort building such products.

Even while parallel processor work proceeds, performance of single processors continues to progress. "Design-in" with 32-bit micros from National Semiconductor and Motorola will start in the spring with Intel six months behind.

Novics has announced a running prototype for a processor with the following characteristics: 10 mips processing speed (faster than most mainframes), an architecture that supports a very long stack, and an instruction set that supports about 100 Forth words. Delivering such a creature for less than $5,000 as an add-on to a fast bus or even to a PC bus should make possible applications that people normally would not consider.

The impact of the resources available to decision support systems in 1993 will be particularly significant to education and entertainment applications. Even current capabilities are pretty impressive. Ray Bradbury is one of several authors active in the development of science fiction material allowing a reader/user to act as a key participant in the plot and development of a story or game. As processing power and resolution improve and storage cost decreases, this medium is likely to experience tremendous success.

Simulation games will become very popular once the under $1,000 microcomputers have resolutions similar to the Macintosh and sufficient speed.

Low-price software is becoming more common. IBM, from "IBM Direct", is selling 40 products with prices ranging from $10 to $45. In particular, one of its products selling for $10 offers features that seem the same as those in another product by a company spending significant amounts merchandising a product they sell for $100.

Venture capital money for software companies has pretty well dried up. It is likely, in another nine months or more, the venture capital community will start to differentiate between companies doomed by me-too strategies and companies working in safe niches.

About 100,000 programmers with about 120,000 licensed machines work with UNIX now, according to AT&T Technologies.

TopView compatibility will be important to the success of many products fairly quickly.

The VDI graphics standard supported by Graphics Software Systems is catching on well. It got a big boost when IBM agreed to carry its products.

Home computer sales have slipped. A large proportion of those buying such systems are buying them with disk drives, making it easier to distribute commercial software.

ANSI BASIC looks like it has a large following. The market struggle between the de facto Microsoft standard and the ANSI specification will be interesting. There is probably a substantial marketing opportunity there. A large *critical mass* (a term I coined for the relevant factors in a product's marketability—including technical quality of documentation and software, scope of marketing effort, market positioning, money, major account sales efforts, manufacturing and quality control, and a healthy combination of key company employees) is necessary to sell properly to and support hardware OEMs.

Experimentation with robotics has reached a stage that is likely to mushroom. With home products that support reasonable functionality at $300 to $500, the market should become enormous beginning late next year.

The software industry is maturing quickly. Distribution channels for software are likely to see some substantial shifts in the next year. Margins paid to contributors will change markedly. A strong polarization will occur between outlets that sell generic products and other more specialized and consultative outlets.

Technical support after purchase will become a product purchased separately on a widespread basis. Through experimental efforts, IBM has evolved a pricing strategy that seems appropriate to market needs and fulfillment costs. $40 is paid for each problem solved whether it takes one call by the user or many.

Object management for C is moving forward, along with natural language interfaces.

SMALL C FOR IBM-PC $40

Small-C Compiler Version 2.1 for PC-DOS/MS-DOS
Source Code included for Compiler & Library
New 8086 optimizations
Rich I/O & Standard Library

CBUG SOURCE LEVEL DEBUGGER FOR SMALL C $40

Break, Trace, and Change variables all on the source level
Source code included

Datalight
11557 8th Ave. N.E.
Seattle, Washington 98125
(206) 367-1803

ASM or MASM is required with compiler.
Include disk size (160k/320k), and DOS version with order.
VISA & MasterCard accepted. Include card no. & expiration date.
Washington state residents include 7.9% sales tax.
IBM-PC & PC-DOS are trademarks of International Business Machines
MS-DOS is a trademark of Microsoft Corporation.

CIRCLE 19 ON READER SERVICE CARD

WIZARD C

Fast compiles, fast code and great diagnostics make Wizard C unbeatable on MSDOS. Discover the powers of Wizard C:

- ALL UNIX SYSTEM III LANGUAGE FEATURES.
- UP TO A MEGABYTE OF CODE OR DATA.
- SUPPORT FOR 8087 AND 80186.
- FULL LIBRARY SOURCE CODE, OVER 200 FUNCTIONS.
- CROSS-FILE CHECKS OF PARAMETER PASSING.
- USES MSDOS LINK OR PLINK-86.
- CAN CALL OR BE CALLED BY PASCAL ROUTINES.
- IN-LINE ASSEMBLY LANGUAGE.
- 240 PAGE MANUAL WITH INDEX.
- NO LICENSE FEE FOR COMPILED PROGRAMS.

The new standard for C Compilers on MSDOS!

Only $450

WSS

For more information call (617) 641-2379
Wizard Systems Software, Inc.
11 Willow Ct., Arlington, MA 02174
Visa/Mastercard accepted

CIRCLE 86 ON READER SERVICE CARD

Specialized processors and new, creative approaches to search algorithms will make possible the huge applications required to allow people to work the way they would prefer rather than the way that the computer now imposes. Those efforts are proceeding quickly.

If you are intrigued by artificial intelligence you should take the time to at least lightly experiment with an expert system building tool. There are six to eight for use on PCs now that look reasonable. You can learn a lot with a standalone product like EXSYS for less than $300. Others act as supplements to a language (usually PROLOG) or are fairly costly ($2,000 to $20,000).

The continued increase in the number of programmers who have purchased the low-cost implementations of LISP, PROLOG and other tools should yield some pretty impressive experimental products by late spring. People with many years of pragmatic experience are building serious developer tools and end-user products. Many should be available this summer.

An interesting by-product of these efforts is the creation of more and more translators to translate from one language to another in computer and spoken languages. The pattern-matching technologies necessary for artificial intelligence will find fruitful application and translation work.

One very common practice results in use of illegal copies. Company A has six PC compatibles and 12 people who use the machines. Manuals and disks are kept wherever is convenient. The disks in use are not originals. They are copies made so that the original can be kept safe as backup.

It is likely that a user will not know or care to determine if she or he is using the legal copy. The user probably does not know or care how many copies the company has the right to use. It would not be unusual for six machines to have ready access to a software product, while the company has license for only one.

Should the company with six or 60 users pay the same price as the company with only one user? No. If 60 people are getting value, then it is not right to pay for only one copy.

If at all possible, copy protection and administration should not cause users to suffer. Any frustrations or decreases in the value delivered by a software product should be avoided when attempting to decrease the number of illegal copies.

The risk of noncompliance should be practically nonexistent for a company that wants to cooperate. Compliance should be easy.

ADAPSO, an industry trade group, has been of great service to the computer industry for many years. A group of its members have been sharing ideas on copy-protection issues.

Through efforts largely led by Lotus and supported by Microsoft, Ashton-Tate, and many other companies, ADAPSO is in the process of developing a standard copy-protection system. It seems likely to provide an answer addressing the needs of the software publishers and at the same time allowing legal users to avoid frustrations.

The hardware design will become available for public domain distribution from ADAPSO. Contact the group for a transcript of a recent meeting where the issues and alternatives were discussed. The address is: Software Protection Committee, ADAPSO, 1300-PS North 17th St., Arlington, Va. 22209.

It seems likely that production quantities of the same device will not be available until the first quarter of 1986.

Only $95 with FULL SOURCE CODE!

Q/C

"... an incredible learning tool." *Byte*

For only $95, Q/C is a ready-to-use C compiler for CP/M with complete source code. Here's what *BYTE* (May 1984) said: "Q/C ... has a portable library and produces good code quality. If you want to learn compiler construction techniques or modify the standard language, Q/C is the obvious choice."

- Source code for compiler and over 75 library functions.
- Strong support for assembly language and ROMs.
- No license fees for object code.
- Z80 version takes advantage of Z80 instructions.
- Q/C is standard. Good portability to UNIX.

Q/C has casts, typedef, sizeof, structure initialization, and function typing. It is compatible with UNIX Version 7 C, but doesn't support long integers, float, parameterized #defines, or bit fields. Call about our new products: Q/C profiler, Z80 code optimizer, and Z80 assembler and virtual linker, all with full source code!

THE CODE WORKS
5266 Hollister, Suite 224
Santa Barbara, CA 93111
(805) 683-1585

Q/C, CP/M, Z80, and UNIX are trademarks of Quality Computer Systems, Digital Research, Zilog, Inc., and Bell Laboratories respectively.

CIRCLE 42 ON READER SERVICE CARD

LOWER PROGRAMMING MAINTENANCE AND DEVELOPMENT COSTS

{SET:**SCIL**™}

The Source Code Interactive* Librarian *for microcomputers.

- **SCIL** keeps a historical record of all changes made to the library.
- **SCIL** maintains any source code regardless of language, including user documentation and text material.
- **SCIL** allows software engineers to work with source code as they do now, *using any ASCII text editor.*
- **SCIL** saves disk space by storing only the changes made to the program.
- **SCIL** provides a labeling capability for ease of maintaining multiple versions and multiple releases.
- **SCIL** offers unlimited description in the program library directory.
- *High visibility displays* with varied intensity for ease of viewing insertions and deletions.
- **SCIL** is available on CP/M, MP/MII, MS-DOS, PC-DOS and TurboDOS.

{SET} Get {SET} for Success
{SET:**SCIL**™} is a product of System Engineering Tools, Inc.
645 Arroyo Drive, San Diego, CA 92103

Registered Trademarks: CP/M, MP/MII: Digital Research Inc.; MS-DOS: Microsoft Corp.; PC-DOS: IBM Corp.; TurboDOS: Software 2000, Inc.

For more information call (619) 692-9464.

CIRCLE 64 ON READER SERVICE CARD

One particular proposal appears to be gaining acceptance as the members search for a solution. Since the proposed device incorporates its own hardware, we will call it H-Protect for now. Remember, the specification has not yet been finalized.

The design and development of H-Protect should make it largely independent of the operating system and operating environment.

End users will pay a price as low as $25 on a one-time basis for the master hardware. Software publishers would have a couple of options when deciding how to use the device. One is to ship hardware "keys," which should cost somewhere around $3 or $4 to the publisher. The other option is to tie the install program of the software product to the master key installed in the device.

H-Protect will attach to an RS-232 port and should make possible complete use of that port for two-way communication at speeds up to 19.6KB. The device comes with one master key and a variable number of auxiliary slots capable of supporting keys delivered by the software publishers.

Any user with the appropriate key could use the commercial software product protected by the system without going through any special effort. Hard disk, LANs, and multi-user systems all would be scheduled for support—in a manner transparent to the end user, we hope. The user could take a key if he or she wanted to walk down the hall to use a different machine.

H-Protect would include a special microprocessor to generate tables and other algorithms that can be integrated by the commercial software product. Commercial software developers could develop custom keys and custom algorithms to support whatever degree of sophistication they would like to have for their own copy-protection scheme. The design seems likely to make support of very secure algorithms possible.

The computer hardware and software components of a decision support system (DSS) in 1993 are likely to consist of the following:

- A central processor capable of processing at about 10 mips
- A word size of 32 to 64 bits
- 50MB of RAM
- Large screens
- The ability to use fingers or wooden pointers to point to things
- Super high resolution
- Integration of video disk capabilities with computer programs
- High-speed networks and interrelationships among computers
- Voice output
- Voice input with reasonable limitations
- Expert system capabilities
- Natural language capabilities
- The ability to retrieve from huge data bases
- Integration of complex combinations of mathematical calculations with what we think of now as data retrieval systems
- Integration of voice and visual communication devices with computers
- Digitizing and optical scanning . . . and real-time characteristics.

What might such a system cost? A wild guess based on long-term technological trends puts such a system in the range of $25,000 to $50,000 in 1993.

Key executives at companies like Northwest Industries, Boeing, and Esmark have been actively involved in the use of DSS for many years. They understand the capabilities well enough to practically eliminate the difficulty of working with technicians. In spite of this, it isn't practical today for them to get the kind of answers that a DSS will provide.

Assume you could use readily available technology and a total budget for one user

TRSDOS TOOLS

MODELS II, 12, 16

TRS/C C COMPILER
Full K&R with source to the function library. UNIX compatible.............................. $85.00

ZSPF EDITOR
SPF, the choice of most mainframe programmers, is now available for Z80 machines. And it's panel driven so you can customize it!.......................... $75.00

MODEL 100 C COMPILER

Now you can write efficient programs for your TRS-80 model 100 with ease. Or, learn the essentials of C programming while traveling!

C/100 - THE "PORTABLE" C COMPILER

Cassette version......................... $49.00
Disk/Video interface version............ $59.00
Model II version (run on mod II, then download object code to model 100).................... $79.00
Model III version (as above, for Mod III)..... $79.00

Write or call for information on other TRS-80 software.

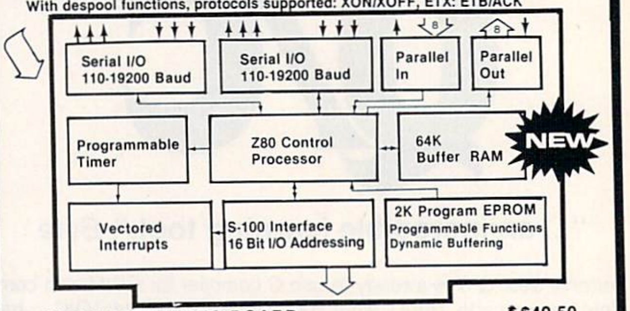
TRSDOS and TRS-80 are trademarks of Tandy Corp.

business utility software
109 minna ste 423 san francisco ca 94105

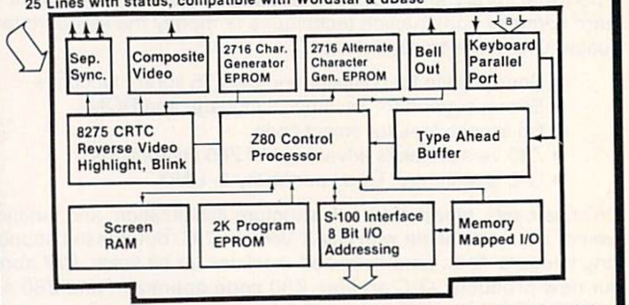
(415) 397-2000

CIRCLE 17 ON READER SERVICE CARD

BUFFERED I/O BOARD — Introductory Price * $59.95
With despool functions, protocols supported: XON/XOFF, ETX: ETB/ACK


Serial I/O 110-19200 Baud
Serial I/O 110-19200 Baud
Parallel In
Parallel Out
8
8
Programmable Timer
Z80 Control Processor
64K Buffer RAM
NEW
Vectored Interrupts
S-100 Interface 16 Bit I/O Addressing
2K Program EPROM Programmable Functions Dynamic Buffering

80 CHARACTER VIDEO BOARD — * $49.50
25 Lines with status, compatible with Wordstar & dBase


Sep. Sync.
Composite Video
2716 Char. Generator EPROM
2716 Alternate Character Gen. EPROM
Bell Out
Keyboard Parallel Port
8
8275 CRTC Reverse Video Highlight, Blink
Z80 Control Processor
Type Ahead Buffer
Screen RAM
2K Program EPROM
S-100 Interface 8 Bit I/O Addressing
Memory Mapped I/O

Includes Bareboard, Heatsink & Documentation Call or write for more information.

SPC

Simpliway Products Co.
P.O. Box 601
Hoffman Estates, IL.60195
(312) 359-7337

VISA

MasterCard

OEM dealer pricing available, $3.00 S/H, IL. Res. add 7% tax
dBase™ - of Ashton - Tate Corp. — Wordstar™ - of Micropro Int'l. Corp

CIRCLE 70 ON READER SERVICE CARD

of no more than $20. Send me your opinions of what would constitute the ideal developer's environment. Try to break up your recommendations into those that are commercially available now, naming the products and the manufacturers, and those that should be readily available now if developers did the work with available technology. Make your recommendations for either of two different types of programmers:

■ A systems programmer with eight years or more of experience who is fluent in at least four different programming languages and three different assemblers.

■ An applications programmer with at least four years of experience with IBM COBOL and Pascal, moderately familiar with assembler and fluent with at least one other high-level programming language.

Here is a flavor of what I am looking for. I have not gone through a thorough analysis or balancing of priorities. I will do this after I get your feedback.

Example: The systems programmer has an integrated development environment where each software component seems to function as if it were all in one system. The components include a source level debugger, a symbolic assembler level debugger, an editor with optional syntax support, a version of a compiler that turns around very fast or an interpreter, a compiler that produces extremely tight code and gives access to all hardware characteristics including registers and ports, an execution profiler, so much RAM that nothing needs to be on disk, numerous utility programs for analyzing and massaging source code, and specialized utilities for supporting link management and library management.

The hardware includes a processor supporting three large compile jobs simultaneously while still providing decent throughput for an edit session to go on. A separate computer is available on a different terminal by swinging one's chair around. Both computers share storage on the same disk. Special support for a circuit emulation is included. What else would be appropriate?

What products are good but need some substantial additional capabilities to be what you see is needed? Let me know your complaints about products that you think are pretty decent. For this purpose, at least, I am not concerned with specific bugs, etc.

Working together we can motivate vendors to make improvements they might otherwise ignore. We can also induce more companies to cooperate to make something that will serve the needs of programmers more effectively than what independent efforts would yield. ■

About ZCPR3 and Z3-DOT-COM

Flexibility is the key to ZCPR3 power. Productivity results from optimum organization of operator and machine resources. You are free to create a thinking and working environment that you choose!

ZCPR3 permits quick computer system re-organization for varying tasks, controls your application programs from integrated, easily produced and changed menus. Generation of aliases permit many commands and keystrokes to be converted to a few. Aliases may be used from within menus. One alias may use another. Control is near absolute using supplied utilities. Menu generation determines how computer is used, simplifying and speeding operations. Single from-menu keystrokes activate complex series of commands.

Shells, multiple commands per line, named directories, file search paths, if-then-goto conditional processing, screen oriented utilities — all major features.

Utilities provided permit file and disk management, easy coordination of many application programs from chained menus with full security and password protection. Online and built-in HELP assists understanding details of each command. ZCPR3, the definitive 8-bit CPM-80 compatible operating system, is a hard worker — one you use, learn from, grow and live with!

ZCPR3 is available in two versions: 1) manual-install system using CP/M MOVCPM, SYSGEN, DDT, and MAC; and 2) auto-install Z3-Dot-COM version. Z3-Dot-Com installation procedure is detailed in eleven (11) lines at the bottom of a Command Reference card. The manual-install version is for two-year-or-over computer users and programmers; the auto-install is ideal for CP/M beginners.

1. **Z3-DOT-COM**
   Auto-Install, load and go, complete full-up ready-to-run system on 4 disks $149.00
2. **ZCPR3 Core and Utilties**
   Manual-Install, source to everything on 10 disks with installation procedure $128.00
3. ***ZCPR3: The Manual***
   Lavish, typeset, over 300 pages $19.95
4. **DISCAT**
   Fancy menu-driven disk catalog system $49.00

A fortnighter newsletter, 24-hour BBS and RCP/M System keep Z3 users informed of microcomputer happenings. Order now! State CP/M disk format desired; add $3.00 shipping & handling; Californians please add 6.5% sales tax. Visa/MC, check, money or purchase order accepted.

(Trademark: CP/M, Digital Research)

Echelon, Inc.

101 First Street • Los Altos, California 94022 • 415/948-3820

CIRCLE 43 ON READER SERVICE CARD

Six Times Faster!

Super Fast Z80 Assembly Language Development Package

Z80ASM

- Complete Zilog Mnemonic set
- Full Macro facility
- Plain English error messages
- One or two pass operation
- Over 6000 lines/minute
- Supports nested INCLUDE files
- Allows external bytes, words, and expressions (EXT1 * EXT2)
- Labels significant to 16 characters even on externals (SLR Format Only)
- Integral cross-reference
- Upper/lower case optionally significant
- Conditional assembly
- Assemble code for execution at another address (PHASE & DEPHASE)
- Generates COM, HEX, or REL files
- COM files may start at other than 100H
- REL files may be in Microsoft format or SLR format
- Separate PROG, DATA & COMMON address spaces
- Accepts symbol definitions from the console
- Flexible listing facility includes TIME and DATE in listing (CP/M Plus Only)

SLRNK

- Links any combination of SLR format and Microsoft format REL files
- One or two pass operation allows output files up to 64K
- Generates HEX or COM files
- User may specify PROG, DATA, and COMMON loading addresses
- COM may start at other than 100H
- HEX files do not fill empty address space.
- Generate inter-module cross-reference and load map
- Save symbol table to disk in REL format for use in overlay generation
- Declare entry points from console
- The FASTEST Microsoft Compatible Linker available

SPEED! SPEED! SPEED!

- Complete Package Includes: Z80ASM, SLRNK, SLRIB - Librarian and Manual for just $199.99. Manual only, $30.
- Most formats available for Z80 CP/M, CDOS, & TURBODOS
- Terms: add $3 shipping US, others $7. PA add 6% sales tax

CIRCLE 59 ON READER SERVICE CARD

For more information or to order, call:

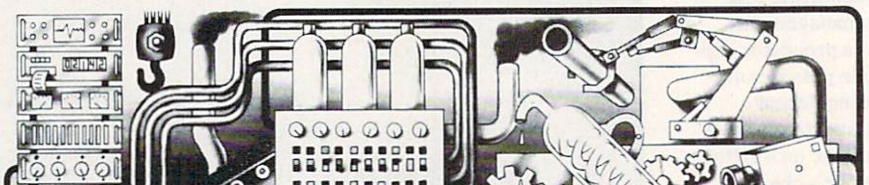
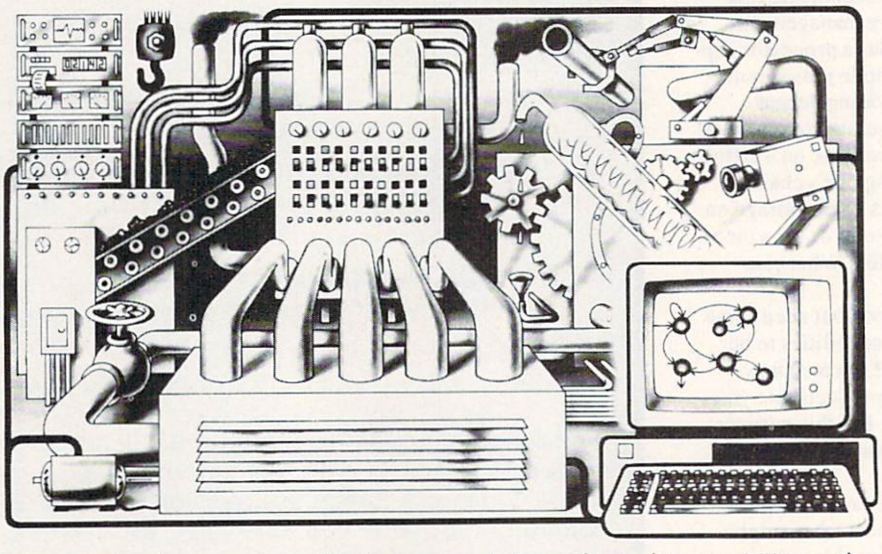
1-800-833-3061

In PA, (412) 282-0864

Or write: SLR SYSTEMS
1622 North Main Street, Butler, Pennsylvania 16001

SLR Systems

Hard problems? **Csharp** can help!

Cut your development time with C source code for realtime data acquisition and control. The **Csharp Realtime Toolkit** includes: graphics, event handling, scheduling, and state systems. Processor, device, and operating system independent.

SYSTEMS GUILD

Systems Guild, Inc., P.O. Box 1085, Cambridge, MA 02142
(617) 451-8479

CIRCLE 84 ON READER SERVICE CARD

FORTRAN vs. FORTRAN 8x

By Ken Takara

This month we go to the mainframe world with FORTRAN as our subject. In particular, we are concerned with a language known as FORTRAN 8x. FORTRAN has a long history as one of the earliest high-level languages, first appearing in the mid-1950s. An acronym for FORmula TRANslation, it has been the mainstay language for mathematical and engineering programmers.

FORTRAN has appeared in various incarnations, beginning with IBM's FORTRAN I in 1956, including FORTRAN II and FORTRAN IV. So far FORTRAN has been through two ANSI revisions, FORTRAN 66 (known as X3.9-1966) and FORTRAN 77 (X3.9-1978). The third revision, FORTRAN 8x, is still in process. Perhaps, following the pattern, it may become FORTRAN 88.

During the course of the committee meetings and public forums, many people have contributed to the 8x effort. The following individuals are represented in this column: Jeanne Martin, committee member and secretary; James Matheny, FORTRAN pioneer; Jeanne Adams, X3J3 chairperson; Rick Lutowski, commercial FORTRAN software developer; Dr. Loren Meissner, former X3J3 member and editor of the ACM SIGPLAN FORTRAN newsletter; Harvey Lynch, ex-FORTRAN programmer; and Bob Upshaw, FORTRAN observer.

Considering the advanced age of FORTRAN, some people wonder why it is still around at all and, even more, whether attempts to revitalize it by adding modern programming constructs are of any value. Will FORTRAN survive to the year 2000?

Upshaw: The one thing FORTRAN has going for it is its name. Sure, it will be around for a long time, at least in name. Other than that, I doubt it will be recognizable to contemporary FORTRAN programmers.

Lynch: FORTRAN is such an archaic language. It's a good 30 years old and totally obsolete. In modernizing FORTRAN, it will be necessary to keep the older features so that earlier code, which is quite abundant, will still be usable. The result will be a large, unwieldy language. One does not require new compilers for old code.

Lutowski: Incidentally, it's not just a mainframe language. It's found on micros now also. I use FORTRAN in my product line for several reasons. First and foremost is portability. In the microcomputer industry, a hardware system has a technological lifespan of about two to four years. In such an environment, it is essential for a commercial software company to write its products in a language that is guaranteed to be portable across hardware and across time. New and better is not portable; old but consistent is portable.

Martin: The FORTRAN standards committee is engaged in an experiment, the modernization of an existing programming language. The committee is aware of the burden of existing code and the absolute necessity of not abruptly invalidating any of it.

Matheny: With a language you want not only software portability; you also want programmer portability. If you change programmers or change jobs, you shouldn't have to start with a totally new language. For this reason, FORTRAN will be around a long time.

Alan Clarke in Britain has said that FORTRAN 8x will bridge the gap between FORTRAN 77 and other modern languages while retaining FORTRAN's unique advantages. As it stands now, 8x contains modern control structures, modular data and procedure definitions, abstract data types and data structures.

It will retain its unique advantages. It is the language of choice for large-scale scientific computing. This capability is enhanced by array processing and new precision facilities. FORTRAN 8x also retains its powerful and flexible I/O features. We hope and intend that the language will continue to facilitate the creation of efficient running programs.

Lynch: There is a lot of code lying around written in FORTRAN. Obviously, you can't ignore the considerable investment. X3J3 recognizes the need for modern features such as data structures and the like but they're also trying to be consistent with this rubbish of 1954.

Rather than fool with all this, they ought to just freeze it as FORTRAN 77. Break with the past. Leave that archaic code alone with the existing compilers. Then they could spend their time working on inter-language linkages. You could write programs using newer languages but still have access to all those FORTRAN subroutines.

Lutowski: Standards ought to be permanent. They shouldn't change. FORTRAN 77, with its features, has made both FORTRAN 66 code and programmers obsolete. I would suggest returning to FORTRAN 66 with the addition of an *INCLUDE* directive for accessing libraries, and a couple of added data types, with no additional foreign syntax. This is a radical suggestion, though, and I'm certain it wouldn't even be considered.

Adams: It is true that FORTRAN 66 is not upwardly compatible with FORTRAN 77 in the case of Hollerith. However, an appendix in FORTRAN 77 completely defines Hollerith for compilers that choose to implement this facility as an extension to FORTRAN 77.

This action has already been taken, though, and the current committee must work with the current standard, FORTRAN 77, in which Hollerith has been replaced with the character data type.

Lynch: A more dangerous incompatibility between FORTRAN 66 and 77 is the difference in the DO loop protocols, which cannot be flagged by the compiler. In FORTRAN 66, a DO loop is always executed at least once. In FORTRAN 77, the loop is executed only if the conditions are satisfied on entry. Sadly, there are some bits of code which assume that the FORTRAN 77 DO loop acts in the same manner as in FORTRAN 66. Such programs could deliver wrong answers with no diagnostic messages.

There seems to be a lot of discussion about the idea of deprecated features. What are deprecated features? And what does that mean to FORTRAN users?

Matheny: We say, FORTRAN will

live, not necessarily forever, but for the forseeable future. Deprecated means "a mild disapproval." We say that we think this feature is not good. We think that the next committee 11 years from now will take it out. We can't tell this next committee what to do. We can only say we think these things aren't good. If you're wise, you won't use it. We've provided something we think is better.

Deprecated features should be around 15 years; it takes a while for processors to come out after the standard is released. We are required by the users that FORTRAN 77 standard conforming programs continue into the future, until the next committee, having given 15 years notice, has the opportunity to take away as they choose.

We feel, though, that we have provided more attractive replacements for those features marked as deprecated. Because of pressure from the user community, X3J3 has decided to add new features on top of old language constructs. This way, the only trap is the use of implementor-defined extensions to FORTRAN 77 (and, of course, to FORTRAN 66).

Lynch: Once the deprecated features are really not supported, no existing program of any worthwhile magnitude will run. Mixing new FORTRAN with old FORTRAN will actually make conversion to the new form more difficult.

Meissner: I like the definition of deprecated I heard; you not only don't like it but you sort of sorrowfully don't like it. Everybody has taken FORTRAN 77 and extended it with all sorts of wonderful new features. Along comes X3J3 with all these beautiful, bold ideas and that's "Futuretran." That's where we're trying to get. The question is, how can we get there from here?

The answer that X3J3 came up with is to take everything you want and concatenate it to everything you've got and you build this huge compiler. You live with this huge compiler until the 1990s. It has to handle all of FORTRAN 77 plus everything you want in that compiler of the year 2001.

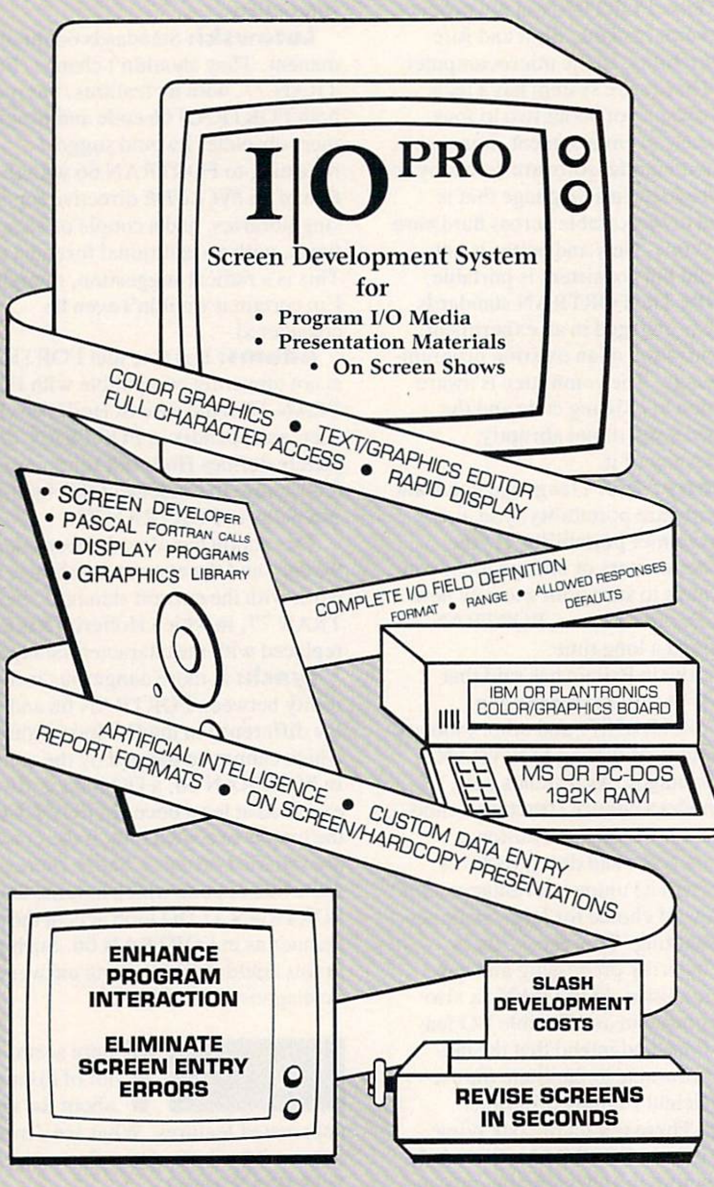
Then in the year 2000 these deprecated features fly away, leaving you with Futuretran. You get there, then, at the expense of having an unnecessarily large language for at least one revision cycle.

Lutowski: The X3 directive that requires language features to be marked "user discouraged" at least one full standard before removal is a step in the right direction. X3 should carry this policy to its logical conclusion by directing that no improvements to the methodology, semantics, or syntax of a language be made once the initial standard has been released. Advances in software technology should form the basis for new languages, rather than revisions in the old ones. Beyond developing the initial standard, the role of the language committees should be to correct errors and clarify the intent of the original standard, not to periodically redefine it. Standards must not be redefined, no matter how obsolete they become.

Sure, FORTRAN has its weaknesses. It has been denounced as unable to support maintainable code due to a lack of structure. Quite often, though, I've seen code full of these nested *IF* statements, where the programmer goes, *INDENT, INDENT, INDENT.* Pretty soon, you don't know where he is.

Many people feel that FORTRAN's greatest asset is its simplicity. As a somewhat primitive language, it has very few features to baffle a new programmer. It provides sufficient abstraction to permit one to work with mathematical formulas easily, yet allows one to work at a level close to the machine. Some people, in fact, consider it a high-level assembler.

I/O A BORE? NOT ANY MORE!


I|O PRO
Screen Development System
for
• Program I/O Media
• Presentation Materials
• On Screen Shows
COLOR GRAPHICS • TEXT/GRAPHICS EDITOR
FULL CHARACTER ACCESS • RAPID DISPLAY
• SCREEN DEVELOPER
• PASCAL FORTRAN CALLS
• DISPLAY PROGRAMS
• GRAPHICS LIBRARY
COMPLETE I/O FIELD DEFINITION
FORMAT • RANGE • ALLOWED RESPONSES
DEFAULTS
IBM OR PLANTRONICS COLOR/GRAPHICS BOARD
MS OR PC-DOS 192K RAM
ARTIFICIAL INTELLIGENCE • CUSTOM DATA ENTRY
REPORT FORMATS • ON SCREEN/HARDCOPY PRESENTATIONS
ENHANCE PROGRAM INTERACTION
ELIMINATE SCREEN ENTRY ERRORS
SLASH DEVELOPMENT COSTS
REVISE SCREENS IN SECONDS

M|E|F Environmental, Inc.
P.O. Box 26537
Austin, TX 78755 (512) 251-5543

$450 : OEM Pricing and Licenses Available
Demonstration Diskette $10
applicable to purchase

CIRCLE 55 ON READER SERVICE CARD

Martin: Although FORTRAN 8x has some features that are similar to features in PL/I and Ada, it is still simpler than either of those languages. There were very good reasons why those features were put into Ada and PL/I. The fact that they exist in other languages does not make them readily accessible to the FORTRAN programmer who has many thousands of lines of FORTRAN that would be difficult to convert to any other language.

Matheny: Simplicity is, of course, a desirable feature in anything. Because we must retain all of the deprecated features of FORTRAN 77, FORTRAN 8x will not be simple. We believe that the language without the deprecated features is concise, consistent, complete and, if you will, simple.

Lynch: I don't like the idea of waiting ten years for another language. Even if 8x is adopted by 1987, it's going to be 1990 by the time anyone gets around to putting out a compiler for it. That's one of Ada's problems. It's so big, that writing a compiler for it is a tremendous task. So, by the time FORTRAN 8x is available, it will be superceded by the next revision.

Lutowski: Look at some of the things being deprecated, *COMMON* blocks and conditional *GOTOs*. The *GOTO* has always been on the blacklist of the structured-code people. Of course, you lose the three-branch *IF*. A lot of the new school of thought is all structured code and strong typing. They don't like *COMMON* because of the coupling problems. That's because people never learned how to use it properly.

If you think about it, there are two ways to transfer data to a subroutine, either via global variables (which are the same as *COMMON* blocks) or argument lists. If you get rid of globals, all you have left are argument lists. If you are passing a lot of parameters, it gets pretty unreal. From the standpoint of execution, it becomes pretty inefficient since these things have to be pushed onto a stack or to some global location. If you have 50 parameters in the list, you end up doing 50 pushes and 50 pops, and that's a lot more overhead. Using *COMMON*, you avoid all that overhead.

Adams: Your concern about your own software is a problem that we recognize as one that is generally felt. However, from its beginnings in 1966 certain features have fallen into disuse among the user community or have become less effective than a newer one. Some of these features are the *PAUSE* statement, and the *ASSIGN* and *ASSIGNED GOTO* statements (the unconditional *GOTO* is retained). These features are quite generally agreed to be obsolete.

Lutowski: I seriously believe that there are no computer scientists today who know how to use *COMMON*, as they all concluded a long time ago that it was bad and have avoided it ever since. I use it heavily within my graphic package, where huge lists are being passed around. But its all hidden from the person using it, since he or she only has to declare the block once, and all access to it is indirect via graphics subroutines. If we had an *INCLUDE* statement, it would become even more transparent.

Computer scientists are always trying to educate the neophyte. They penalize the experienced programmer by trying to protect the beginner. I think that as a result of this, FORTRAN will become much less desirable as a production language from execution and flexibility standpoints. I'd rather have things like *COMMON* there for use by the experienced, self-disciplined programmer rather than to have to declare these argument lists through 10 levels of subroutines.

Martin: *COMMON* is based on the concept of storage association. This has many unfortunate characteristics. It allows for inadvertently making variables equivalent, which is inherently unsafe. It requires that an integer variable and a real variable each occupy one numeric storage unit while a complex variable occupies two numeric storage units. This is unnecessarily restrictive and prevents the true portability numerical analysts have requested.

Heterogeneous data structures, which were universally requested by users, could not be global entities since, if they contained both character and numeric variables, they could not appear in a *COMMON* block. By latching onto Ada's *PACKAGE* concept, we were able to solve problems there. The utility of *COMMON* is provided by the *MODULE* and *USE* constructs, and *USE* provides the functionality of *INCLUDE*.

Lutowski: That restriction is only found in FORTRAN 77; it did not exist in FORTRAN 66. In fact, some implementations of FORTRAN 77 allow both character and numeric variables to exist in *COMMON* together.

There is a dichotomy between the original nature of FORTRAN and the attitude of X3J3. FORTRAN initially allowed the programmer to work at a level very close to the machine. It is now being changed to correspond to the notion of the machine as an abstraction. The deprecation of *COMMON* implies that storage association is bad. This is true only if you want to be kept far from the computer.

Matheny: These things (*COMMON* blocks, etc.) make FORTRAN very powerful but also very dangerous to the user. More formal control and data structures let the user say what he or she means in a way that some other reader of the program can understand.

FORTRAN isn't competing with C,

(*Continued on page 77*)

F77L, THE FORTRAN THAT CHALLENGES THE BIG SOFTWARE COMPANIES.

When you have one software product to sell, you had better make it count. F77L, our complete implementation of the ANSI FORTRAN 77 Standard for the IBM PC, is for programmers who buy a language system based on features and performance.

By specializing in FORTRAN, we see ourselves as different from our competition. Rather than be jacks of all languages, LCS prefers to be experts in FORTRAN. Our competitors, on the other hand, tend to be large and ferocious software firms that offer computer users an assortment of products. At LCS, we specialize in FORTRAN. We have been successfully implementing FORTRAN language systems for over 15 years. We may not have the name recognition or the advertising budgets of the big firms, but we do have what matters most to you—a great product.

Here are a few of the many reasons to buy F77L:

- *Full* FORTRAN 77 Language with popular extensions.
- Compile time: multiples faster than any Goliaths'. (100's statements/min.)
- User oriented interface.
- Numerous, specific English level diagnostics displayed during compilation.
- Command Line compiler options.
- Execution error traceback: program unit line number.
- Selective protection for constants, bounds, interfaces.
- Standard or Free Format source files.
- Lattice C compatibility.
- Easy to follow manual includes appendices on interfaces to Lattice C and Assembly language.
- 30-day-money-back guarantee and ongoing user support.

If you're tired of betting on the software Goliaths and losing, call LCS, the FORTRAN specialist.

$477 for complete package: one 5¼" floppy and manual. Visa/MC Multiple copy discounts. *Requires: 256K/8087.

To order or for more information:

Lahey Computer Systems, Inc.
904 Silver Spur Road, Suite 417
Rolling Hills Estates, CA 90274
213/541-1200

Serving the FORTRAN community since 1969

CIRCLE 53 ON READER SERVICE CARD

MULTI-BASIC

The Compatible BASIC Compiler from Alcor

(Supported Features Chart)

	MBASIC	MBASIC Compiler	CBASIC	CBASIC Compiler	Multi-Basic
IF ELSE/FOR NEXT	●	●	●	●	●
WHILE WEND	●	●	●	●	●
OPEN/CLOSE	●	●	●	●	●
CREATE/DELETE			●	●	●
PRINT/PRINT USING	●	●	●	●	●
LPRINT/LPRINT USING	●	●			●
PRINT @	●	●			●
INPUT/LINE INPUT	●	●	●	●	●
INPUT #/LINE INPUT #	●	●			●
READ #/READ # LINE			●	●	●
LOC/LOF/EOF/ERROR	●				●
FIELD/GET/PUT	●	●			●
RSET/LSET/ERASE	●	●			●
MKD$/MKI$/MKS$	●	●			●
CVI/CVS/CVD/SPC	●	●			●
READ/DATA/RESTORE	●	●	●	●	●
TRON/TROFF	●				●
DEF FN/RANDOM/RND	●	●	●	●	●
DEF USR/SWAP/WAIT	●	●			●
CALL (ASSEMBLY LANG)	●				●
DEFSTR/DBL/SNG/INT	●	●			●
DOUBLE/REAL			●	●	●
INTEGER/STRING			●	●	●
GOTO/GOSUB	●	●	●	●	●
ON ERROR GOTO	●			●	●
RESUME/RESUME NEXT	●				●
ERL/ERR	●			●	●
ON number GOTO/GOSUB	●	●	●	●	●
NAME/RENAME	●	●	●	●	●
PEEK/POKE/INP/OUT	●	●	●	●	●
SYSTEM/SOUND	●				●
SADD/MATCH/UCASE$			●	●	●
VAL/TAB/STR$/VARPTR	●	●	●	●	●
SIN/COS/TAN/ATN	●	●	●	●	●
LOG/EXP/ABS/SQR	●	●	●	●	●
COMMANDS/IF END			●	●	●
INKEY$/INPUT$	●	●			●
TIME$/DATE$/HEX$/OCT$	●	●			●
STRING$/SPACE$	●	●			●
LEFT$/RIGHT$/MID$	●	●	●	●	●
CHR$/ASC/LEN/SGN	●	●	●	●	●
OPTION BASE	●				●
ROW/POS/LPOS	●				●
DIM/MEM/FRE	●	●	●	●	●
MOD/MFRE				●	●
GET/PUT (CHARACTER)			●	●	●
KILL/CLS/CLEAR/INSTR	●	●			●
INT/FIX/CINT	●	●			●
CSNG/CDBL	●	●			●
CHAIN	●		●	●	●
CONSTAT%/CONCHAR%			●	●	●
CONSOLE/LPRINTER			●	●	●
BINARY RANDOM FILES	●	●			●
ASCII RANDOM FILES			●	●	●
255 CHARACTER NAMES				●	●
REDIMENSIONED ARRAYS	●		●	●	●
UNLIMITED STRING SIZE				●	●
MULTI-LINE FUNCTIONS			●	●	●
PROCEDURES/RECURSION					●
FUNCTION TRACING					●
LOCAL VARIABLES			●	●	●
NESTED FUNCTIONS					●
OPTIONAL LINE NUMBERS			●	●	●
DESCRIPTIVE LABELS			●	●	●
SINGLE PRECISION	●	●			●
DOUBLE PRECISION	●	●	●	●	●
LINK TO PASCAL & C					●

For TRS80 models I, II, III, 4, 12, or 2000 using TRSDOS, CP/M, or MSDOS and for IBM PC using PCDOS
Multi-Basic, Pascal, or C $139
Add shipping $6.00 USA, $28 Overseas

ALCOR Systems
13534 Preston Road, Suite 365
Dallas, Texas 75240
(214) 238-8554

Multi-Basic is a trademark of Alcor Systems
TRSDOS, TRS80 is a registered trademark of Tandy Corporation
CP/M, CBASIC are trademarks of Digital Research
MSDOS, MBASIC are trademarks of Microsoft

CIRCLE 1 ON READER SERVICE CARD

YOU DON'T NEED EUREKA!™ ??

CONGRATULATIONS!!

We admire your talents. After all, few people can remember where to find that six month old letter to Wonder Waffle Works, or which of the twenty versions of IMPORTNT.BAS is the one you need yesterday.

Or maybe we should envy your spare time. Ah, to be able to haul out a stack of disks, slip each one into a drive, browse through the directory, and TYPE the various prospects to find that one file or program.

Or perhaps you're the adventurous type who thrills to the challenge of groping through scantily labeled disks, cheering that magical moment when hidden treasures are uncovered.

On the other hand, it occurs to us that you just may not know the advantages of EUREKA!, the fast, menu driven disk cataloger for CP/M. EUREKA! puts your entire disk library at your fingertips. Files may be found quickly and easily - by name or by comments you can put in the file itself. Of course the manual includes a tutorial to help you get started.

Still only $50. Ask your dealer, or contact:

MENDOCINO SOFTWARE COMPANY, INC.

Dept. L-1
P.O. Box 1564
Willits, CA 95490
Phone: (707) 459-9130

Add $2.50 Shipping;
Calif. residents add $3.00 sales tax.

We accept VISA
& Mastercharge

CP/M is a registered trademark of Digital Research Corp.
EUREKA! is a trademark of Mendocino Software Co., Inc.

CIRCLE 48 ON READER SERVICE CARD

C UTILITY LIBRARY

The **C UTILITY LIBRARY** is a set of **200+** functions designed specifically for the PC software developer. Use of the Library will speed up your development efforts and improve the quality of your work.

- **BEST SCREEN HANDLING AVAILABLE**
- **WINDOW MANAGEMENT, COLOR GRAPHICS**
- **DOS 2 DIRECTORIES, COMMUNICATIONS**
- **KEYBOARD, PRINTER, TIME/DATE**
- **EXECUTE PROGRAMS, BATCH FILES**
- **STRINGS, BIOS, AND MUCH MORE**
- **ALL SOURCE INCLUDED—NO ROYALTIES**

Available for Microsoft/Lattice $149, Computer Innovations $149, Mark Williams $149, DeSmet $99. Add $3 shipping. N.J. residents add 6% sales tax. Visa, MC, checks—10 days to clear.

Order direct or through your dealer. Dealer/Distributor inquiries welcome.

ESSENTIAL SOFTWARE, INC.
(914) 762-6605
P.O. Box 1003
Maplewood, N.J. 07040

CIRCLE 28 ON READER SERVICE CARD

Sol Libes—A man without a magazine

By Carl B. Landau

Sol Libes reminded me of Woody Allen. His dry wit, nasal voice, New York background, and pessimistic outlook brought to mind the characters Woody has portrayed in his movies.

Even the setting for the interview would have met with Woody's approval. We had lunch at the Ecko Queen Diner off Route 22 in northern New Jersey. I knew this must be an authentic diner because the waitresses had beehive hairdos and asked, "what can I get for you, honey?"

And like Woody, you can't help but like Sol Libes.

Meeting with Libes was particularly interesting at this point in time because *Microsystems*, one of the first technical microcomputer publications, had just stopped publishing with its November issue. Libes was the founder of *Microsystems* and its editor for five years.

Libes was born and raised mostly in the New York metropolitan area and partially in Los Angeles, Calif., in the 1930s "when it was mostly orange groves." As he puts it, he's "a graduate of publicly-funded universities." He received his undergraduate degree in economics from City Univ. of New York. His masters degree in technical education was earned from Rutgers Univ. in New Jersey.

People have been mispronouncing Libes's last name for all of his 55 years—it's Lee-bus not Lybs. Although he doesn't make a big deal of the frequent mispronunciations, it may have been partly why he became a teacher back in 1968.

The real reason he went into and stayed in teaching, Libes explained, was that he couldn't stand the rat race. He currently teaches digital electronics and microcomputer design at Union County College in New Jersey.

Libes joked about the communication skills of engineers. "Engineers can't write, read, or communicate verbally. But there are some terrific hardware and software designers out there," he said.

Libes first became interested in computers in the early seventies. Using an article Don Lancaster had published in a 1973 issue of *Radio and Electronics*, he and his students converted a TV into a terminal, interfaced it to a modem, called up their IBM computer at school, and "we were shocked at what we had done and how easy it was to do."

The next experiment was to build a small microcomputer with an Intel 8008 microprocessor chip. "In trying to do that project we learned that Intel was not providing any help at all," Libes said. "If you were going to buy 10,000 parts from them they would provide some assistance. But if you were just building one system they didn't want to know you."

"I started looking around for help. I found there was a very active computer society called the Amateur Computer Society, and I joined it. Steve Grey had organized this group back in 1965," Libes said.

"Then the Altair came out in 1975, and we had even more problems with it than the Intel 8008." As an effort to educate himself and others, Libes formed the New Jersey Amateur Computer Group in May 1975. One of the major accomplishments of this pioneering organization was the creation of the first personal computer show. Thus the Trenton Computer Festival was first held in 1976 and has been held annually ever since.

Libes first became involved with magazine publishing when he was writing a regular column—a gossip column on industry happenings—for *BYTE* starting in 1978. He also was using CP/M and an S-100 system while doing consulting work for a number of companies designing microcomputer systems.

Although several computer magazines were springing up during that time, no magazine was specifically oriented to CP/M and S-100 system users. "I approached several publishers with the idea of starting a magazine for this audience, and they thought I was crazy. Finally, after being turned down by everyone, I decided if they didn't want to publish one I would—being the fool that I was!"

In December 1979, *Microsystems* was born on Libes's Ping-Pong table in the basement of his house. Libes can laugh now about the mail list program that crashed the night before mailing the first issue. The subscriber list was lost for three hours.

"The magazine immediately turned into a big business. It took up every spare

minute of my time. My wife and I were literally working 24 hours a day, 7 days a week," said Libes, remembering the exhaustion. A few friends also pitched in and helped out.

Libes did not enjoy the publishing side of the magazine, especially dealing with the advertisers, subscribers, printers and mailing services. "I would receive long distance subscriber calls at 3 a.m. from Denmark. At that point I decided I wanted out."

"David Ahl [publisher of *Creative Computing*] was trying to build a publishing empire at the time," said Libes. Libes sold most of his equity interest to Ahl after six months of publishing *Microsystems* himself. At that point the magazine had a circulation of about 8,000.

When hard times fell on Ahl he sold *Creative Computing* to Ziff-Davis Publishing. *Microsystems* was part of the deal. Libes remained at *Microsystems* as editor until March 1984.

He always did his editing at home. "I'm not the type of person who can work in an office 9 to 5," Libes said. "If I went to an office I wouldn't accomplish a damn thing. Offices are an endless series of coffee breaks and meetings."

"What made *Microsystems* such a unique magazine was its practical orientation," said Libes. But despite a fervent

Of course, POWER!™ saves your Bad Disk.

NOW! WINDOWS FOR IBM!

It also does 54 other things to keep your disk in line.

EVERYTHING YOU ALWAYS WANTED TO DO, BUT WERE AFRAID TO TRY

Unlike some utility programs that are a headache to use, POWER! is engineered to spoil you with 55 features, simple and uniform commands, and utter simplicity of use. POWER! automatically alphabetizes and numbers your files. You select by the number and never type file names again. Need to **[COPY]**, **[RENAME]**, **[ERASE]**, or **[RUN]** programs? Just type in their menu number! POWER! also locks out your disk's bad sectors **[TEST]** without destroying files–a critical difference from other utilities that search and destroy, without informing you what they've done, leaving you to wonder why your programs won't run. (And POWER! still has 50 commands to go!)

POWER! ONE PROGRAM DOES IT ALL!

You may own a few utility programs for your computer housekeeping, each with its own commands to memorize. POWER! has all the programs rolled into one 16K integrated package, so you do things you've never tried before–every day. Save sensitive data from prying eyes with **[PASS]** word protect, move a block of memory **[MOVE]**, look for data **[SEARCH]** or compare files **[CHECK]**. POWER! also makes easy work of patching, **[DISPLAY/SUBSTITUTE]**, customizing software **[LOAD/SAVE]**. Among the other commands are **[SIZE]**, **[STAT]** **[LOG]**, **[DUMP]**, **[TYPE]**, **[JUMP]**, **[FILL]**, **[SET]**, and the CP/M version lets you restore erased files–even when you don't remember the filename–at a flick of the POWER! **[RECLAIM]** command. (Still 31 commands to go!)

POWER! NOW FOR IBM's PC-DOS AS WELL AS CP/M

We first developed POWER! for CP/M two years ago, and a stack of testimonials from FORD to XEROX testify to its excellence. For IBM-PC™ users, special features like managing sub-directories, **[CHANGE]**, and a separate creation of up to 8 simultaneous, on-screen **[WINDOWS]** have been added.

MONEY-BACK GUARANTEE AND A 10 DAY TRIAL

POWER! has the Seal of Approval from the Professional Software Programmers Association, and you, too, must be happy with POWER!–or your money back! For only $169 you can now really be in control of your computer. Call Computing! at (415) 567-1634, or your local dealer. For IBM-PC or any CP/M machine. Please specify disk format.

The company that earns its exclamation point.

COMPUTING!

2519M Greenwich, San Francisco, CA 94123

TO ORDER CALL 800 TOLLFREE
800-428-7825 Extension 96M
In CA: 800-428-7824 Extension 96M

IBM and IBM-PC are registered trademarks of International Business Machines Corporation.

CIRCLE 37 ON READER SERVICE CARD

"I think CP/M will gradually, over a period of time, fade into oblivion."

following, a "circulation problem" led to Ziff-Davis ending the publication.

"We were never able to get the circulation up to the point where it was worthwhile for Ziff-Davis," said Libes. "They needed a circulation of over 100,000, and we were catering to a very sophisticated readership, which is a very limited audience."

Libes had a very good relationship with Ziff-Davis. He feels they are a professional publishing house that treated him well. "But they're in business first and foremost. The bottom line is what they're interested in."

Libes has some strong opinions about CP/M, the operating system that was largely responsible for the launch of *Microsystems*. "CP/M-86 really had the potential to be something," he said. "But Digital Research was very slow in getting it out. They really did not seem to put a push on it."

"It came out so late that Seattle Computer Products, which was putting out an 8086-based system, gave up waiting for it. They had Tim Patterson write an operating system for them that emulated CP/M. Microsoft bought the operating system, which came out on the market six months to a year before CP/M-86."

"If Digital Research had brought out CP/M-86 when it had promised, MS-DOS would have never been born," said Libes. "And IBM would have gone with CP/M-86. They only have themselves to blame for giving away the market to Microsoft and MS-DOS. DRI is not an aggressive organization, and they are not marketing oriented, while Microsoft is hungry."

Libes is convinced OEM manufacturers and application software developers are moving away from CP/M. "I think it will gradually, over a period of time, fade into oblivion," he said.

"There is no doubt that there is a tremendous amount of application software running under CP/M," said Libes. "But since the introduction of the IBM PC and MS-DOS, commercial software suppliers are no longer developing software for CP/M. Probably only 300,000 to 500,000 systems are running on CP/M. There are probably about 3 million IBM PCs and clones. CP/M is a minor segment of the marketplace and CP/M-86 has never even gotten off the ground."

Although a CP/M and S-100 magazine, *Microsystems* also covered UNIX quite thoroughly. "With the amount of emphasis we put on UNIX, we were really too far ahead of the marketplace," said Libes.

"In early 1983 I became concerned with the future of CP/M. At that point we were very closely wedded to CP/M. We were *the* CP/M magazine. We had to shift our direction. I started to give coverage to MS-DOS."

"But Ziff-Davis was bringing out several magazines catering to the IBM PC user—*PC* and *PC Tech Journal*. I really did not want to step on their toes, so we sort of limited our coverage of MS-DOS."

"Instead I decided to direct our attention to UNIX. I felt that more and more sophisticated micro software developers would be moving toward the UNIX area. We probably made a mistake, because a magazine cannot lead its readers in a direction. I think you have to follow the readers and support them. You can't show them the way."

"Although CP/M was declining, it was declining at a slow rate," said Libes. "We shouldn't have moved so quickly and so aggressively. Our readers still wanted support for CP/M."

"I still think we were moving in the right direction by giving support to UNIX," reflected Libes. "Now we have IBM introducing Xenix on the AT. This would have given *Microsystems* a tremendous boost. It is unfortunate that Ziff-Davis did not want to stick with the magazine during this transitional period."

"As for the S-100 market, it will probably continue for quite some time. Most of the manufacturers are putting together multi-user systems and have largely abandoned the single-system market. Three of the S-100 manufacturers are already selling UNIX-based systems," said Libes.

"This microcomputer industry is not really that unique if you look at other industries. We are following in their footsteps, repeating the same evolutionary process."

"If you go back before World War I, there were over 1,000 auto manufacturers.

Many of them were just garage-type operations. Now we have four automobile manufacturers. I'm sure that is what it will come down to in the microcomputer industry. Independents will be forced out. And I have no doubt IBM will be the leader."

Libes certainly misses *Microsystems*, but his life sounds full without it. He is in his sixteenth year as a professor of electronic technology at Union County College in New Jersey.

He also is working on a grant for the state of New Jersey. The grant has to do with the use of microcomputers in electrical engineering—specifically, integrating spreadsheet programs and data aquisition systems in electrical engineering.

On the side, Libes is very active in the distribution of public domain software. He still works with the New Jersey Amateur Computer Group and the SIG-M PC Blue public domain software library. He proudly proclaims himself to be a computer hobbyist who likes to be both a teacher and a student.

Libes is also an amateur musician—he plays classical and jazz guitar. He also enjoys hiking in the mountains. But he admits he still spends quite a bit of time playing with his machines.

His wife, Lennie, is a professor of mathematics at the County College of Morris in New Jersey. His two children are both highly involved in science. His daughter, Susan, is a professor of ocean chemistry at the Univ. of South Carolina. "She likes to work with large quantities of chemicals," Libes chuckled. His son, Don, is involved in researching the next generation of robotics.

What does the future hold for Libes? He is interested in starting some new version of *Microsystems*—though with a different title since Ziff-Davis owns the rights to the name. He wants it to be a small magazine that comes out possibly every other month.

"I don't want to worry about circulation and advertising the way Ziff-Davis was constantly concerned," said Libes. Ideally he'd like to team up with a small publisher who will handle all the publishing aspects of the magazine. He would just concentrate on editing.

To date Libes has received no takers. But there is still a sparkle in his eye that indicates he is not finished with his magazine career.

Perhaps the next interview will be titled, "Play it again, Sol." ■

Carl Landau is publisher of COMPUTER LANGUAGE.

BRIEF

Program Editing is finally both Intuitive *and* Powerful

... and configurable to suit your style

BRIEF lets you concentrate on programming by keeping the Editor "out of the way," while combining power and natural flow:

The New Standard. No longer does an Editor have to be "in your way" to provide full power. By combining power *with* natural flow, the new advanced BRIEF is in a class by itself.

BRIEF lets you concentrate on programming. Your thoughts flow smoothly, intuitively. 15 minutes is all you need to become fully productive. You can then do precisely what you want quickly, with minimum effort and without dull repetitions.

BRIEF adapts to your style. You can use BRIEF without modification, because it's distributed with an "ideal" configuration. Or you can make any change you want, add any feature of your own. Reconfigure the whole keyboard or just the Function Keys. Change the way the commands work or just the start-up defaults.

Availability: PCDOS-compatible systems with at least 192K and one floppy drive are required. Though your initial copy is protected, an unprotected version is available when you register BRIEF. Ask for special IBM AT or Tandy 2000 versions.

Pricing: Only $195 . . . with discounts for volume end-users. A demonstration version is available for only $10 and can be available towards any Solution Systems purchase.

Win $1,000 and substantial recognition for the Outstanding Practical BRIEF Macro. Other awards will also be given. Macros are fully programmable.

BRIEF'S PERFORMANCE IS NOT EQUALLED IN MICROS, MINIS AND MAINFRAMES

- Full UNDO (N TImes)
- Edit Multiple Large Files
- True Automatic Indent for C
- Exit to DOS Inside BRIEF
- Uses All Available Memory
- Intuitive Commands
- Tutorial
- Repeat Keystroke Sequences
- Windows (Tiled and "Pop Up")
- Unlimited File Size
- Reconfigurable Keyboard
- Online Help
- Search for Complex Patterns
- Mnemonic Key Assignments
- Horizontal Scrolling
- Comprehensive Error Recovery

PLUS a Complete, Powerful, Readable, Compiled MACRO Language

RISK-FREE Editing is possible because of UNDO. All operations can be undone, including global revisions, massive deletions, typing, etc. Up to 100 such operations can be undone. Experiment while you edit. No other known mirco software product has such a powerful UNDO.

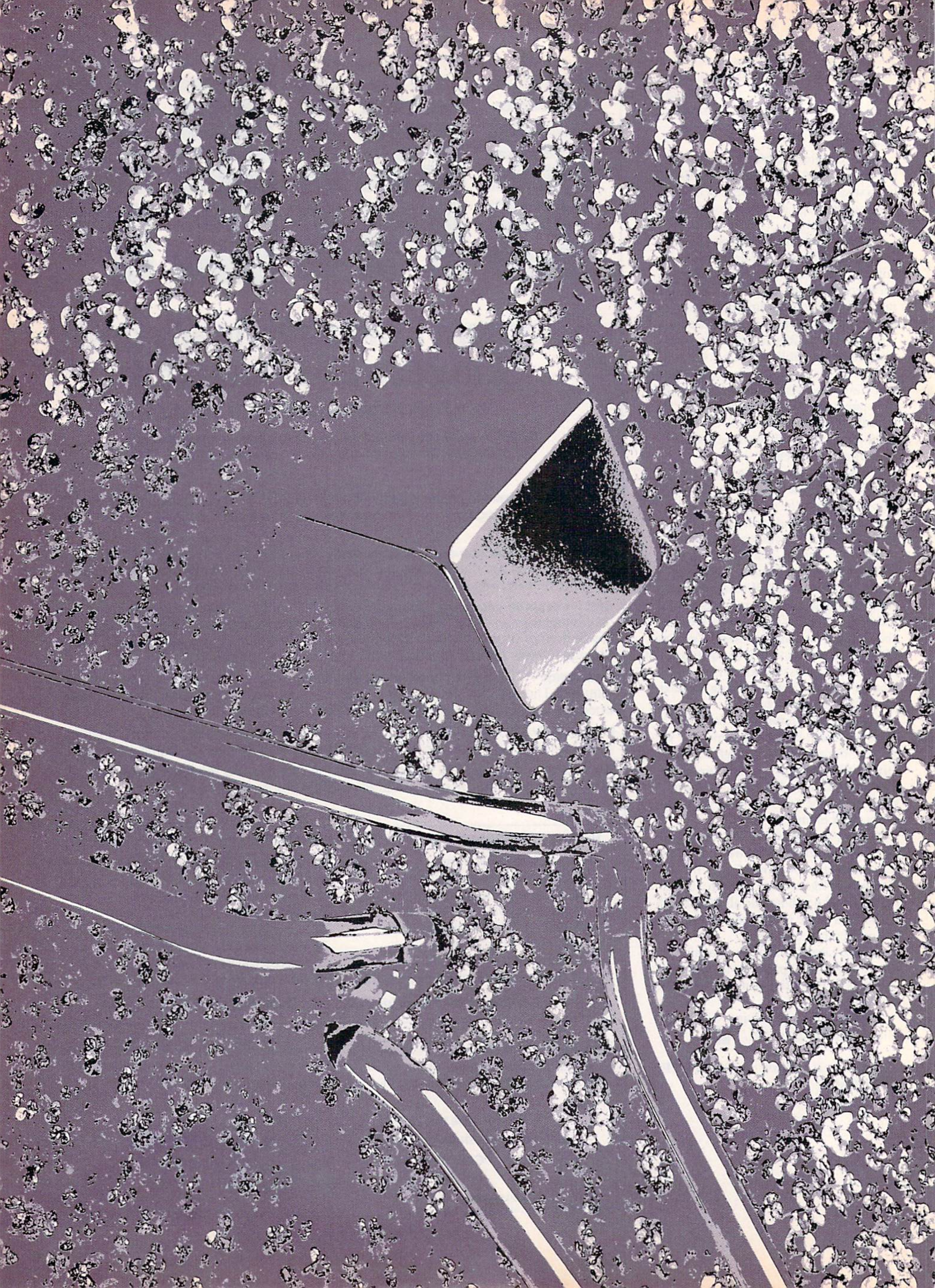
**Try BRIEF. Use the Demo . . .
or the full product for 30 days.
Call or write us . . .
617-659-1571**

Solution Systems™

BRIEF is a trademark of UnderWare.
Solution Systems is a trademark of Solution Systems.

335-L Washington St., Norwell, MA 02061

CIRCLE 80 ON READER SERVICE CARD



Macros & Procedures

By Morton F. Kaplon

One way to identify a real mathematician, I am told, is to give him (generically speaking) a problem of some complexity and see how he goes about solving it. The real mathematician's approach will always be one in which he does a few introductory steps and then says—"I have now reduced this to a problem previously solved."

Good programming reflects the real mathematician's approach even though we may not perceive it that way. Every programming language utilizes this approach with built-in routines that make it easy to do arithmetic, take a square root, evaluate trigonometric functions, handle strings, etc.

In BASIC, for example, the ability to multiply two numbers and display the result with the command *PRINT 345*57* shows this approach. The person who wrote the interpreter developed a routine for multiplying two numbers and then set things up so that it could be used over and over again by following the rule indicated above; for your benefit this person reduced the problem to one already solved.

The higher level languages take this one even further. Besides including a variety of "ready done" things for you (depending on the language and its realization, of course) many of them have created ways that allow you to do the same things for yourself.

Depending on the language, these things are given different names. BASIC has the subroutine concept, C has functions, Pascal has functions and procedures, and so on.

For example, if an engineer requires some very esoteric function that is used over and over again, practically any high-level language offers the ability to write, just one time, the code to evaluate the function and then, as needed each time, call for its evaluation by the code already written. High-level languages even allow you to nest procedures (to include procedures inside procedures or subroutines inside subroutines, etc.).

My intent is to discuss how the real mathematician is implemented at the assembly language level and to focus on one aspect of its implementation—macros—that does not have a ready counterpart in higher level languages. At the same time I will compare macros with procedures and consider the relative merits and utilities of each.

The particular assembly language used for examples is that of the Intel 8088 as realized in the macro assembler for the IBM PC. The comparison will primarily contrast execution time, run-time code size, and ease of use, and it will indicate the ability to accomplish certain tasks with macros that are not attainable by procedures.

What is a macro?

The word macro is used in several contexts in relation to programming and computers, but they all appear to imply the same thing. The basic idea is that a macro is standing in for something, and when the macro is used, it is replaced by what it was standing in for.

The concept involves two things: defining the macro(s) and using the macro(s). Depending on the context, the method of definition and the method of use is different, but the end result, so to speak, is the same.

A rather common and most useful form is the keyboard macro. A variety of programs enable the user to define a certain key or combination of keystrokes to represent a string of characters or even other keystrokes.

Anyone who has ever been introduced to macros finds them enormously useful tools. If you are writing text and the same expression recurs, rather than type it in each time, you assign some specific keystroke to represent it. When you enter that keystroke, the expression in its entirety is entered. The assigned keystroke is a stand-in for the desired expression, and when the keystroke is entered (the macro is invoked), the expression is displayed (the macro is expanded).

The application we are interested in is programming, and I will proceed somewhat generally for the moment and illustrate the use of macros with the multiplication example referred to in the introduction.

Suppose we want to define a macro to multiply two numbers. To do this we have to write in assembly language the code that does the multiplication, and we have to do something that defines a name for the macro, where its code begins, and where its code ends. We also need some way of passing to the program the numbers to be multiplied together.

I will use 8088 macro assembly language and explain what it means as I go along. The required commands relevant to the use of macros are displayed in UPPER CASE and the input from the programmer (symbolically reflecting the code to accomplish the task) in lower case; comments will follow on each line after the ";".

To define a macro whose name is *multiply*, the lines of code in Listing 1 must be included in the assembly language source code at a location whose specification we will discuss later. Because macros and procedures employ very different methods of passing parameters to be used, particular attention must be paid to this. To use this macro in the source code to multiply two specific numbers together, the macro is invoked as indicated. The "—" stand for lines of other code in the program that precede and follow the invocation of the macro.

Note that the lines in Listing 2 are in source code, and before you have a program that can be executed, the source code has to be translated (assembled) to machine code. When the assembler does the translation from source code to machine code, it does the following:

When the assembler scans the source code, if it comes across a name that it does not recognize as a valid command code or pseudo-op code, it looks in its table of macro names to see if that name corresponded to a macro defined somewhere in

```
Defining the macro "multiply"

multiply  MACRO  x,y        ;"multiply" is name of macro--this line marks
                            ;the beginning of the macro and says the two
                            ;numbers  represented by the  parameters x,
                            ;y are to be used in the code below as the
                            ;stand-in operands for multiplication
line 1                      ;the programmer supplied code in the MACRO
line 2                      ;of programmer supplied macro code
--------
mov    ax,x                 ;the parameter x is put in register ax
mov    bx,y                 ;the parameter y is put in register bx
                            ;these lines of code will be referred to as
                            ;the p(arameter passing) code.
--------                    ;note that somewhere in this section of
--------                    ;macro code we might have a command
--------                    ;moving the result to some specific location
line n                      ;(last line) of programmer supplied code
          ENDM              ;this indicates the end of the MACRO code.
```

Listing 1.

```
Using the macro "multiply"

------------------          ;the two numbers to be actually multiplied
------------------          ;are passed to the program and replace the
multiply 345,47             ;stand-in parameters when the macro is
------------------          ;expanded
------------------
------------------
```

Listing 2.

the source code. If the assembler finds that macro, it then expands the macro by replacing its invocation by the actual lines of code contained within the body of the macro, substituting for the parameters *x* and *y* the actual values of 345 and 47, respectively, that were passed with its invocation.

Thus when the source code is prepared for assembly into machine code, what the translator sees is as in Listing 3.

When the macro is assembled, it is expanded at that location to its full textual value, and any parameters that were defined for it are substituted for in the expanded text, as indicated previously for *x* and *y*.

We will characterize the code, in its assembled form, as consisting of *a* bytes associated with the non-parameter passing aspect and *p* bytes associated with the parameter passing. Thus this macro when assembled will consist of $a + p$ bytes of executable code. If it is a non-parameter passing macro, then it has only *a* bytes. In the previous example, the parameter passing aspect comprises the two lines involving *x* and *y* in the macro definition.

Thus, if the macro *multiply* was used 37 times in the source code and if the size in bytes of the assembled code inside the macro was 50 bytes $(a+p)$, then the assembled code will contain 37 × 50 bytes of code, representing the use of the macro *multiply* 37 times.

This is quite different from the way the real mathematician is handled in the translation of higher level languages. It is important to understand and consider that difference, as it is reflected at the assembly language level where the counterpart is called a procedure in the 8088 assembly language.

Procedures

In 8088 assembly language the real mathematician may also be implemented by the use of procedures. The way that procedures are used is rather similar conceptually to subroutines in BASIC and, more importantly, what actually occurs is similar to the utilization of functions and procedures in higher level languages.

Using procedures (or their equivalent in other assembly languages) is again a two step process—they must be defined and they must then be used. Again the example is for the Intel 8088 in the IBM macro assembler, using *multiply*. The operands for the multiplication will be passed using the same form as in the macro; this will enable meaningful comparisons to be made between procedures and macros and it also represents the minimal way of effecting the procedure with respect to code utilization.

Somewhere in the source code must be the lines of code to define the procedure, as presented in Listing 4. In order to understand the difference between a

```
Source code of expanded macro as presented to the assembler

line 1              ;of programmer supplied macro code
line 2              ;of programmer supplied macro code
--------
mov     ax,345      ;345 replaces x and is put in register ax
mov     bx,47       ;47 replaces y and is put in register bx
--------
line n              ;(last line) of programmer supplied code
```

Listing 3.

```
Defining a procedure

multiply  PROC  FAR  ;define a procedure of type FAR, named "multiply"
line 1 of procedure  ;programmer supplied code
line 2 of procedure  ;of programmer supplied code
---------            ;note, no parameter passing code inside PROC
---------            ;again, the code here must contain the location
---------            ;of the answer; for procedures, the stack is
---------            ;not always the ideal location.
line n of procedure  ;(last line) of programmer supplied code
          RET        ;RETurn to location where "multiply" was invoked
multiply  ENDP       ;Indicates the End of PROCedure "multiply"
```

Listing 4.

macro and a procedure we must again look in a little detail at how the procedure is handled by the assembler (Listing 5).

When the assembler sees a procedure being defined, it takes the code inside the procedure, translates it into executable machine code, and stores it at that location in the program where it was defined. When it comes across the invocation of the procedure *CALL multiply*, it checks to see that such a procedure name exists, determines the location in the program sequence where it is, and then proceeds to transfer program execution from the current location to the location of the executable code for *multiply*.

Before the assembler does the transfer, it saves the address in the program code of the next instruction after *CALL multiply* by pushing it on the run-time stack; for a procedure of type *FAR* it would first save the address of the segment in which the program is currently running by pushing that on the run-time stack and then pushing the address of the next instruction.

The assembler then transfers to the location where the machine code for *multiply* is, executes it, and in that execution, when it sees the code for *RET*(urn), takes the values stored on top of the stack to use to return back to the next instruction after *CALL multiply*.

Each time the assembler sees a *CALL multiply* it does the same thing. Thus we note the first difference. The source code that does the *multiply* is only translated into executable machine code once. No matter how many times a *CALL* is made for the services of *multiply*, the same executable code in the same location is used. An obvious savings occurs in the amount of executable code as compared to the macro for repeated invocations—just how much we shall see shortly.

But there is, as with anything, a price to be paid. Note that no parameters are passed on the line that was used to invoke *multiply*. They were passed before the procedure *multiply* was *CALL*ed, in this example, utilizing the same code as was utilized for the body of the macro in passing parameters. Thus, when parameters are to be passed, we must account for this code.

In addition, there is another overhead in execution time. With a macro, the machine code is in-line, ready to be executed each time the macro is invoked. But with a procedure, the machine has to save one (or two, if the procedure is of the *FAR* type) addresses on the stack; transfer execution to the location of the procedure *multiply*; and at the conclusion of the procedure, recover those addresses and transfer back. So a procedure will take more time to execute than a macro— the extra time is required for the execution of these actions (the *CALL* and the *RET*urn).

Because the addresses needed for transfer back to the *CALL*ing location are on the stack, the programmer must be very careful that the code in the procedure does not change the top of the stack at the conclusion of the procedure from its initial value entry to the procedure. Therefore it makes the stack, as a means of passing parameters to and from the procedure, much less flexible for use than it would ordinarily be. (Some CPUs maintain separate run-time stacks, not accessible to the programmer, so this problem is eliminated.)

Finally, a brief explanation is needed on the type of procedure. The 8088 processor, for example, has a segmented structure, with each segment 64K bytes in size. If the code for a procedure is in the same segment as existed in the program execution at the time of its *CALL*, then the procedure type is said to be *NEAR*. If the procedure is in a different segment, then the type is *FAR*. For a *FAR* type, the address of the current segment must be saved as well as the address of the next instruction to be executed in that segment so that the processor knows which segment to return to.

Table 1 compares features of macros and procedures by showing their dependence on various parameters.

We can, at this point, draw some general conclusions as to relative performance. The most interesting question is the value of *N* for which macro and pro-

```
Using a procedure

----------------        ;the values to be used in the procedure must be
----------------        ;passed to it before it is CALLed since there
                        ;is no way to pass them on the CALLing line as
----------------        ;for macros.  We do it in the same way as in macro
mov        ax,345       ;body; pass one of the operands needed for
mov        bx,47        ;multiply and then pass the other operand
CALL     multiply       ;CALL for the procedure "multiply"
----------------        ;continue on with program code
----------------        ;
```

Listing 5.

cedure run-time code is the same. Equating the two values for run-time code yields, after some very elementary algebra, $N \doteq (a+R)/(a-C)$, independent of p (which should not be surprising, considering how it was handled).

Since R and C may depend on the type (*NEAR* or *FAR*), we can specify no more at this point, except to note that if a is very large, much larger then either R or C, then N will be approximately 1, for that case. C is 3 or 5 bytes depending on whether the type is *NEAR* or *FAR*. R is 1 byte for either type if there are no additional stack pops.

It is clear that the run-time code size increases much more rapidly with repeated invocation for the macros than it does for the procedures. However, consider execution time, listed in Table 2, in terms of clock cycles. There we see clearly displayed the execution time penalty for the use of procedures. A procedure will always take at least $N^*[t(C)+t(R)]$ clock cycles longer per invocation than a macro.

Whether this is acceptable or not depends of course on the application. If the procedure is being used to display error messages or prompts, then clearly the execution time per invocation is relatively unimportant and the savings in run-

Comparison of macros and procedures

Function	**Macros**	**Procedures**
Define beginning	macroname MACRO p1,.,pn	procname PROC TYPE
Executable code	<between Define Beginning and Define Ending>	
Define ending	ENDM	RET procname ENDP
How invoked		
Without parameters	macroname	CALL procname
With parameters	macroname v1,.,vn	parameters v1,.,vn passed in code before CALL procname
Execution	Code expanded in-line	At location of PROCEDURE RETurn to CALL location
Source code size	Size of MACRO + N * (size of invocation)	Size of PROC + N * (size of invocation) + N * (parm passing code)
Run-time code	N * (a + p) bytes	(a * R) + N * (C + p) bytes
Execution overhead	None	Stack use, IP transfer
LIBRARY inclusion	Relatively easy, no additional overhead	Requires care and planning, overhead

a = assembled code size in bytes, non-parameter passing
p = assembled code size in bytes for specific parameter passing
N = Number of invcations iof MACRO or PROCEDURE
C = bytes of assembled code associated with CALL
R = bytes of assembled code associated with RET(urn)
vn = actual value of nth parameter value passed
pn = place value of nth parameter
IP = instruction pointer/program counter

Table 1.

Pascal and C Programmers

Your programs can now compile the *FirsTime*™

FirsTime is an intelligent editor that knows the rules of the language being programmed. It checks your statements as you enter them, and if it spots a mistake, it identifies it. *FirsTime* then positions the cursor over the error so you can correct it easily. *FirsTime will identify all syntax errors, undefined variables, and even statements with mismatched variable types.* In fact, any program developed with the *FirsTime* editor will compile on the first try.

Unprecedented

FirsTime has many unique features found in no other editor. These powerful capabilities include a zoom command that allows you to examine the structure of your program, automatic program formatting, and block transforms.

If you wish, you can work even faster by automatically generating program structures with a single key-stroke. This feature is especially useful to those learning a new language, or to those who often switch between different languages.

Other Features: Full screen editing, horizontal scrolling, function key menus, help screens, inserts, deletes, appends, searches, and global replacing.

Programmers enjoy using *FirsTime*. It allows them to concentrate on program logic without having to worry about coding details. Debugging is reduced dramatically, and deadlines are more easily met.

FirsTime for PASCAL	$245
FirsTime for C	$295
Microsoft PASCAL Compiler	$245
Microsoft C Compiler	$395
Demonstration disk	$25

Get an extra **$100 off** the compiler when it is purchased with ***FirsTime.***
(N.J. residents please add 6% sales tax.)

Spruce
Technology Corporation
110 Whispering Pines Drive
Lincroft, N.J. 07738
(201) 741-8188 or (201) 663-0063

Dealer enquiries welcome. Custom versions for computer manufacturers and language developers are available.

FirstTime is a trademark of Spruce Technology Corporation.

MasterCard VISA

CIRCLE 33 ON READER SERVICE CARD

time code is preferable. On the other hand, if the application is one in which time-consuming operations become important, such as sorting, searching, or screen manipulation, then the savings in execution time may have priority over any savings in run-time code.

Though a flat statement cannot be made to cover all possibilities, experience strongly suggests that in those cases where the *a* code size is appreciable and repeated invocation is anticipated, the use of procedures is to be preferred unless execution time is of the absolute utmost priority.

If you're really ambitious about following up on this discussion on macros and procedures, call the *COMPUTER LANGUAGE* Bulletin Board Service or this magazine's account on CompuServe to download two very long examples that use the same services but implement them as macros (testmac.asm) and as procedures (testproc.asm). The file name on the BBS/CompuServe will be called MACPRO.LTG. In each case a detailed accounting in the source code of run-time byte size is kept.

These are executable (albeit trivial) programs written in a format to produce a .COM file. They provide a useful framework for testing macros or procedures in such a way that the assembled run-time code displayed in the directory for the .COM file is the same as that calculated using the data from the Intel manual on code size.

So many diverse ways exist to utilize macros that it is difficult not to continue. However, good instructional practice requires the reader to explore and discover on his or her own. Think of the many things you do repetitively in writing programs and create the macro structures to accommodate them. Build your libraries and use them. Doing this will make programming go much faster.

Morton Kaplon has a Ph.D. in physics and teaches computer science at City College of New York. He considers this job to be his third career; he was formerly a practicing physicist—with more than 70 articles on physics published—and vice president of administration at the City College of New York.

Execution time

Function	**No. of invocations**	**Clock cycles: No parameters passed**	**Clock cycles: Parameters passed**
Macro	N	N*t(a)	N*[t(a)+t(p)]
Procedure	N	N*[t(a)+t(C)+t(R)]	N*[t(a)+t(p)+t(C)+t(R)]

t(a) = number of clock cycles for code a
t(p) = number of clock cycles for code p
t(C) = number of clock cycles for the CALL (type dependent)
t(R) = number of clock cycles for the RETurn (type dependent)
t(C) is 19(23) or 28(36) depending on the type being NEAR or FAR. t(R) is 20 or 32 depending on the type being NEAR or FAR. (The number in the ( ) is used if the instruction operand is a word.)

Table 2.

NGS FORTH

A FAST FORTH OPTIMIZED FOR THE IBM PERSONAL COMPUTER AND MSDOS COMPATIBLES.

***79 STANDARD**

***FIG LOOKALIKE MODE**

***PC-DOS COMPATIBLE**

***ON-LINE CONFIGURABLE**

***ENVIRONMENT SAVE & LOAD**

***MULTI-SEGMENTED**

***EXTENDED ADDRESSING**

***AUTO LOAD SCREEN BOOT**

***LINE AND SCREEN EDITORS**

***DECOMPILER & DEBUGGING AIDS**

***8088 ASSEMBLER**

***BASIC GRAPHICS & SOUND**

***NGS ENHANCEMENTS**

***DETAILED MANUAL**

***INEXPENSIVE UPGRADES**

***NGS USER NEWSLETTER**

A COMPLETE FORTH DEVELOPMENT SYSTEM.

PRICE: $70

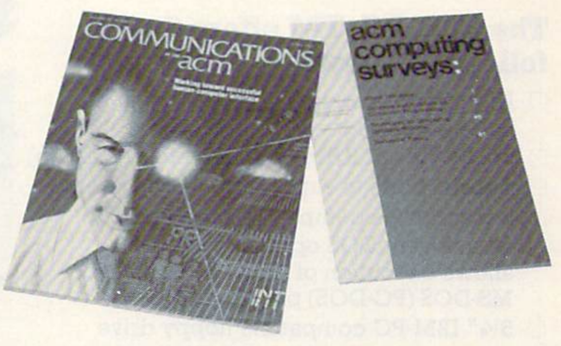
PLEASE INCLUDE $2 POSTAGE & HANDLING WITH EACH ORDER.
CALIFORNIA RESIDENTS: INCLUDE 6.5% SALES TAX.

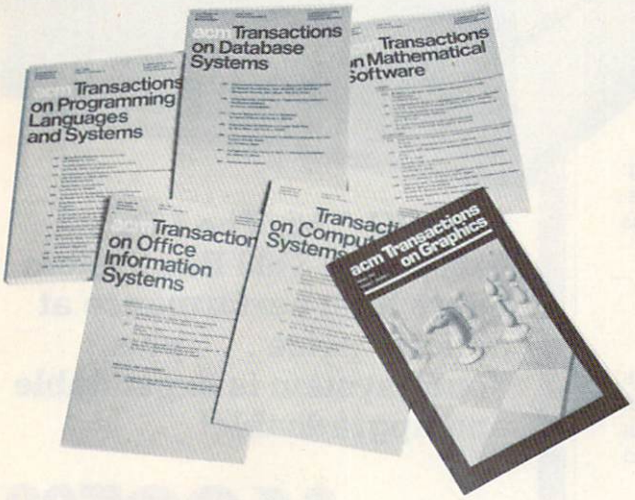
NEXT GENERATION SYSTEMS
P.O.BOX 2987
SANTA CLARA, CA. 95055
(408) 241-5909

CIRCLE 45 ON READER SERVICE CARD

Journals you read cover to cover

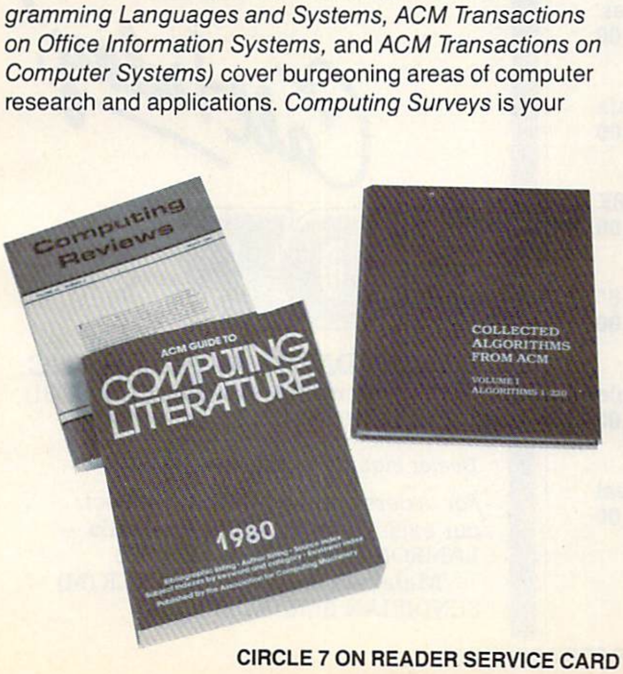
From computing applications to computing theory, from matters of practical importance to those of scientific research, ACM journals offer you high quality informative articles. Each undergoes a thorough review to insure its accuracy, topicality, and pertinence. Every journal is specially tailored to serve the specific needs of the computing community.




acm Transactions on Database Systems
Transactions on Mathematical Software
Transactions on Programming Languages and Systems
Transactions on Office Information Systems
Transactions on Computer Systems
acm Transactions on Graphics

While *Communications of the ACM* has made its mark publishing landmark research papers in computer science, today the magazine is moving into a broader overview role. The editorial aim is to publish broad-gauge, high quality, highly readable articles on key issues and major technical developments in the field. The various transactions *(ACM Transactions on Mathematical Software, ACM Transactions on Database Systems, ACM Transactions on Graphics, ACM Transactions on Programming Languages and Systems, ACM Transactions on Office Information Systems,* and *ACM Transactions on Computer Systems)* cover burgeoning areas of computer research and applications. *Computing Surveys* is your source for comprehensive surveys, tutorials, and overview articles on topics of current and emerging importance. The *Journal of the Association for Computing Machinery* presents fundamental ideas that are of lasting value to the understanding of computation.

ACM also publishes unique reference sources. *Computing Reviews* contains original reviews and abstracts of current books and journals. The *ACM Guide to Computing Literature* is an important bibliographic guide to computing literature (available annually on Standing Order Subscription). *Collected Algorithms from ACM* is a collection of ACM algorithms available in printed version, on microfiche, or on machine-readable tape.


Computing Reviews
ACM GUIDE TO COMPUTING LITERATURE
1980
COLLECTED ALGORITHMS FROM ACM
VOLUME I
ALGORITHMS 1-220

For further information about ACM journals, send for your free ACM Publications Catalog today!

☐ Please send me a free ACM Publications Catalog.
☐ Please send me subscription information for the following journal(s):

___ Communications of the ACM	ACM Transactions on:
___ Journal of the ACM	___ Mathematical Software
___ Computing Reviews	___ Database Systems
___ Computing Surveys	___ Programming Languages & Systems
___ Collected Algorithms from ACM	___ Graphics
___ ACM Guide to Computing Literature	___ Office Information Systems
	___ Computer Systems

☐ Member No. ___________ Expiration Date ___________
☐ I am not an ACM member, please send me membership information.

Name_______________________________

Address_____________________________

City_______________ State_______ Zip_______

Mail to. Publications Department,
ACM, 11 West 42nd Street, New York, NY 10036.

CIRCLE 7 ON READER SERVICE CARD

LOMAS DATA PRODUCTS INVITES YOU TO:

SHARE THE THUNDER.

The S100-PC-TM offers the following standard features:

- ☐ High performance THUNDER186 8Mhz 80186 processor
- ☐ 512K bytes of RAM (expandable to 1Mbyte)
- ☐ 4 serial ports to support up to four users
- ☐ 3 Centronics compatible parallel ports
- ☐ Concurrent DOS operating system allows execution of both CP/M-86 and MS-DOS (PC-DOS) programs
- ☐ 5¼″ IBM-PC compatible floppy drive
- ☐ 40 Mbyte high performance Winchester drive
- ☐ Attractive 10 slot desktop enclosure

In addition, a number of options are available including: larger Winchester drives, more user ports, 80286 processor, graphics support and additional operating systems (MS-DOS and CP/M-86).

S100 BUS boards products & support for the system integrator . . .

All of LDP boards are fully tested to exacting standards and carry a one year warranty. We specialize in 16-bit products & support the major operating systems for 16-bit processors: CP/M-86*, CONCURRENT CP/M-86*, and MS-DOS (PC-DOS).

■ THUNDER186 — THE ONLY COMPLETE S100 BUS, 16 BIT SINGLE BOARD COMPUTER AVAILABLE TODAY.
Concurrent CP/M-86, which in addition to running CP/M-86 programs, runs MS-DOS programs. Comes complete, ready to plug into an enclosure and run. 256K bytes of RAM only . . . . **PRICE $1595.00**

■ LIGHTNING ONE*8086/8088 CPU**
8086 or 8088, with 8087 and 8089 coprocessors. Up to 10 MHZ operation . . . . **PRICES start at $425.00**

■ HAZITALL SYSTEM SUPPORT BOARD
2 serial, 2 parallel ports, battery protected clock calendar. Hard disk controller host interface . . . . . . **PRICE $325.00**

■ LDP 128/256K DYNAMIC RAM
Advanced dynamic RAM with LSI controller for failsafe operation, parity **PRICE** 128K—**$495.00**, 256K—**$795.00**

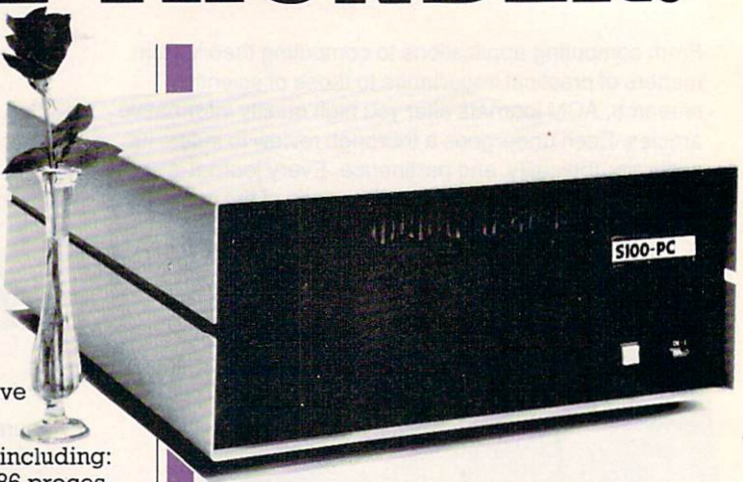
■ RAM67 HIGH PERFORMANCE STATIC RAM
High speed (100ns) low power CMOS static RAM. 128K bytes, extended addressing . . . . **PRICE $995.00**

■ LDP72 FLOPPY DISK CONTROLLER
Single/double density, single/double sided disks, both 8″ and 5¼″ inch drives simultaneously . . . . . **PRICE $275.00**

■ LIGHTNING 286—80286 CPU BOARD
Offers 4 times the performance of a 5MHZ 8086 CPU while maintaining software compatibility . . . . . **PRICE $1395.00**

■ OCTAPORT 8 PORT SERIAL BOARD
0 to 19200 baud operation real time clock interrupt. Ideal for multi-user systems such as MP/M-86* **PRICE $395.00**

S100-PC-TM is a trademark of Lomas Data Products, Inc.
**CP/M-86, MP/M-86 and CONCURRENT CP/M-86 are trademarks of Digital Research. **MS-DOS is trademark of Microsoft.*
****Lightning One is trademark of Lomas Data Products, Inc.*



S100-PC-TM: The LDP Multi-user S100 Bus System offers high performance at a "low" price . . . plus, "our" system is expandable and upgradeable!

PRICE
An unbelievable **$6995.00**

Call today!

LDP

LOMAS DATA PRODUCTS, INC.
66 Hopkinton road, Westboro, MA 01581
Tel: (617) 366-6434 ☐ Telex: 4996272

Dealer inquiries invited.

For orders outside the U.S., contact our exclusive dealers: ☐ **Australia** — LAMRON PTY. LTD., (02) 85-6228
☐ **Malaysia** — EXA COMPUTER (M) SENDIRIAN BERHAD, 795284

CIRCLE 54 ON READER SERVICE CARD

The IIIrd Dimension

Writing programs in dBASE III

By Darryl Rubin

How do you repeat a stunning success? In an industry of one-product companies and technological brinkmanship, this is a multimillion dollar question.

One company, Ashton-Tate, seems to have at least part of the answer: build on your strengths.

In developing a new data base product, Ashton-Tate had a marvelous strength to build on—its dBASE II programming language. Ashton-Tate is no slouch at marketing products, but it was dBASE II's programmability that made it such a longstanding winner at the retail counter against its more sophisticated but nonprogrammable competitors.

Enter dBASE III. Not only is this whiz of a data base loaded with features customers asked for, it sports an enhanced language that brings data base programming into a new dimension.

In this article we'll look at the unique features of dBASE III's language and learn how to write programs using them. We'll also highlight what's changed since dBASE II and present two utility programs for your library.

You never know, we might even uncover some bugs and undocumented features. So if you're new to data base programming or an old dBASE II hand who's ready for a step up, read on.

To download copies of the software presented in this article, call the *COMPUTER LANGUAGE* Bulletin Board Service at (415) 957-9370 or CompuServe (GO CLM) and read the file DBASEIII.TOC.

II steps forward

If you're a dBASE II fan, you'll find a lot to like about dBASE III because much of the language has been carried forward, with welcome improvements.

dBASE III is an interpreted rather than a compiled language, though its syntax hardly suggests this. Procedures are symbolically named—no line numbers here—and the block-structured *DO WHILE* and *CASE* statements resemble PL/I's.

Data types include character string, numeric, logical, and a unique one, date. This major advance over dBASE II directly supports date comparison and arithmetic, as in *NEXTWEEK = TODAY + 7*. Two new functions, *DTOC* and *CTOD*, let you convert date variables between internal format and printable character strings.

dBASE III has room for up to 256 variables, way up from dBASE II's limit of 64. You can implicitly create or retype variables just by assigning them values. Fleet-fingered dBASE II users will certainly appreciate the new, terser syntax for assignment (*A = B* vs. *STORE A TO B*).

Unlike their dBASE II counterparts, variables in dBASE III are locally scoped. A subprogram inherits all its caller's variables and any new variables created by that subprogram disappear when it returns.

By introducing these new rules, Ashton-Tate has at once simplified modular programming and thrown a monkey wrench into every major dBASE II program ever written. This is because dBASE II variables are globally scoped and persist until explicitly released.

If you need global variables in dBASE III, declare them PUBLIC and they'll behave just like dBASE II variables. You can also declare variables PRIVATE, which has the effect of hiding any variables of the same name that already exist. This lets a subprogram define variables using any names it pleases without unintentionally modifying variables it inherits. (The hidden variables reappear with their values intact when the subprogram returns.)

Now let's evaluate a few expressions using dBASE III's BASIC-like print statement, *?*:

```
    ? 1 < 2 .AND. 2 < 1
.F.
    ? 'CON' + 'CAT'
CONCAT
    ? 'A' = 'ABC'
.F.
    ? 'ABC' = 'A'
.T.
```

Does it surprise you that *'ABC'* seems to equal *'A'* even though *'A'* doesn't equal *'ABC'*? This isn't a bug—really!—because dBASE doesn't compare strings for equality, it does prefix matching. In other words, it checks whether the second string matches the prefix of the first. This makes it a snap to program queries like "Display names starting with A" but beware: the result of a string comparison depends on the order of the strings.

If you prefer exactitude in your comparisons, you can *SET EXACT ON*. This statement also circumvents a particularly bothersome side effect of prefix matching, namely, that all strings compare equal to the null string.

dBASE III has more surprises in store, both good and bad, as we'll see. But first let's explore the part of the language that sets it apart from all others: those magical statements for manipulating data bases.

Tables for III

Like its illustrious predecessor, dBASE III is a relational data base. This means that it stores data in two-dimensional tables having a fixed number of columns and a variable number of rows.

In dBASE III, tables are called data base files, columns are called fields, and rows are called records. You students of relational algebra will know these same things as relations, attributes, and tuples, respectively. (And people have accused Ashton-Tate's manuals for being abstruse!)

Relational algebra defines three primitive operations that create new data bases from existing ones: *select*, *project*, and *join*. *Select* creates a new data base by copying selected rows from an existing one, while *project* copies selected columns. *Join* combines two data bases into one by concatenating records that have matching values in a common column.

Let's save *join* for later and see how dBASE III accomplishes *select* and *project*. We'll work with a sample data base called CATALOG.DBF that contains directory information from several diskettes of public domain software (Figure 1).

```
USE CATALOG
* Project two columns
COPY TO PROJECAT
     FIELDS NAME, SIZE

* Select certain rows
COPY TO SELECAT
     FOR VOL = 'DISKA'
     .AND. SIZE > 2048
```

Do you see how field names can be used in expressions just like variable names? This is true anywhere that dBASE III permits expressions.

You can simultaneously select and project a data base by including both the *FOR* and *FIELDS* clauses in the same *COPY TO*. In fact, one or both of these clauses can be used in almost all of dBASE III's data base statements, including *LIST, DISPLAY, SORT, REPLACE, APPEND, JOIN, COUNT, SUM, AVERAGE, TOTAL,* and *DELETE*. Amidst this diversity, selection and projection are anything but primitive operations. Here are some more examples:

```
* Count old files
COUNT FOR DATE
     < CTOD('12/01/84')

* Convert size to Kbytes
REPLACE ALL SIZE WITH
     INT(SIZE/1024)

* Sort on three fields
SORT TO SORTCAT ON
     NAME, EXT, DATE/D
```

If you're a dBASE II user, please try not to drool at how easily we just sorted a data base on three fields. dBASE III will let you sort on up to 10, with per-field options for ascending (/A), descending (/D), and ignore-case (/C) sort order. (Now you may drool.)

Meaningful relationships

So far we've seen how dBASE III lets you manipulate individual data base files. But the real power of relational data bases lies in their ability to relate information in different files based on common key fields.

Figure 2 shows a set of data base files you might use in conjunction with CATALOG.DBF to sell public domain software by mail order. Notice how key fields like BBSID, ORDERID, and FILEID are used to relate files so that data fields like CUST_NAME and BBS_NAME are never duplicated.

Compared to its predecessor, dBASE III's ability to relate files is positively incestuous. This is because you can now manipulate up to 10 files at once rather than two and create dynamic (run-time) relationships.

dBASE III gives you two ways to combine related information: with *JOIN* and

Catalog of public domain software

CATALOG.DBF

FILEID	Name	Ext	Vol	Path	Size	Date	Time	BBSid
0001	CATALOG	PRG	DISKA	\DB3	2667	11/03/84	10:02p	0003
0002	CALANDR	085	DISKB	\ART	8096	11/18/84	4:32a	0001
0003	BOZO	PIC	DISKB	\ART	9999	11/19/84	5:20a	0001
0004	123	ZAP	DISKC	\HMM	456	12/23/84	11:19p	0002
0005	WONDER	PRG	DISKA	\DB3	1024	12/23/84	12:02a	0003

Figure 1.

SET RELATION TO. As we explained in the last section, *JOIN* physically concatenates the records from two data bases based on matching values in a common field. The idea behind *SET RELATION TO* is similar, but rather than physically concatenating records, this statement dynamically links them at run time (a much more efficient process).

Here's an example that uses *JOIN* to concatenate ORDERS and ITEMS records having the same ORDERID; the resulting DETAILS data base would tell you, for example, what files each customer ordered.

```
USE ITEMS
SELECT 2
USE ORDERS
JOIN WITH ITEMS TO
    DETAILS FOR ORDERID=
    ITEMS->ORDERID
```

Notice how the *SELECT* statement differs from dBASE II's. You now specify the work area to select as a number from 1 to 10 or a letter A to J. Once you've opened a file in a work area, you can also specify its name on the *SELECT* statement rather than the work area ID.

The other thing to notice is how the new pointer notation (->) lets you refer to fields in other work areas. Reminds you of PL/I pointers, doesn't it? The same notation comes in handy for manipulating files linked by *SET RELATION TO*. For example, it's a cinch to relate several files and list selected information from each:

```
SELECT 3
USE CUSTOMER
SELECT ORDERS
SET RELATION TO
    CUSTID INTO CUSTOMER

SELECT ITEMS
SET RELATION TO
    ORDERID INTO ORDERS
LIST FIELDS AMT_PAID,
    CUSTOMER->CUST_NAME,
    ORDERS->ORDER_DATE
```

Deleting a relationship is just as easy: just *SELECT* the source file and say *SET RELATION TO* with no parameters. In dBASE III, breaking up isn't hard to do!

Command performance

Writing programs in dBASE III isn't hard to do, either, because it's an interpreter. Just type your program into a text file with a name of your choosing and the extension .PRG, then say *DO <NAME>* to run it. Try this simple example, which you should save as REPT.PRG:

```
PARAMETERS CH,N,STROUT
SET TALK OFF
DO WHILE N > 0
    STROUT = STROUT + CH
    N = N - 1
ENDDO
```

```
Data bases related on key fields

CUSTOMER.DBF                                    BBS.DBF
|         |                                     |     |
|CUSTID   Cust_name          Address       |    |BBSID    BBS_name    Phone   |
|========================================= |    |================================|
|0001     Fred Flintstone    Boulder, CO   |    |0001     Fun Spot    123-4567|
|0002     Nowhere Man        Nowhere Land  |    |0002     Techie      314-1592|
|                                          |    |0003     Nerdmicro   555-5555|
|                                          |    |                             |

ORDERS.DBF                                      ITEMS.DBF
|         |                                     |         |
|ORDERID  Custid   Order_date    Ship_date|     |ORDERID   FILEID    Amt_paid|
|=========================================|     |============================|
|0001     0001     12/18/84      12/19/84 |     |0001      0002      0.01    |
|0002     0002     12/20/84       1/03/85 |     |0001      0003      0.01    |
|0003     0002     12/23/84       1/03/85 |     |0002      0002      1.00    |
|0004     0001      1/02/85       1/03/85 |     |0003      0001      0.50    |
|                                         |     |0004      0001      0.00    |

Note:  Field names in ALL CAPS designate key fields.
```

Figure 2.

You've just added a new command to dBASE III that builds repeated character strings. For example, *DO REPT WITH* '-',5,X will add five dashes to a character variable *X*.

This little example shows off one of dBASE III's towering improvements over dBASE II: the ability to pass parameters to a command file. Banish forevermore the clumsy and error-prone method of passing arguments through global variables.

As if one towering improvement weren't enough, here's another: dBASE III has procedure libraries. This means you can put up to 32 commands into a single .PRG file. Surround each one with *PROCEDURE/RETURN* statements as follows and you're all set:

```
PROCEDURE FOO
     <statements>
RETURN
```

To use a procedure file, just *SET PROCEDURE TO <FILE NAME>* and *DO* the procedures like any other commands. You'll find that procedures run much faster than command files. You'll also find they have an undocumented and most welcome property—they're recursive. (Just be sure to declare any local variables as PRIVATE.) One of the programs we present later puts this discovery to excellent use.

Speaking of unexpected properties, try this experiment: write a procedure that takes two arguments and call it with a duplicated variable, as in *DO REPT WITH X,5,X*. dBASE III will complain "Variable not found." Not found? What do you mean not found? I passed it to you twice!

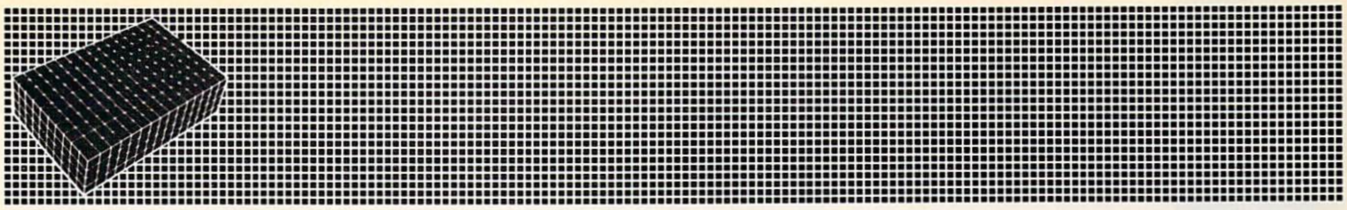
Diskette jockey

If you liked the idea of having a diskette directory like CATALOG.DBF, do I have a program for you (Listing 1). It's called

```
CATALOG.PRG -- Program to build catalog data bases

* Structures of data bases used by this program:
* CATSPEC: LINE - C 80
* CATWORK: NAME - C 9; EXT - C 4; SIZE - C 10; DATE, TIME - C 8.
* CATNEW:  Like CATWORK, but DATE is D and add VOL - C 11, PATH - C 29
*
parameters pathspec,catfile
close databases
select 1
use catspec
set talk off
set safety off
zap
*1: Call DOS to put the desired directory into file CATDIR.TXT
if file('catdir.txt')
  erase catdir.txt
endif
cmd = 'dir ' + pathspec + '>catdir.txt'
run &cmd
*2: Specified path bad if CATDIR.TXT not found or < 3 records long.
if file('catdir.txt')
  append from catdir.txt for recno() <= 3 sdf
  if recno() >= 3
    goto 2
*3: Path is OK, extract Volume and Path names from dir listing
    voln = upper(trim(substr(line,23,11)))
    goto 3
    pathn = upper(trim(substr(line,at('\',line),29)))
```

Listing 1 *(Continued on following page).*



CATALOG.DBF and I'll bet you can guess what it does.

Try *DO CATALOG WITH 'A:','MYCAT'*, then *USE MYCAT* and manipulate it with the data base statements we discussed earlier. You'll go where no directory has gone before.

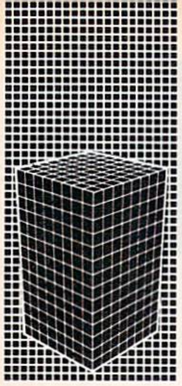
CATALOG.PRG shows off some of dBASE III's hottest new features. Foremost among these is *RUN*, which passes commands to PC-DOS. CATALOG.PRG uses this to list the desired directory into a file called DIR.TXT (see *1). The program then uses *APPEND FOR RECNO() < = 3* to read the first three records of DIR.TXT into a data base called CATSPEC (*2) from which the volume and path names are extracted for assignment to *VOLN* and *PATHN*.

Neat but that's just the beginning. CATALOG.PRG next uses another data base, CATTEMP, to pull in the whole DIR.TXT file (*4). Although most of this is the directory information we're looking for, a few records have other stuff and need to be discarded. What would be a chore in dBASE II is child's play in dBASE III, thanks to the new statement *SET FILTER TO*.

Remember *SET RELATION TO*, which dynamically joins records? Well *SET FILTER TO* dynamically selects them, based on a specified filter condition. Records not satisfying the condition become totally invisible to the accessing program. By saying *SET FILTER TO SIZE > 0*, CATALOG.PRG deftly banishes the unwanted records from the directory listing.

```
*4: Now prepare a temp data base and read the directory into it
    copy file catwork.dbf to cattemp.dbf
    use cattemp alias temp
    append from catdir.txt sdf
    set filter to size > 0
    select 2
*5: If the specified catalog doesn't exist, create it.
    if .not. file('&catfile..dbf')
      copy file catnew.dbf to &catfile..dbf
    endif
    use &catfile
    goto bottom
    select temp
    goto top
*6: Read cattemp into catalog, reformatting date & adding vol/path
    do while .not. eof()
      select &catfile
      append blank
      replace name with temp->name, ext with temp->ext,          ;
              size with temp->size, date with ctod(temp->date), ;
              time with substr(temp->time,3), vol with voln,    ;
              path with pathn
      select temp
      skip
    enddo
  endif
endif
close databases
erase catdir.txt
erase cattemp.dbf
set safety on
set talk on
```

Listing 1 *(Continued from preceding page).*



Macro economics

Have you noticed the strange *&CMD* and *&CATFILE* constructs below *1, *5, and *6 in Listing 1? You old-time dBASE II'ers should be smiling out of your seats by now, because here we talk not about magic but about wizardry.

About macros.

Alas, in this one area dBASE III grovels before its ancestor.

Mundanely speaking, a macro is a character variable prefaced by an ampersand, like *&CATFILE*. Seems harmless enough but so does a chain reaction at first glance.

What dBASE does with macros is deceptively simple. Before interpreting a command line, dBASE scans it for macros, replacing any it finds with their assigned string values. That's it. But suddenly the command line has changed.

dBASE II would repeat this process, looking for nested macros, scanning and substituting again and again until no more macros materialized. A chain reaction indeed. This process could turn lead into gold!

Not so with dBASE III. In the name of performance, it quits after one scan. Say good-bye to recursive macros. Still, even one-level macros have their uses.

For example, CATALOG.PRG uses the macro *&CATFILE* to reference the catalog name you pass as a parameter. If CATFILE equals 'MYCAT', *USE &CATFILE* becomes *USE MYCAT*. This is why the *USE* statement near *5 opens the file you specify rather than a file called CATFILE, as it would without the ampersand.

Macros also let you do math on character variables or change their type to numeric:

```
N = '1'
N = STR(&N+6)  (= '7')
N = &N         (= 7)
```

Macros can even turn memory variables into arrays, a data structure that dBASE III otherwise lacks. For example, create a set of variables *A_1*, *A_2*, *A_3*, and *I*. Assign *I* = '1' then *I* = '2' then *I* = '3'. For each value of *I* what is the meaning of *A_&I*? *A_1* and *A_2* and *A_3*.

Here are some more time-saving tricks you can play with macros:

```
*Show a centered string
&CENTER =
    '80 - LEN(TRIM(PROMPT))'
PROMPT = 'ANY STRING'
@24,&CENTER SAY PROMPT

*Abbreviate a command
&CMD = 'DIR *.PRG'
&CMD
```

Macros are also the source of a small mystery. The dBASE III manual makes an elaborate show of explaining how Ashton-Tate disabled the use of macros in the condition clause of *DO WHILE* statements in exchange for a 30% speedup. How odd. When I tried them, *DO WHILE* macros worked every which way. Maybe there really is magic here.

Screen gems

And now for our starring attraction, the program that pulls it all together: data base access, macros, recursive procedures, full screen I/O, even a guest appearance by an undocumented feature.

This program—a pair of procedures, really—is called MENUSYS.PRG (Listing 2). It implements a surprisingly compact yet fancy menu system. The menus it draws have a highlighted selection bar you can move up and down with F3/F4 or skip right to an item by typing its first character. Help text for the highlighted item is shown centered on line 25. To select an item hit Enter, and your wish becomes dBASE III's command.

You might like to try this out using the demo menus I uploaded to the *COMPUTER LANGUAGE* BBS. Just type *SET PROCEDURE TO MENUSYS* followed by *DO MENU WITH 'MENUDEMO'*.

A menu is a .DBF file having up to 23 records, one for each screen row. Each record in the menu file has three fields: the ITEM to display, the associated HELP text for line 25, and the COMMAND to execute when the item is selected.

COMMANDs can be anything dBASE III would understand at the dot prompt, except they mustn't contain macros. If you leave a COMMAND field blank, the associated ITEM will be treated as a title—the selection bar will automatically skip over it.

Can one menu call another? Of course—just use *DO MENU <NAME>* for the COMMAND. The menu system will call itself recursively, and when the recursive invocation returns, the original menu will be redisplayed. (Menus return when the user hits F2 or selects a menu ITEM whose COMMAND is *RETURN*.)

Like CATALOG.PRG, MENUSYS puts *SET FILTER* to good use (*0). It filters based on *TRIM(COMMAND) <> "*, which hides records with blank COMMAND fields. The result? We don't need special logic to make the record seeking commands *SKIP*, *LOCATE*, and *GOTO* bypass menu titles.

Macros are also put to good use, as you can see right below *5, where the COMMAND field is assigned to the variable *EXEC* for execution via *&EXEC*. (Sorry, field names can't themselves be macros.)

Now for that undocumented feature: the *WAIT TO* statement that inputs a keystroke to a character variable is capable of reading certain function keys, including backspace and F2 through F10. These keystrokes produce the respective character values CHR(127) and CHR(254) down to CHR(246). This is the secret behind MENU.PRG's testing for F3 and F4 at *3 and *4 in the listing.

Third base

Undoubtedly, many other features lie unknown in dBASE III, awaiting discovery. So too must many bugs. Like its predecessor, dBASE III may become known as much for its quirks and secret features as it will for its scope and programming power.

But scope and power it has. What we've seen here is only a slice through a structure of much broader dimensions. Although we can't be sure what dBASE III is destined to become known for, one thing seems certain: it is a base to build on.

Darryl Rubin is section manager for network products at ROLM Corp.

```
MENUSYS.PRG -- Interactive menu system

procedure drawmenu
  use &menufile
  set talk off
  set exact on
  set filter to
  goto top
  clear
  do while .not. eof() .and. recno() <= 23
    @recno()-1,20 say trim(item)
    skip
  enddo
  set filter to trim(command) <> ''
return
*0
procedure menu
* Presents a menu from a specified menu file.  Menu files have
* structure ITEM (C 38), HELP (C 40), COMMAND (C 80).
  parameters menufile
  private first, last, exec, lastpos
  do drawmenu
*1: Determine screen rows of first and last selectable items
  goto bottom
  do while recno() > 23
    skip -1
  enddo
  last = recno()
  goto top
  first = recno()
  key = chr(255)
  do while key <> chr(254)
*2: Highlight the currently selected item and get next keystroke
    @recno()-1,20
    selectn = trim(item)
    set color to 7+
    @recno()-1,20 say selectn
    @24,0
    @24,(80-len(trim(help)))/2 say help
    set color to 7
    set console off
    wait to key
    set console on
    @recno()-1,20
    @recno()-1,20 say item
```

Listing 2 *(Continued on following page).*

```
    @24,0
    do case
      case key = chr(252)
*3: User hit F4.  Select next item.
        if recno() < last
          skip 1
          if recno() > last
            goto first
          endif
        else
          goto first
        endif
      case key = chr(253)
*4: User hit F3.  Select previous item.
        if recno() > first
          skip -1
          if recno() < first
            goto last
          endif
        else
          goto last
        endif
      case len(key) = 0
*5: User hit enter or other extended key.  Exec the COMMAND field.
        clear
        exec = command
        lastpos = recno()
        set exact off
        &exec
*6: We're back, now pause if not returning from another menu.
        if .not. upper(trim(substr(exec,1,8))) $ 'DO MENU HELPASSIST'
          ?
          wait to key
        endif
        do drawmenu
        goto lastpos
        key = ' '
      otherwise
*7: User hit some other key, skip to matching menu item, if any
        lastpos = recno()
        locate for substr(item,1,1) = key
        if eof()
          goto lastpos
        endif
    endcase
  enddo
*8: All done, restore settings and quit
  clear
  set exact off
  set talk on
return
```

Listing 2 *(Continued from preceding page).*

Extensibility in Forth

By Michael Ham

Forth is extensible—that is, programmers can easily add commands to the language. If the job at hand requires some basic tools that Forth lacks, the programmer creates them and they become part of the language, either permanently or for that application.

Programmers who do not use extensible languages may not realize the power that extensibility confers. This article shows an example of using Forth's extensibility to create new commands that make it possible to use bit arrays.

An array is a string of "boxes"—often bytes or cells—in which a program can store data. For an example of the use of an array, consider a mailing list on diskette where each record must contain the state of the mailing address. Since Forth can use a single byte to store numeric values up through 255, you can save room on the diskette by storing a numeric state code, from 00 through 50, in one byte instead of using the two-letter state abbreviation, which would require two bytes. But when the list is printed, the numeric code must be replaced by the abbreviation: IA, CA, etc.

An easy way to do the conversion is to include in the print program an array of two-byte cells, each cell containing one of the two-letter abbreviations for the 50 states or the District of Columbia. Suppose the array was initially defined by

```
CREATE STATES 102 ALLOT
```

When the word *STATES* is executed, it puts on the stack the address of the first cell of the array. Then by doubling the state code from the record (because each abbreviation occupies two bytes) and adding the result to the top of the stack, you can retrieve the state abbreviation.

For example, the following word, given the state code, will display the state abbreviation: *15 STATE* displays IA.

```
: STATE  ( n – ) 2* STATES + 2 TYPE ;
```

If you wanted to use the complete state name, the array would consist of larger entries, each 15 or so bytes long. It is much easier to retrieve data from an array when every element is the same length, even if you must pad short names with blanks so that they take the same number of bytes as the longer names.

If every entry is the same length, you merely multiply *i* by the length of an entry to go directly to the *i*th entry. But if the entries of an array vary in length, perhaps with the end of each entry marked by some special character, then to pick out the *i*th entry you must do one of two things. Either you have to read every entry until the *i*th one, counting your way through the array, or else you must maintain a collection of pointers, with the *i*th pointer containing the address of the beginning of the *i*th element of the array.

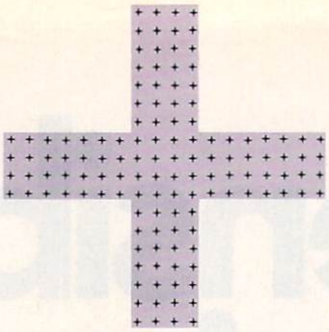
Note that the pointers are themselves kept in an array of their own, with each entry the same length, so that you can readily retrieve the *i*th pointer.

Two-dimensional arrays are also used. Two numbers—the row and the column—are used to locate an entry. Two-dimensional arrays require as many bytes as the product of the number of rows, the number of columns, and the number of bytes in each entry. This product tends to be large. Arrays of three and four dimensions take correspondingly more room.

Even one-dimensional arrays can take up a lot of room. Recently I wrote a program in which an array was used to show which of 160 possible characteristics an organization possessed. I could have used an array of 160 bytes in the organization record, with each byte either Y (if the organization had the characteristic) or N (if it did not). But the file consisted of 3,700 organizations; at 160 bytes each, these arrays would take 592,000 bytes, which exceeded the capacity of the diskette. The diskette had to hold not only the arrays, but also other data about the organizations and computer program.

The solution was to use an array of bits rather than bytes. Since each characteristic was either present or not, I could assign each characteristic a single bit position in the array. The bit was 1 if the organization possessed the characteristic, 0 if it did not.

Eight characteristics could then be recorded in a single byte, and an organization's entire list of 160 characteristics



would occupy only 20 bytes—or only 74,000 bytes for the entire file. The problem was that Forth has no commands to read or store individual bits. The solution? Add such commands to Forth for this application.

Logic operators

To create commands that make it easy to use a bit array, you use Forth's bit operators *AND*, *OR*, and *NOT* or *XOR*. In playing with these words to see how they work, it would be nice to have a command that displays the bit structure of the number on top of the stack. Forth doesn't have that command either, but the lack is readily remedied. The word *.B* will display the bit structure of the top of the stack:

```
: .B ( n -- n ) BASE @    2 BASE ! OVER
        BASE ! ;
```

BASE is a variable that holds the current base; executing *BASE* puts on the stack the address of the variable. @ (pronounced fetch) replaces the address on the stack with the contents of the address. *!* (pronounced store) takes what is second on the stack and puts it at the address that is on top of the stack. *OVER* puts on top of the stack a copy of what was second on the stack, and . (pronounced dot) takes the top of the stack and displays it on the screen as a numeral in the current base.

The original contents of *BASE* are then restored. The comment in parentheses shows that the effect of this word is to leave the stack undisturbed.

The binary (base 2) representation of a number shows its bit pattern, with 1 being a bit that's on and 0 a bit that's off. The high-order (leftmost) zeros are not displayed. Thus a byte with only bit 3 on would be displayed as decimal 8 or binary 1000. The three zeros to the right of this 1 represent bits 0, 1, and 2 (reading from right to left), and the high-order bits (bits 4, 5, 6, and 7) are not displayed since they are also zero.

Note that I number the bits with the low-order (rightmost or least-significant) bit in each byte being bit 0 and the high-order (leftmost or most-significant) bit being bit 7. This convention is common but by no means universal.

Experiment by entering numbers and using *.B* to see the resulting bit patterns. You will see that the number 0 consists of all bits off and −1 of all bits on (16 bits will be shown since each stack entry is a cell that is two bytes wide).

The effects of the bit operators, shown here as acting on bytes, is shown in Figure 1. The *NOT* operation shown is the Forth-83 Standard *NOT*. It can be simulated in earlier Forths by using the phrase *−1 XOR* or *NEGATE 1−*.

As you can see, *NOT* operates on only one operand (the top of the stack) and simply flips each bit: 0 becomes 1 and 1 becomes 0. *AND*, *OR*, and *XOR* do a position-by-position comparison of bits, and each bit of the result is determined by the corresponding two input bits:

- *AND*—result is 1 if and only if both input bits are 1
- *OR*—result is 1 if and only if at least one input bit is 1
- *XOR*—result is 1 if and only if exactly one input bit is 1.

These words are used to construct commands that can set and fetch individual bits. I also needed to be able to toggle a bit (turn it on if it was off or off if it was on) without having to know its initial state. This capability was needed so the user could indicate a given characteristic in the selection list and the program would turn it on if it were off (meaning the user was selecting that characteristic) or turn it off if it were on (meaning that the user had previously selected it and was now "un"selecting it).

Operands:	11011001 00111000	11011001 00111010	11011001 00111010	11011001
Operator:	AND	OR	XOR	NOT
Result:	00011000	11111011	11100011	00100110

Figure 1.

```
 ( Bit Operators                                      M Ham
14Jul84)
 FORTH-83    DECIMAL
 CREATE BITS  1 C, 2 C, 4 C, 8 C, 16 C, 32 C, 64 C, 128 C,

 : MASK  ( bit# - mask )  BITS +  C@ ;

 : +BIT  ( bit# adr - )  SWAP MASK OVER C@  OR SWAP C! ;

 : -BIT  ( bit# adr - )  SWAP MASK  NOT  OVER C@ AND SWAP C! ;

 : @BIT  ( bit# adr - ? )  C@ SWAP MASK AND ;

 : ~BIT  ( bit# adr - )  2DUP @BIT IF  -BIT  ELSE  +BIT THEN ;

 : ~BIT&FLAG  ( bit# adr - ? )  2DUP ~BIT @BIT ;

```

Figure 2.

The list of new commands is shown in Figure 2. These definitions take a total of 160 bytes of memory, using Laboratory Microsystems' PC/Forth version 3.0.

FORTH-83 will execute (but do nothing else) in systems that satisfy the 83 Standards. If *FORTH-83* fails to execute, then the Forth is not an 83 Standard Forth and the *NOT* in line 8 will probably have to be replaced by *−1 XOR*. The word *DECIMAL* executes to define the appropriate base for the numbers that are stored into the *BITS* array in the second line of the listing. The command *C,* puts into the Forth dictionary the byte that is on top of the stack.

MASK accepts a bit number and executes *BITS* to put on the stack the address of the first byte of the *BITS* array. The command *C@* (C fetch) then replaces the address with the byte's contents. Thus *MASK* accepts a bit number and puts on the stack a cell with all bits off except for a specified bit in the low-order byte. For example, *3 MASK* provides a byte with bit 3 as 1, all other bits as 0. (In this application I work with bytes rather than cells, which are two bytes wide.) Notice that *MASK* itself uses an array, in which each byte has a single bit turned on.

In operating on specified bits at specified addresses, I decided that it was most comfortable to present first the bit number and then the address of the byte, on the analogy of the way *!* works: it expects the value to be entered first and then the address, so the address is on top of the stack. Similarly, *+BIT* (turn bit on) expects bit number followed by address even though *+BIT* then swaps the two stack entries.

+BIT stores a 1 bit at a specified address. By using *MASK*, a byte is obtained with (only) the specified bit set to 1, and *OR* combines that byte with the byte brought from the specified address; this produces a byte with all bits left as they were except that the specified bit in the result must be a 1 (since the specified bit in the byte created by *MASK* is a 1). The resulting byte is then stored back at the address by *C!*.

−BIT (turn bit off) uses a similar procedure to store a 0 bit at a specified address. In this case, however, the bits in the byte provided by *MASK* are reversed by using *NOT* (or, if your Forth is not 83 Standard, by using instead the phrase *−1 XOR*).

The result is a byte that consists of all 1 bits except for the specified bit, which is 0. The effect of using *AND* to combine this resulting byte with the byte from the specified address is to leave all bits as they were except for the specified bit: the 0 bit forces the result of the *AND* to also have a zero bit in that location.

@BIT first uses *MASK* to get a byte with the bit of interest set to 1 and then uses *AND* to combine that byte with the byte fetched from the address. The resulting flag will be zero if the specified bit at the address is 0—otherwise the result will be nonzero (either 1, 2, 4, 8, 16, 32, 64, or 128, depending on which bit was specified). This nonzero value will function as a true flag in any of the logical tests, such as *IF*, since Forth treats zero flags as false and nonzero as true. The other bits in the specified byte have no effect on the result since they are wiped out by the 0 bits in the *MASK* byte.

~BIT (toggle bit) first uses *2DUP* to make a second copy of the bit number and address. It uses one copy to fetch the given bit and the other copy to store the result of the toggle. If the fetched bit was 1, *~BIT* stores 0 in its place, or if it was 0, stores 1 in its place. The effect is to toggle the bit. *~BIT&FLAG* does the same thing except that it leaves a flag (zero or nonzero) on the stack to show the result of the toggle.

These are the elemental commands that can be used to define a variety of array commands. I have made the elemental commands a permanent part of my Forth, but the words defined with them vary by application.

For example, suppose the array I mentioned was created with the name *TYPES*

ATTENTION: ENGINEERS PROGRAMMERS

PolyFORTH® II

the operating system and programming language for real-time applications involving ROBOTICS, INSTRUMENTATION, PROCESS CONTROL, GRAPHICS and more, is now available for . . .

IBM PC*

PolyFORTH II offers IBM PC users:

- **Unlimited control tasks**
- **Multi-user capability**
- **8087 mathematics co-processor support**
- **Reduced application development time**
- **High speed interrupt handling**

Now included at no extra cost: Extensive interactive **GRAPHICS SOFTWARE PACKAGE!** Reputed to be the fastest graphic package and the only one to run in a true multi-tasking environment, it offers point and line plotting, graphics shape primitives and interactive cursor control.

PolyFORTH II is fully supported by FORTH, Inc.'s:

- **Extensive on-line documentation**
- **Complete set of manuals**
- **Programming courses**
- **The FORTH, Inc. hot line**
- **Expert contract programming and consulting services**

From FORTH, Inc., the inventors of FORTH, serving professional programmers for over a decade.

Also available for other popular mini and micro computers.

For more information contact:

FORTH, Inc.

2309 Pacific Coast Hwy.
Hermosa Beach,
CA 90254
213/372-8493
RCA TELEX: 275182

Eastern Sales Office
1300 N. 17th St.
Arlington, VA 22209
703/525-7778

*IBM PC is a registered trademark of International Business Machines Corp.

CIRCLE 30 ON READER SERVICE CARD

NEW FEATURES

(Free update for our early customers!)

- Edit & Load multiple memory resident files.
- Complete 8087 assembler mnemonics.
- High level 8087 support. Full range transcendentals (tan, sin, cos, arctan, logs and exponentials) Data type conversion and I/O formatting.
- High level interrupt support. Execute Forth words from within machine code primitives.
- 80186 Assembler extensions for Tandy 2000, etc.
- Video/Graphics interface for Data General Desktop Model 10

HS/FORTH

- Fully Optimized & Tested for:
  IBM-PC IBM-XT IBM-JR
  COMPAQ EAGLE-PC-2
  TANDY 2000 CORONA
  LEADING EDGE
  (Identical version runs on almost all MSDOS compatibles!)
- Graphics & Text (including windowed scrolling)
- Music - foreground and background includes multi-tasking example
- Includes Forth-79 and Forth-83
- File and/or Screen interfaces
- Segment Management Support
- Full megabyte - programs or data
- Complete Assembler (interactive, easy to use & learn)
- Compare
  BYTE Sieve Benchmark jan 83
  HS/FORTH 47 sec BASIC 2000 sec
  w/AUTO-OPT 9 sec Assembler 5 sec
  other Forths (mostly 64k) 70-140 sec

FASTEST FORTH SYSTEM AVAILABLE.

TWICE AS FAST AS OTHER FULL MEGABYTE FORTHS!

(TEN TIMES FASTER WHEN USING AUTO-OPT!)

HS/FORTH, complete system only: $250.

Visa Mastercard

Add $10. shipping and handling

HARVARD SOFTWORKS

PO BOX 2579
SPRINGFIELD, OH 45501
(513) 390-2087

CIRCLE 47 ON READER SERVICE CARD

since each characteristic defines a type of organization. Then for a given characteristic you need to obtain the precise bit number and address in the array. Assuming the characteristics are identified by sequential numbers beginning with zero and the array is in memory, you can define the command *AIM*:

```
: AIM    ( c# -- bit# adr )   8  /MOD
    TYPES + ;
```

AIM uses */MOD* to divide the characteristic number by 8, so the quotient (on top of the stack) is the byte number within the array and the remainder (underneath) is the bit number within the byte. *TYPES* is executed, which puts on top of the stack the beginning of the array, and this is simply added to the quotient from the */MOD*.

The result is the address of the byte on top of the stack and the bit number directly beneath, which is exactly the configuration needed by my bit-manipulation commands. The stack diagram comment within the parentheses documents the behavior of the command.

Then, to fetch the flag for a given characteristic, simply put on the stack the number indicating which characteristic and execute *@TYPE* (fetch type), as defined below, and the result will be a flag (zero or nonzero—shown in the stack diagram as "?") that will indicate whether the particular characteristic is present (i.e., the bit is 1) or not. For example, to see if characteristic 37 is present, use the phrase *37 @TYPE* and the stack will be topped by a true or false flag that tells whether that bit is set or not.

```
: @TYPE ( c# -- ? ) AIM @BIT ;
```

To store a characteristic, on the other hand, you need only put the characteristic number on the stack and execute *+TYPE*, as defined below. If you wanted to set characteristic 113 on, then you would use the phrase *113 +TYPE*.

```
: +TYPE ( c# -- ) AIM +BIT ;
```

Other words can be similarly defined to turn off the bit for a given characteristic (*−TYPE*), toggle the bit for a given characteristic (*~TYPE*), and so on.

The preceding definitions assume that *TYPES* is an array in memory. What if you instead want to work with the array as stored in the individual organization records on diskette? That can easily be done by changing the definition for *AIM*.

Forth typically works with 1K blocks on the diskette, using the absolute block number to reference the block. A 320K diskette thus consists of 320 blocks.

Suppose each organization record is 64 bytes long and the records are sequentially stored beginning with organization number 0 at the start of block 165, with 16 organizations per block. Suppose further that the array begins in the 20th byte of each record. *AIM* would then be defined in this way:

```
: AIM ( c# org#—bit# adr ) 16 /MOD
    165 + BLOCK
    SWAP 64 * + 20 + SWAP 8
    /MOD ROT + ;
```

This version of *AIM* expects *two* numbers on the stack: the characteristic number and the organization number. The first phrase (*16 /MOD 165 + BLOCK*) divides the organization number by 16 to get the relative block number for the block this organization record is in and adds 165 to it (since the organization records begin in block 165) to get the absolute block number. *BLOCK* accepts a block number and leaves in its place the address of the beginning of the block. (*BLOCK* reads the diskette if necessary so the block is in memory and the address is simply a memory address.)

The second phrase swaps the block address and the remainder from the */MOD* division. This remainder tells us which record within the block is the record for this particular organization. We multiply that record number (within the block) by 64 (since there are 64 characters per record) and then add this product to the address of the beginning of the block to get the address of the first byte of the organization record. Then 20 is added to that sum to produce the address of the first byte of the array for the organization we are interested in.

The final phrase swaps this address and the characteristic number to put the latter on top of the stack and then calculates the byte offset into the array. The *ROT* puts the earlier address on the top of the stack so the offset can be added to it.

Exactly as in the earlier *AIM*, we are left with the byte address on top of the stack and the bit number beneath it. This definition may seem somewhat complex (though it is a pattern that quickly becomes familiar), but it allows us to treat arrays on the diskette exactly as if they were in memory. All the complexity is hidden within the new command *AIM*.

You now can define *@TYPE*, *+TYPE*, *−TYPE*, *+TYPE*, and so on, exactly as described, except for two minor changes.

First, the stack diagram comment in the definition (which shows what values the definition expects) should be changed to show two values instead of one on the input side: characteristic number and organization number.

Second, the words that alter the contents of the block, such as *+TYPE*, *−TYPE*, and *~TYPE*, must include the word *UPDATE* and the end of the definition, or they must be followed by *UPDATE* when they are executed. *UPDATE* flags the block as being updated, which ensures that the revised block will be written back to diskette.

With these exceptions the new definitions of these commands are word for word the same as the old ones because we hid in *AIM* all the work of setting up the array.

Of course, if you are working with arrays in memory as well as on diskette, you will want two sets of commands with different (though probably related) names—one set for each type of array. Or you can use the same names for both types of commands by using a Forth technique known as vectored execution.

In this technique you define two words—say *DISKETTE* and *MEMORY*—that control the array commands. *DISKETTE* will set the commands to work on diskette arrays, and they will be diskette commands until *MEMORY* is executed, whereupon they will revert to being memory array commands. But vectored execution is a topic for another article.

Bit arrays work well if each entry has only two values: on/off, yes/no, true/false, male/female, red/green, left/right, big/little, new/used, on hand/out of stock, and so on. If the range of values is greater than two but doesn't exceed four, you can use two bits to cover the range (00, 01, 10, and 11—decimal values 0, 1, 2, and 3) and thus use a single byte to hold four entries. Similarly, you can use four bits to cover a range of 16 or fewer values, packing two entries per byte.

The ease with which you can create high-level commands that permit individual bit manipulation is an illustration of the power of extensibility. A language is extensible if it is easy for the programmer to add commands to the language. Because most programming languages are not extensible, the criterion of extensibility is seldom included among the selection criteria for a programming language.

Forth's extensibility is especially powerful because the Forth compiler consists of commands that are a part of Forth (for example, the *:* and the *;* in the definitions above). This means that even new compiler commands can be created and added to the language so that the Forth compiler is itself extensible.

This example of bit-array commands illustrates why programmers who have used an extensible language value that extensibility so highly. Extensibility is a basic and essential characteristic for a language; every spoken language is extensible, and after using an extensible language, programmers want never to go back to more limited languages. ■

Michael Ham has worked in program and systems design, development, and documentation for many years. He currently is a free-lance programmer and technical writer/editor residing in Santa Cruz, Calif.

FOR TRS-80 MODELS 1, 3 & 4
IBM PC, XT, AND COMPAQ

THREE TOUGH QUESTIONS
WITH ONE EASY ANSWER:

1. **WHEN IS A COMPUTER LANGUAGE NOT A LANGUAGE?**
   MMSFORTH includes DOS, Assembler and high level commands and extraordinary utilities, extends to become any other language (or application), is an interpreter and a compiler, and is remarkably fast and compact!
2. **WHICH SOFTWARE RUNS THE SAME DISKS IN IBM PC AND TRS-80 MODEL 4?**
   MMSFORTH disks run on those and Compaq, and TRS-80 Model 3, and Tandy 1200, and TRS-80 Model 1, and AT&T 6300, etc., with your choice of formats up to 200K single-sided or 400K double-sided!
3. **WHO OFFERS SOURCE CODE WITH ITS LANGUAGE, UTILITIES, DATABASE, WORD PROCESSOR AND COMMUNICATIONS SOFTWARE?**
   Nearly all MMSFORTH software includes source code.

MMSFORTH

All the software your computer may ever need.

The total software environment for IBM PC, TRS-80 Model 1, 3, 4 and close friends.

- Personal License (required):

MMSFORTH System Disk (IBM PC)	$249.95
MMSFORTH System Disk (TRS-80 1, 3 or 4)	129.95

- Personal License (optional modules):

FORTHCOM communications module	$ 39.95
UTILITIES	39.95
GAMES	39.95
EXPERT-2 expert system	69.95
DATAHANDLER	59.95
DATAHANDLER-PLUS (PC only, 128K req.)	99.95
FORTHWRITE word processor	175.00

- Corporate Site License Extensions . . . . **from $1,000**
- Some recommended Forth books:

UNDERSTANDING FORTH (overview) . . .	$ 2.95
STARTING FORTH (programming)	18.95
THINKING FORTH (technique)	15.95
BEGINNING FORTH (re MMSFORTH) . . .	16.95

Shipping/handling & tax extra. No returns on software.

Ask your dealer to show you the world of MMSFORTH, or request our free brochure.

MILLER MICROCOMPUTER SERVICES
61 Lake Shore Road, Natick, MA 01760
(617) 653-6136

CIRCLE 61 ON READER SERVICE CARD

Use ALL the Power of Your MS-DOS, IBM PC-DOS, or CP/M-80 System with UNIX-Style Carousel Tools

```
ch "CP/M" "MS-DOS" <doc >newdoc
diff newdoc doc | more
ed newdoc
kwic newdoc | sortmrg | uniq | unrot >index
make -f makdoc ndx
```

CAROUSEL TOOLS are a proven set of over 50 programs designed to be used with pipes, redirected I/O and scripts. In the style of UNIX each Tool does one thing well, and the Tools can be used together to do more complex tasks.

YOU ACCOMPLISH MORE using Carousel Tools: better programming and documentation support, simpler data and file housekeeping, more general file handling.

TOOLS FOR PC/MS-DOS 2.x AND CP/M-80 are available now. The DOS ToolKit is $149. The CP/M ToolKit is $249 and includes a *shell* to provide pipes, redirected I/O, and scripts. Source code is available for $100 more.

ORDER YOUR TOOLKIT TODAY.

VISA

CALL OR WRITE:

CAROUSEL MICROTOOLS, INC.

609 Kearney Street, El Cerrito, CA 94530 (415) 528-1300

CIRCLE 8 ON READER SERVICE CARD

8087 AND 80287 TECHNICAL TOOLS

87FFT™ performs Forward and Inverse FFTs on real and complex arrays which occupy up to 512K bytes of RAM. Also does convolutions, auto correlations, hamming, complex vector multiplication, and complex to radial conversions. Callable from MS Fortran or 87BASIC/INLINE..... $150

87FFT-2™ performs two-dimensional FFTs. Ideal for image processing. Requires 87FFT...$75

MATRIXPAK™ manages a **MEGABYTE!** Written in assembly language, our runtime package accurately manipulates large matrices at very fast speeds. Includes matrix inversion and the solution of simultaneous linear equations. Callable from MS Fortran 3.2, 87MACRO, 87BASIC/INLINE, and RTOS........ each $150

DATA ACQUISITION PACKAGE
Interactive, user-oriented language which allows the acquisition and analysis of large data streams.................................... CALL

GRAPHICS PACKAGES

Energraphics (stand alone)	295
Grafmatic for MS Fortran or Pascal	125
Plotmatic for Grafmatic	125
Halo for Basic, C or Fortran	each 150

OTHER TOOLS

Alpha Software ESP	595
Borland Sidekick, Toolbox, or Graphics	45
COSMOS Revelation	850
Lattice C	299
PSI MATHPAK	75
smARTWORK	895
SPSS/PC	695
STSC APL★PLUS/PC	475

RTOS - REAL TIME OPERATING SYSTEM
RTOS is a multi-user, multi-tasking real time operating system. It includes a configured version of Intel's iRMX-86, LINK-86, LOC-86, LIB-86, OH-86, and MicroWay's 87DEBUG. Runs on the IBM-PC, XT, PC-AT and COMPAQ.................. 400

INTEL COMPILERS[1]

FORTRAN-86	750
PASCAL-86	750
PL/M-86	500
87C (LATTICE/MICROWAY)	750
ASM-86	200

URS™ - Universal Run Time System[1]
Generates programs with the Intel compilers which run on other operating systems. MS-DOS version is included with RTOS.
Xenix-286 Version.......................... 300

SoftScope Symbolic Debugger[1] 500

[1]Requires RTOS or iRMX-86. All Intel compiler names and iRMX-86 TM Intel Corp.

87BASIC/INLINE™ converts the output of the IBM Basic Compiler into optimized 8087 inline code which executes up to seven times faster than 87BASIC. Supports separately compiled inline subroutines which are located in their own segments and can contain up to 64K bytes of code. This allows programs greater than 128K! Requires the IBM Basic Compiler and Macro Assembler. Includes 87BASIC.............. $200

PC AT and 86-310 DRIVES

30 MEGABYTE WINCHESTER	2000
53 MEGABYTE WINCHESTER	2600
SYQUEST FIVE MEGABYTE	950

HARDWARE AND LANGUAGES

8087-3 5mhz....... $149
Including DIAGNOSTICS and 180-day warranty For IBM PC and compatibles

8087-2 8mhz....... $275
For Wang, AT&T, DeskPro, NEC, Leading Edge

80287-3 5mhz..... $275
For the IBM PC AT

64K RAM Set.............. $30

256K RAM Set......... $195

128K RAM Set PC AT..$225

NUMBER SMASHER™...CALL
10mhz 8087 coprocessor board for the IBM PC

FORTRAN and UTILITIES

Microsoft Fortran 3.2	239
IBM Professional Fortran	595
Intel Fortran-86[1]	750
FORLIB+	65
STRINGS and THINGS	65

BASIC and UTILITIES

IBM Basic Compiler	270
87BASIC/INLINE	200
Summit BetterBASIC™	175
Summit 8087 Module	87

MACRO ASSEMBLERS

IBM Assembler with Librarian	155
87MACRO	150

PASCAL

Microsoft Pascal 3.2	209
Borland Turbo	45
Turbo with 8087 Support	85

NO CHARGE FOR CREDIT CARDS
ALL ITEMS IN STOCK
CALL FOR COMPLETE CATALOG

Micro Way P.O. Box 79
Kingston, Mass.
02364 USA
(617) 746-7341

You Can Talk To Us!

CIRCLE 57 ON READER SERVICE CARD

A Structured FORTRAN

By David Salomon

Back in the early seventies, before FORTRAN 77 became established, there was considerable interest in redesigning and extending FORTRAN IV to reflect the ideas and principles of structured programming.

Most works in this area[1-6] approached the problem by including new keywords or symbols in their designs. In many of those cases the results bore very little resemblance to FORTRAN IV, 77, or any other version, and looked more like a new language. *Structured FORTRAN With No Preprocessor*,[7] though, was an example of a different approach.

The design presented there is novel and based on the following principles:

- The control structures commonly used by structured languages, such as *IF-THEN-ELSE*, *DO-WHILE*, *REPEAT-UNTIL*, and *CASE*, should be included
- No new keywords or symbols should be added to the original language
- The new, extended version should be compatible with FORTRAN IV in the sense that every valid statement in FORTRAN IV should be valid (and have the same meaning) in the new version.

To satisfy these principles, the proposed version assigns special meaning to:

- Certain combinations of existing keywords (like *IF . . . DO* or *IF . . . CONTINUE* in the same statement)
- The keyword *CONTINUE* (in some cases)
- Statement indentation (in one case).

Several control structures are proposed: the *IF-THEN-ELSE*, *REPEAT-UNTIL*, *WHILE-DO*, and *CASE* constructs.

IF-THEN-ELSE

Two forms, long and short, are proposed (Figure 1). When analyzing this figure, several points need to be considered.

The short form is the case where only the *THEN* part exists and contains a single statement. The short form is therefore identical to the *IF* statement in FORTRAN IV.

The word *CONTINUE* precedes the *ELSE* clause. The *CONTINUE* should only appear if there is an *ELSE* clause and, to avoid any dangling *ELSE*s, should be aligned with the corresponding *IF*. This statement is the only case in the design where statement indentation is syntactically significant.

All the statements in the *THEN* clause should have the same indentation as well as all the statements in the *ELSE* clause. The two parts, however, need not have the same indentation.

Both *THEN* and *ELSE* clauses may have nested *IF*s. Any of the *ELSE*s may be omitted (with their corresponding *CONTINUE*s) without any ambiguity. Figure 2

```
Long form                   Short form

IF(condition)               IF(condition) statement

      statement

         .

         .

      statement

CONTINUE

      statement

         .

         .

      statement
```

Figure 1.

shows a typical nested *IF*, and Figure 3 is a quadratic equation subroutine with a nested *IF*.

REPEAT-UNTIL
The *REPEAT-UNTIL* construct also has two forms, as follows:

```
      DO n

      statement(s)

n     IF(condition)CONTINUE
```

This form repeats while the condition is true, at least once.

```
      DO n

      statement(s)

n     IF(.NOT.(condition))CONTINUE
```

This form repeats until the condition becomes true—again, at least once.

```
IF(condition)
        __________
        __________
        __________
     IF(condition)
           __________
           __________
           __________
     CONTINUE
           __________
           __________
           __________
CONTINUE
        __________
        __________
        __________
```

Figure 2.

```
SUBROUTINE QUAD(A,B,C,H1,H2,MODE)
DISC=B*B-4*A*C
IF(DISC.GT.0)
     MODE=1
     D=SQRT(DISC)
     H1=(-B+D)/(2*A)
     H2=(-B-D)/(2*A)
CONTINUE
     IF(DISC.EQ.0)
          MODE=2
          H1=-B/(2*A)
          H2=H1
     CONTINUE
          MODE=3
RETURN
END
```

Figure 3.

```
      Long form                          Short form

      GO TO Var n                        GO TO Var n
           const1[,const2]...                 const1[,const2]...statement
                statement                     const3[,const4]...statement
                .                             .
                statement                n    const5[,const6]...statement
           const3[,const4]...
                .
                statement
                .
           const5[,const6]...
                statement
                .
n               statement
```

Figure 4.

WHILE-DO

The *WHILE-DO* construct is:

```
      IF(condition) DO n
            .
            .
      statement(s)
            .
n     statement
```

The range of statements is executed while the condition is true. As in any *WHILE-DO*, the range of statements will be skipped if the condition is initially false. This construct uses no special indentation and the only problem may be a long condition that separates the *IF* from the *DO* and makes it hard to recognize the construct as a *WHILE-DO*. This, however, can be solved by writing the *DO* directly under the *IF* on a continuation line.

CASE

The *CASE* construct has two forms, long and short (Figure 4). In this figure, notice how:

- Brackets denote an option and ellipses (. . .), a repetition. Each case must therefore include at least one constant and could optionally have more.
- The *CASE* variable can be of any type. The constants must be of the same type.
- No special indentation is used. The end of every group of statements (except the last) is specified by a new line containing constant(s). Constants are easily identified in FORTRAN—they start with a digit, a sign, a quote, a period, or the letter O (for the octal constants, but then they must be longer than the longest variable name to remove any ambiguity)— so they cannot be confused with the beginning of a statement.
- The short form is used when a group contains a single statement. The two forms can be mixed (Figure 5).
- The *CASE* construct is terminated with a statement number and not with a *CONTINUE* since the special use of *CONTINUE* should be limited.
- This construct presents a certain difficulty to the compiler. When the compiler reads the first line, e.g., *GO TO COLOR 4*, it does not know whether the line is the beginning of a *CASE* statement or is an assigned *GO TO*. To decide, the compiler has to read the next line. If the

```
          GO TO OPRTR 1000
                'PLUS'  X=X+Y
                'MINUS' X=X-Y
                '*'     X=X*Y
                'ABS'   IF(X.LT.0)X=-X
                'AND','OR','NOT'
                      PRINT 15
      15              FORMAT('LOGICAL OP')
    1000              CALL LOGOP(OPRTR,X,Y)
```

Figure 5.

```
      ASSIGN 2000 TO COLOR 4
         _____
      COLOR = 'BLUE'
         _____
      GO TO COLOR 4  an assigned go to (GO TO 2000) because
      IF...          this line starts with a non-constant.
         _____
      GO TO COLOR 4       a case statement since
          'BLUE' Y=1      this line starts with a constant.
          'RED'  Y=2
    4     'GREY' Y=3
```

Figure 6.

```
         FUNCTION SQRT(X)
         DATA EPS/E-6/
         X1=1.
         DO 10      a repeat-until construct
         X0=X1
         X1=.5*(X0+X/X0)
      10 IF(.NOT.(ABS(X1-X0).LE.EPS))CONTINUE
         SQRT=X0
         RETURN
         END
```

Figure 7.

ACTIVE TRACE

"Software that lives up to its promises. *When a Basic program doesn't work the way you want it to, this package... will help you track the problem down... Scope is a tool for the beginning, advanced, or professional programmer, and it begins where the cross reference maps leave off."*

Howard Glosser, *Softalk for the IBM Personal Computer* July '84, pp 120-121

"Extremely useful program... *Anyone doing much programming in Basic should appreciate Active Trace a lot."*

Jerry Pournelle, *Byte Magazine* April '83, p 234

"A marvelous Basic programming aid... *It's just amazing to watch a program you wrote run under Scope, and debugging becomes if not trivial, then at least doable"*

Thomas Bonoma, *Microcomputing*, Dec. '83, p 22

*"...**a really neat utility**... designed to untangle even the most convoluted Basic program.... The documentation is almost worth the price of the package."*

Susan Glinert-Cole, *Creative Computing*, July '84, p 210

Active Trace will lead you through your program letting you know variable values (all variables or just those you specify) as they change. Your program's internal activity is presented on your screen, or printer, or it can be saved on disk. **It's simple, effective and works with the BASIC you already own.**

Active Trace **$79.95**

Includes Scope, XREF mapping and documentation

Active Trace is available for most **MS-DOS** and **CPM 2.2** systems and supports the special features of Brand specific versions of Microsoft Basic such as **Basica** on the **IBM-PC**.

AWARECO
Active Software

P.O. Box 695 Gualala, CA 95445
(707) 884-4019
800-358-9120(US) 800-862-4948(CA)

Active trace, Active software, and Scope are trademarks of AWARECO—CPM is a trademark of Digital Research—MS-DOS and Microsoft are trademarks of Microsoft Corporation—IBM-PC is a trademark of IBM Corp.

CIRCLE 3 ON READER SERVICE CARD

```
      SUBROUTINE BINSRC(AR,N,X,MID)
      DIMENSION AR(N)
      INTEGER HIGH
      LOW=1
      HIGH=N
      DO 35                         repeat-until
      MID=(LOW+HIGH)/2
      IF(AR(MID).LT.X)              if
           LOW=MID+1                then
      CONTINUE                      else
           IF(AR(MID).GT.X)         nested if
                HIGH=MID-1
           CONTINUE
                RETURN
   35 IF(.NOT.(LOW.GT.HIGH))CONTINUE
           MID=-1
           RETURN
           END
```

Figure 8.

```
      LOGICAL FUNCTION VALDAT(MONTH,DAY,YR,DWEEK)
      LOGICAL LEAP
      INTEGER DAY,YR,DWEEK,CENT,DAWEEK
      IF(YR/100*100.EQ.YR)
           LEAP=.FALSE.
           IF(YR/400*400.EQ.YR)LEAP=.TRUE.
      CONTINUE
           LEAP=.FALSE.
           IF(YR/4*4.EQ.YR)LEAP=.TRUE.
      GO TO MONTH 100               case
           'JAN','MAR','MAY','JUL','AUG','OCT','DEC'
                MAXDAY=31
           'FEB'
                IF(LEAP)
                     MAXDAY=29
                CONTINUE
  100                MAXDAY=28
      CENT=YR/100
      YR=YR-100*CENT
      DAWEEK=INT(2.6*MONTH-0.2)+DAY+YR+YR/4+CENT/4-2*CENT
      DAWEEK=MOD(DAWEEK,7)
      IF(DAY.GE.1.AND.DAY.LE.MAXDAY.AND.DWEEK.EQ.DAWEEK)
           VALDAT=.TRUE.
      CONTINUE
           VALDAT=.FALSE.
      RETURN
      END
```

Figure 9.

next line starts with a constant, the construct is a *CASE*. Otherwise it is an assigned *GO TO*. Figure 6 is an example of this situation.

This difficulty is general to FORTRAN and is not introduced by the new design. It arises because of two syntax rules in FORTRAN. One rule says that keywords are not reserved (thus *IF* and *DO* may be variable names), and the other says blank spaces do not count (except inside a string) and may appear even in a variable name (thus *COLOR4* and *COLOR 4* are the same name).

These rules make it hard for any FORTRAN compiler to identify statements. A statement such as *GO TO GO TO* is valid if *GOTO* (or *GO TO*) is an assigned variable. The statement *IF(IF)* . . . is valid if *IF* is a logical variable. When the compiler reads *DO 10 I = 1* . . . it has to read beyond the 1 in order to identify the statement. It could be a *DO* statement (if the 1 is followed by a comma) or an assignment statement (if the 1 is the start of an expression).

Figure 5 is an example of the *CASE* construct with short and long groups.

The proposed extension

Figures 7, 8, and 9 are examples of a square-root function, a binary search subroutine, and a date validation function. They all contain the proposed control structures, and it is clear that they look familiar. They in fact look like FORTRAN IV.

A quick look at these short examples is enough to convince anyone that they are in fact written in FORTRAN. The question of clarity and readability still exists, though.

The proposed version does not use any new keywords and the special use of indentations or *CONTINUE* is limited. It could be argued that writing *CONTINUE* instead of *ELSE* does not contribute to clarity and is not really an advantage. It should be noticed, however, that the way *CONTINUE* is used in FORTRAN IV (as a loop terminator) does not suggest continuation either. A word like *RESUME* (or *REPEAT*) would have been a better choice for a loop terminator. The fact that FORTRAN users quickly get used to the meaning of *CONTINUE* suggests that they might quickly get used to its new meaning as well.

The use of indentations in the *IF-THEN-ELSE* construct implies that indentations cannot be used to add clarity to an *IF*. This is a disadvantage of the proposed design, especially for long *IF*s. In practical cases, however, long *IF*s usually contain nested *IF*s that will be indented (because of our syntax rules), thereby making the entire *IF* easier to read.

The next natural step in the extension of FORTRAN should be the introduction of block structures with local and global variables. Three possibilities seem promising and should be explored:

■ Extending the compound statements proposed by L.P. Meissner[3] to become blocks with declarations.

■ Defining blocks as sections of code that start with declaration(s) and end with an END. This seems to conform nicely to the other extensions proposed here and seems worth experimenting with.

■ Defining a block to be a subprogram (a subroutine or a function). This means that *internal* subprograms will have to be defined, which goes against the grain of FORTRAN IV, but at least can be done without adding new keywords. ■

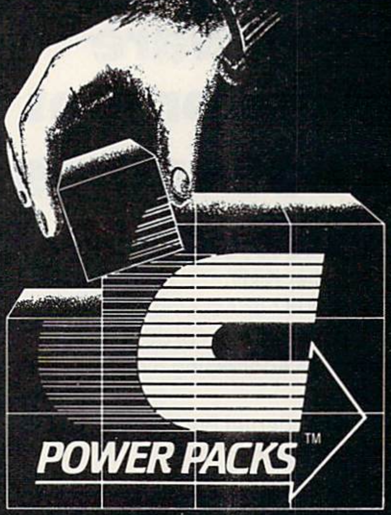
References

1. Meissner, L.P. *Proposed Control Structures for Extended FORTRAN*. 11:1, p. 17 (1976).
2. Meissner, L.P. *A Compatible Structured Extension to FORTRAN*. 9:10, p. 29 (1974).
3. Meissner, L.P. *On Extending FORTRAN Control Structures to Facilitate Structured Programming*. 10:9, p. 29 (1975).
4. Higgins, D.S. *A Structured FORTRAN Translator*. 10:2, p. 42 (1975).
5. O'Neill, D.M. *SFOR--A Precompiler for the Implementation of a FORTRAN Based Structured Language*. 9:12, p. 22 (1974).
6. Bond, R. *Free Form Structured FORTRAN Translator*. 10:10 p. 12 (1975).
7. Gales, L.E. *Structured FORTRAN With No Preprocessor*. 10:10, p. 17 (1975).

Note: All items appeared in SIGPLAN Notices.

David Salomon has a Ph.D. in physics from Hebrew Univ. in Israel. He has taught for the past 12 years—first at the State Univ. of New York, Buffalo, N.Y., and now at the State Univ. of California, Northridge, Calif. He is interested in computer algorithms and graphics.

"This is a beautifully documented, incredibly comprehensive set of C Function Libraries."
— Dr. Dobb's Journal



COMPLETE SOURCES

☐	**PACK I: Building Blocks I** 250 Functions: DOS, Printer, Video, Asynch	$149
☐	**PACK 2: Database** 100 Functions: B-Trees, Variable Records	$399
☐	**PACK 3: Communications** 135 Functions: Smartmodem™, Xon/Xoff, Modem-7, X-Modem	$149
☐	**PACK 4: Building Blocks II** 100 Functions: Dates, Text Windows, Pull-down Menus Data Compression	$149
☐	**PACK 5: Mathematics I** 35 Functions: Log, Trig, Square Root	$99
☐	**PACK 6: Utilities I** Archive, Diff, Replace, Scan, Wipe (Executable Files only)	$99

Lattice™, Microsoft™, DeSmet™, CI-86™ Compilers on IBM PC/XT/AT™
Small and Large Memory Models.
Credit cards accepted
($7.00 handling/Mass. add 5%)

SOFTWARE HORIZONS INC.

165 Bedford Street
Burlington, Mass. 01803
(617) 273-4711

NOVUM ORGANUM

CIRCLE 25 ON READER SERVICE CARD

C Software Development PCDOS/MSDOS

Complete C Compiler
- Full C per K&R
- Inline 8087 or Assembler Floating Point, Auto Select of 8087
- Full 1Mb Addressing for Code or Data
- Transcendental Functions
- ROMable Code
- Register Variables
- Supports Inline Assembler Code

MSDOS 1.1/2.0 Library Support
- All functions from K&R
- All DOS 2.0 Functions
- Auto Select of 1.1 or 2.0
- Program Chaining Using Exec
- Environment Available to Main

c-window™ Symbolic Debugger
- Source Code Display
- Variable Display & Alteration Using C Expressions
- Automatic Commands
- Multiple Breakpoints by Function & Line Number

8088/8086 Assembler
- FAST — Up to 4 times Faster than IBM Assembler
- Standard Intel Mnemonics
- Compatible with MSDOS Linker
- Supports Full Memory Model

8088 Software Development Package

$199⁰⁰

Includes: C Compiler/Library, c-window, and Assembler, plus Source Code for c-systems Print Utility

c-systems

P.O. Box 3253
Fullerton, CA 92634
714-637-5362

CIRCLE 16 ON READER SERVICE CARD

THE PROGRAMMER'S SHOP™

helps compare evaluate and find products. Get answers.

SERVICE: FREE LITERATURE

One free call covers all programmer's software. Ask for a "Packet" on: "AI", BASIC, C, COBOL, Debuggers, Editors, FORTH, FORTRAN, Libraries, PASCAL, UNIX/PC or 30 "addons" for "C".

"C" Language

	LIST PRICE	OUR PRICE
MSDOS: C86 - 8087, reliable	$395	call
Lattice 2.1 - improved - 30 addons	500	call
Microsoft C2.x	500	349
Williams - NEW, debugger	500	call
Instant C Interpreter, fast, full	NA	500
CPM80: Ecosoft C-now solid, full	250	225
MAC: Megamax - fast, full	NA	295

EDITORS

BRIEF - Intuitive, flexible	PCDOS	NA	195
PMATE - powerful	8086	225	195
VEDIT - full, liked	8086	150	119

ARTIFICIAL INTELLIGENCE

IQ LISP - full 1000K RAM	PCDOS	175	call
TLC LISP - with "classes", nice	MSDOS	NA	250
MicroProlog - by Logic Prog. Assem.	MSDOS	NA	285
PROLOG-86 - standard. Learn fast	MSDOS	NA	125
EXSYS - Expert System	PCDOS	NA	295

Feature

PERISCOPE DEBUGGER — "Reset Box", own RAM, Registers, symbols, line nums, 2 screen, PCDOS. $295.

Recent Discovery

GC LISP - "COMMON" LISP, Help, tutorial, co-routines, arrays, thorough. PCDOS. $475.

FORTRAN

	Runson	LIST PRICE	OUR PRICE
MS Fortran - Improved	MSDOS	350	249
DR Fortran-86 - full '77'	8086	500	349
PolyFORTRAN - XREF, Xtract	MSDOS	NA	165

SUPPORT PRODUCTS

LIBRARIES: BTRIEVE ISAM	MSDOS	245	215
CIndex+ - ISAM, source, no royalt.	8086	NA	375
CSHARP Realtime - source, full	MSDOS	NA	600
CVIEW - screens, validation	MSDOS	245	195
GraphiC - 4200 x 3100, source	MSDOS	NA	195
Greenleaf C - thorough	MSDOS	NA	165
HALO Graphics - fast, full	PCDOS	200	175
TOOLS: Disk Mechanic-rebuild	MSDOS	70	65
MULTILINK - multitask	PCDOS	295	265
Polylibrarian-thorough	MSDOS	99	89
PolyMAKE-compiles	PCDOS	99	89
Profiler-86-easy to setup, symbols	MSDOS	NA	125
XShell-add IF-THEN-ELSE	MSDOS	225	215
XENIX - "true S3", rich, + C-MSDOS	PC	1350	1285

Call for a catalog and *solid value*

800-421-8006

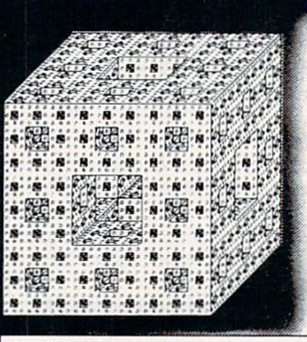
THE PROGRAMMER'S SHOP™
128- Rockland Street, Hanover, MA 02339
Visa Mass: 800-442-8070 or 617-826-7531 MasterCard

Note: All prices subject to change without notice.

Mention this ad. Some prices are specials.

All formats available.
Ask about POs, COD.

CIRCLE 71 ON READER SERVICE CARD



WALTZ LISP(TM)

*The one and only **adult** Lisp system for CP/M users.*

Waltz Lisp is a very powerful and complete implementation of the Lisp programming language. It includes features previously available only in large Lisp systems. In fact, Waltz is substantially compatible with *Franz* (the Lisp running under *Unix*), and is similar to *MacLisp*. Waltz is perfect for Artificial Intelligence programming. It is also most suitable for general applications.

Much faster than other microcomputer Lisps. • Long integers (up to 611 digits). Selectable radix • True dynamic character strings. Full string operations including fast matching/extraction. • Flexibly implemented random file access. • Binary files. • Standard CP/M devices. • Access to disk directories. • Functions of type lambda (expr), nlambda (fexpr), lexpr, macro. • Splicing and non-splicing character macros. • User control over all aspects of the interpreter. • Built-in prettyprinting and formatting facilities. • Complete set of error handling and debugging functions including user programmable processing of undefined function references. • Virtual function definitions. • Optional automatic loading of initialization file. • Powerful CP/M command line parsing. • Fast sorting/merging using user defined comparison predicates. • Full suite of mapping functions, iterators, etc. • Assembly language interface. • Over 250 functions in total. • The best documentation ever produced for a micro Lisp (300+ full size pages, hundreds of illustrative examples).

Waltz Lisp requires CP/M-86 or CP/M 2.2, Z80 and 48K RAM (more recommended). All common 5'' and 8'' disk formats available.

Version 4.4 (Now includes Tiny Prolog written in Waltz Lisp.)

$169*

*Manual only: $30 (refundable with order). All foreign orders: add $5 for surface mail, $20 for airmail. COD add $3. Apple CP/M and hard sector formats add $15.

ProCode(TM) INTERNATIONAL
15930 SW Colony Pl.
Portland, OR 97224

Unix* Bell Laboratories.
CP/M* Digital Research Corp.

Call free **1-800-LIP-4000** Dept. #13
In Oregon and outside USA call 1-503-684-3000

CIRCLE 60 ON READER SERVICE CARD

Program in Style

By William E. Weinman

We often see particular programming techniques emphasized in the literature. Some have been given names, such as structured programming, egoless programming, top-down design, and stepwise refinement. Each of these disciplines has its merits and will be described and even encouraged in this article.

But never let it be said that the practice of any set of specific rules can give your code performance, readability and maintainability unless the purpose of the act of coding, as well as the purpose of the code itself, is kept in the forefront of your mind as it is being written.

The objective of this article is to help us all keep in mind the purpose of our programming efforts from conception through design and execution to completion and delivery of the final product.

For example, let's say that we're writing a set of routines to allow a computer operator to manage a table of data. The operator needs to insert entries into the table, delete them, edit them, move them, etc.

We could start with a general concept and refine it in logical steps, then code it piece-by-piece—top down so to speak—and avoid *GOTO*s by using plenty of modern control structures in their stead (*WHILE*, *FOR*, etc.). Using this approach we would efficiently generate some good-looking and readable code.

Right? Don't be too quick to agree. I've seen plenty of programming efforts fall by the wayside for reasons completely unrelated to those addressed by these techniques.

In this example our purpose was stated as being the management of a specific set of data. The data was of a dynamic nature—having the need for adding to the set, deleting from the set, editing it, etc.—and our coded system was to be used by an operator as opposed to a programmer. This problem is data-related and user interface-related. So, before considering the structure of the code, we would need to design the data base schema and user interface.

This is not in any way meant to imply that coding technique is unimportant. However, the intent here is not to build technicians but to build, with a measure of rationality and coherence, the foundation for good, efficient programmers. This will allow us to concentrate more on the creative aspects of our trade.

First let's analyze each of the previously mentioned programming techniques in some detail. Then, based on their similarities, we can draw some interesting conclusions that give us the benefits of all of the techniques without a lot of confinements that could constrict our creativity.

Structured programming

I put this one first because the term has been so abused over the last 10 years or so that the controversial nature of the title has overshadowed the tremendous effect that the techniques described by it has had on our profession.

The term structured programming was made controversial by a book published in 1972 by the same name.[1] Essentially the technique promotes the modularity of code to the end that each distinct functionality has its own section (or structure) that it operates within.

The advantage of this is that the code becomes maintainable. In other words, when you need to make a change, fix a bug, or add a feature to code that is written in this fashion, it is much easier to isolate the particular section of code that needs attention.

What are the structures to which the name refers? Disagreement on this point is prevalent but, being brave by nature, I'll venture a set of definitions.

Consider the previous example of the set of programs for managing a table of data. For argument's sake, let's say that this system could be written in two ways:

- It could all be written in-line or as a long list of instructions with conditional branches (the abominable *GOTO*) for the various different functions that need to be executed or

■ According to the discipline of structured programming, it could be written with different sections, or structures, for each major function and, in turn, each minor function could be implemented in a separate "workhorse" function, each of which could accommodate more than one of the major functions.

A structured language such as Pascal, ALGOL, Ada, or C is not required to have structured code but it does help. These languages were designed with facilities that not only accommodate good program structure but actually encourage it. They do not guarantee structured code, however. It is not significantly easier to write structured Pascal than it is to write structured BASIC or FORTRAN. It takes mental discipline.

Let's consider an example.

I will present it in Pascal for two reasons. First, I dislike the fact that so many have embraced Pascal as the best thing to happen to programming since M&Ms with peanuts. At best it is a good language for teaching beginners and quickly weaning them from. (I might as well come out and say it—I don't really like Pascal.) The second reason is that people seem to think that all they need to write good code is a Pascal compiler. We'll see why this is a misconception.

In Listing 1 we have a program that compiles without error and executes properly, effecting the function its designer intended. Is it immediately obvious, however, what it is that it was designed to do?

Perhaps it is more obvious in Listing 2. In this example, I have used a control structure to replace the label, the *goto*, and the initialization and incrementing of the index variable. Additionally I've indented the code according to a convention that is consistent, used spaces to clearly delineate operators, and utilized a comment to indicate the location of the main procedure which is by necessity detached from its declaration in Pascal.[2]

Most compilers will generate essentially the same object code for both examples, but which would you rather maintain?

Technically speaking, most of these improvements are not part of what was originally termed structured programming, but they have become part of its scope over the years and therefore are included here.

To summarize, structured programming is the use of structures, control structures, and modularized functionality to improve the performance and readability of computer program code.

Egoless programming
This is a term that makes reference to the tendency of most of us to promote the misconception that programming (and especially good programming) requires a level of mental agility that can only be claimed by a select few.

This elitist attitude has an undesirable side effect, however. It encourages us to write code in a vacuum. In other words, many of us have a bent toward writing code that is so clever and cryptic that others will look at it and say, "Wow! What does THAT do?" and be impressed with our cleverness. However, more often than not, if we strive to write code that is so simple that anyone can read it, the code will be easier and more economical to maintain (and usually more efficient!).[3]

It is suggested that we take several steps to avail ourselves of the help of other programmers. One way is to have someone read the section of code in question, making suggestions where he or she sees the need. Another way would be the telephone test—read the code over the telephone to someone and see if the person can understand what it is supposed to do. If it can not be understood, chances are that it can usually be improved to become clearer.

Many of us have the tendency to think egotistically, "This code is so good that it can't be improved anyway," so this technique has been dubbed egoless programming. However, in the interest of maintaining the dignity of this valuable source of constructive criticism, I would prefer to euphemize the title. Any suggestions?

Top-down/stepwise design
Top-down design and stepwise refinement really refer to the same technique of programming from the top down or as a series of step-by-step refinements. In other words, design the highest level of the program first, leaving the details for

```
program example1 (output);

var i : integer;

label 7;

procedure something; begin  writeln('Something'); end;

begin i:=0;
7:  something;
i:=i+1; if i<10 then goto 7;
end.
```

Listing 1.

later. Then design the next level, still leaving the details for later . . . until they have all been attended to.

For example, for the Pascal program in Listings 1 and 2, we could first pseudocode the problem as:

```
do something ten times
```

and further refine it to:

```
for i := 1 to 10 do
    something;
```

then write the procedure for something. But of course this is too simple an example for a technique that is obviously meant for more sizable problems. Let's start designing the data management system instead:

```
              —MAIN—
display a menu
select one of the following from the
        menu
        insert item
        delete item
        move item
        edit item
        browse through items
        squirziklize items
```

Then we can write each of the individual functions using the same technique:

```
              —INSERT—
display form for data
get data from operator
while data still needs editing
        allow operator to edit data
if data is ok
        ask for correct position in table
        insert in table
```

This really serves two excellent purposes. First, of course, it forces the programmer to fully conceptualize each detail of the code before writing it, thus producing far more functional code on the first few passes (sometimes even the first pass!). Second, it provides us with an excellent start for the dreaded documentation effort.

What they have in common

What do these three programming disciplines have in common? Their objective. They are all intended for the programmer who feels a need to write immortal code, code that will be used and modified by others over a period of time somewhat longer than a few days or weeks.

Structured programming encourages modularity of code so that the author and those reading the code later will be able to discern easily the purpose of the various logical sections of code.

The so-called *egoless programming* technique encourages us to share our code with others for our mutual education. Working together will make all of our code more readable, maintainable, and functional.

The *top-down design* and *stepwise refinement* techniques implore us to carefully consider the design of our programs before setting any of it to code. This would have the effect of adding coherence to our finished products and forcing us to pay attention to details that might otherwise be missed.

Why write code for the future? I hope you can understand why without having an experience like one I once had. I was contracted to debug over 5,000 lines of Pascal that bore a significant resemblance to those in Listing 1 but were longer. The programmer that wrote the code appeared to be of the opinion that if it worked (sort of), then it was OK.

I can't complain though; I learned how to *not* write programs.

References

1. Dahl, O.J., Dijkstra, E.W., and Hoare, C.A.R. *Structured Programming*. Academic Press, 1972.
2. Kernighan, Brian. *Why Pascal is Not My Favorite Programming Language.* Bell Laboratories Computing Science Technical Report No. 100, 1981.
3. Kernighan, Brian, and Plauger, P.J. *The Elements of Programming Style.* McGraw-Hill, Second Ed., 1978.

William Weinman is a programming and hardware consultant in the Los Angeles, Calif., area.

```
program example2 (output);

var
    i : integer;

procedure something;
begin
    writeln('Something');
end;

begin { main procedure }
    for i := 1 to 10 do
        something;
end.
```

Listing 2.

DeSmet C

8086/8088 Development Package $109

FULL DEVELOPMENT PACKAGE
- Full K&R C Compiler
- Assembler, Linker & Librarian
- Full-Screen Editor
- Execution Profiler
- Complete STDIO Library (>120 Func)

Automatic DOS 1.X/2.X SUPPORT

BOTH 8087 AND SOFTWARE FLOATING POINT

OUTSTANDING PERFORMANCE
- First and Second in AUG '83 BYTE benchmarks

SYMBOLIC DEBUGGER $50
- Examine & change variables by name using C expressions
- Flip between debug and display screen
- Display C source during execution
- Set multiple breakpoints by function or line number

DOS LINK SUPPORT $35
- Uses DOS .OBJ Format
- LINKs with DOS ASM
- Uses Lattice® naming conventions

Check: ☐ Dev. Pkg (109)
☐ Debugger (50)
☐ DOS Link Supt (35)

SHIP TO: ______________________

____

______________ ZIP ______

C WARE CORPORATION

P.O. BOX C
Sunnyvale, CA 94087
(408) 720-9696

All orders shipped UPS surface on IBM format disks. Shipping included in price. California residents add sales tax. Canada shipping add $5, elsewhere add $15. Checks must be on US Bank and in US Dollars. Call 9 a.m. – 1 p.m. to CHARGE by VISA/MC/AMEX.

CIRCLE 18 ON READER SERVICE CARD

Once you choose Lattice, our friends will C you through...

LATTICE INC.: LATTICE WINDOWS, CURSES UNIX SCREEN CONTROL LIBRARY, C-FOOD SMORGASBORD, dB-C ISAM COMPATIBLE WITH dBASE II AND III . . . **LIFEBOAT ASSOCIATES:** FLOAT 87 8087 SUPPORT PACKAGE, HALO GRAPHICS PACKAGE, PANEL SCREEN LIBRARY . . . **GREENLEAF SOFTWARE:** THE GREENLEAF C FUNCTIONS . . . **C SOURCE:** BASIC_C C FUNCTIONS FOR BASIC USER . . . **SOFTCRAFT:** BTRIEVE ISAM FILE SYSTEM, BTRIEVE ISAM NETWORK FILE SYSTEM . . . **BLAISE COMPUTING:** TOOLS, TOOLS2, VIEW MANAGER SCREEN PACKAGE . . . **MORNING STAR SYSTEMS:** PROLIBRARY, PROSCREEN . . . **CREATIVE SOLUTIONS:** WINDOWS FOR C . . . **NOVUM ORGANUM:** C POWERS PACKS, MATHEMATICS POWER PACKS, ADVANCED POWER PACKS, DATABASE POWER PACKS, TELECOMMUNICATIONS POWER PACKS W/ SOURCE . . . **PHACT ASSOCIATES:** PHACT ISAM LIBRARY . . . **RAIMA CORPORATION:** db_VISTA DBMS . . . **PHOENIX:** PLINK86, PFIX86 . . . **RELATIONAL DATABASE SYSTEMS:** C-ISAM FILE ACCESS METHOD . . . **MINDBANK:** V-FILE VIRTUAL MEMORY/FILE SYSTEM . . . **HUNTER & READY:** VRTX C INTERFACE LIBRARY . . . **GRAPHIC SOFTWARE SYSTEMS:** GSS DRIVERS, GSS TOOLKIT KERNEL SYSTEM . . . **OPT-TECH DATA PROCESSING:** OPT-TECH SORT . . . **ACCUDATA SOFTWARE:** C-TREE ISAM, C-SORT SORT . . . **TRIO SYSTEMS:** C-INDEX+ ISAM . . . **COMPU CRAFT:** c VIEW FORMS/WINDOW MANAGEMENT . . . **SCIENTIFIC ENDEAVORS:** GRAPHIC PRESENTATION SCIENTIFIC GRAPHICS . . . **LEMMA SYSTEMS, INC.:** C LIBRARY . . . **ESSENTIAL SOFTWARE, INC.:** C UTILITY LIBRARY . . . **SOFTWARE LABS:** C UTILITIES PACKAGE . . . **FAIRCOM:** C-tree BY FAIRCOM ISAM WITH SOURCE

Contact Lattice to learn how we can help your C program development.

LATTICE®

P.O. Box 3072
Glen Ellyn, IL 60138
312/858-7950
TWX 910-291-2190

CIRCLE 36 ON READER SERVICE CARD

TOTAL CONTROL:

***FORTH:* FOR Z-80®, 8086, 68000, and IBM® PC**

Complies with the New 83-Standard

GRAPHICS • GAMES • COMMUNICATIONS • ROBOTICS DATA ACQUISITION • PROCESS CONTROL

- ***FORTH*** programs are instantly portable across the four most popular microprocessors.
- ***FORTH*** is interactive and conversational, but 20 times faster than BASIC.
- ***FORTH*** programs are highly structured, modular, easy to maintain.
- ***FORTH*** affords direct control over all interrupts, memory locations, and i/o ports.
- ***FORTH*** allows full access to DOS files and functions.
- ***FORTH*** application programs can be compiled into turnkey COM files and distributed with no license fee.
- ***FORTH*** Cross Compilers are available for ROM'ed or disk based applications on most microprocessors.

Trademarks: IBM, International Business Machines Corp.; CP/M, Digital Research Inc.; PC/Forth+ and PC/GEN, Laboratory Microsystems, Inc.

FORTH Application Development Systems include interpreter/compiler with virtual memory management and multi-tasking, assembler, full screen editor, decompiler, utilities and 200 page manual. Standard random access files used for screen storage, extensions provided for access to all operating system functions.

Z-80 FORTH for CP/M® 2.2 or MP/M II, $100.00; **8080 FORTH** for CP/M 2.2 or MP/M II, $100.00; **8086 FORTH** for CP/M-86 or MS-DOS, $100.00; **PC/FORTH** for PC-DOS, CP/M-86, or CCPM, $100.00; **68000 FORTH** for CP/M-68K, $250.00.

FORTH + Systems are 32 bit implementations that allow creation of programs as large as 1 megabyte. The entire memory address space of the 68000 or 8086/88 is supported directly.

PC FORTH +	$250.00
8086 FORTH + for CP/M-86 or MS-DOS	$250.00
68000 FORTH + for CP/M-68K	$400.00

Extension Packages available include: software floating point, cross compilers, INTEL 8087 support, AMD 9511 support, advanced color graphics, custom character sets, symbolic debugger, telecommunications, cross reference utility, B-tree file manager. *Write for brochure.*

VISA

LMI

Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to (213) 306-7412

MasterCard

CIRCLE 35 ON READER SERVICE CARD

PUBLIC DOMAIN SOFTWARE REVIEW

By Tim Parker

A few years ago (in this galaxy), a machine was developed that revolutionized the home microcomputer field. This machine had the somewhat unlikely name of Apple. It succeeded Commodore's PET, which had a toy-like keyboard, a somewhat limited screen, and hardly any expansion possibilities.

While the PET was usually regarded as a fun machine, the Apple actually had a function. The Apple, as we all know, caught on rather well. When the next improvement came along in the form of the Apple II, true home computing reached new limits.

The Apple set the market standard for an amazing number of years. The number of home computerists who developed programs to enhance the machine and the enthusiastic support of third-party hardware support made it "the" machine to own for nearly a decade—a remarkable feat in this highly competitive world.

While the Apple has had its day in th~ sun and now has been generally dismissed as a viable programmer's microcomputer, lots of them are still out there. They are still being bought in one reincarnation or another and hang on as tenaciously as a mosquito enjoying a bit of skin (which may not be the ideal simile, but you get the idea).

So a rather awkward situation can arise when someone such as your narrator has finished describing the wonderful things that can be done with a CompuPro 8/16 with an 8087 coprocessor and how it compares with the 68000 running at 10MHz and asks the question "what machine(s) do you have?" Instead of a similar description of a home brewed 286/287 board, the person will proudly say, "an Apple IIe with CP/M!"

At its peak, my Apple public domain collection numbered almost 300 disks, all crammed with a bewildering variety of material available. There was (and still is) a plethora of companies and software distribution centers that catered entirely to Apple public domain material. Some specialize in their topics—for example, one company handled only scientific and engineering software.

A complete list of programs available takes several books; you can see this by wandering into any local computer bookstore. Also, the Apple is used by a large percentage of *COMPUTER LANGUAGE* readers, so we hereby dedicate this section of the column to the world of the Apple.

One source of Apple programs is The Public Domain Exchange (address, as usual, at the end of the column). They have compiled a catalog of several hundred disks broken down by topic and each checked for content.

As Apples became the machine in use in most educational institutions (thanks in part to Apple's policy to donate machinery), the amount of education-oriented material rose phenomenally. For instance, the Public Domain Exchange lists 24 Apple dissemination disks contributed by the San Mateo, Calif., County Office of Education and Computer-Using Educators. These disks are aimed at secondary and elementary school levels and have both integer and floating point routines.

Most disks have game-like approaches to teaching. Concepts are introduced by use of text or high-resolution graphics. Most programs require input from the user and provide a dialog.

The disks have the usual assortment of hangman and similar games of interest to youngsters and some nicely executed high school physics programs such as Millikan's Oil Drop and Simple Harmonic Motion. However, these are probably not of too great an interest to *COMPUTER LANGUAGE* readers.

Neither will be the games, of which there are lots, including the typical assortments as well as the still growing Eamon Adventure series. (For those who liked Adventure from Crowther & Woods, this set goes further.) Apparently the number of available adventures exceeds 100 now, but I haven't got the time to play that many.

So the relevant material arises at last. Twenty-nine disks are devoted to Pascal and are packed with utilities for writing Pascal programs and manipulating the Apple. Most of the disks are full, with programs available in source code format. The disks, unfortunately, are rather randomly assembled, so any disk examined will have an assortment of applications, games and utilities. This may be preferable for some people, but theme disks have proven their value in almost all public domain distributions and would be a nice touch here.

Apple CP/M is supported by eight disks, again of a general variety. The ubiquitous MODEM7 version abounds on a number of the disks in various reincarnations as do adaptations of most standard CP/M utilities. These include the squeezer/unsqueezer, cyclic redundancy check (CRC), disk utility (DUU), NCAT-XCAT cataloging program, disassembler (REZILOG and RESOURCE), and yet more games.

Utilities for DOS cover 16 disks and include copy routines by the dozen, sorts, debuggers, disk examination routines, catalogers, system tests and diagnostics, printer drivers, graphic aids, and so on. A couple of disks cover the HELLO program in all its glory and have menu drivers as well.

On the practical side, several disks of mathematical and statistical routines are available. Several curve fitting programs are presented, with various approaches to the least-squares fitting of straight line, logarithmic, exponential, and curve data with a best fit. Simple calculus (definite integrals) is handled by a number of programs, as is matrix manipulation. As expected, routines such as matrix complement take quite a while, as do some of the statistical analyses. Fourier transforms, Poisson distributions, chi square, and the other standard stats functions work well in most cases, although some tend to hiccup at times, losing the programmed data.

Eight disks in the catalog are business and finance applications and include amortization, finance routines such as present and future values, inventory management, and loan calculations.

Finally, a number of disks are devoted to graphics, sound and (gads!) "passion (sex)." No comment on the latter. (Maybe it's time to dust off the old Apple, though . . .)

The Public Domain Exchange charges a very reasonable $5.00 a disk, plus $4.00 more for shipping and handling. Turnaround on orders is usually fast, and so far I've heard no complaints about their service.

The Apple blazed its own path. Now we'll consider the IBM PC, a machine that had a path blazed for it by a company name. Two months ago I discussed some of the software available for the PC and its clones from the PC-SIG organization. Since then I have talked to the people who run the group and have been very impressed with their support and attention.

PC-SIG vol. 194 has the ROFF text formatter that is becoming the rage these days. ROFF got quite a boost from an article in one of the recently demised magazines. Vol. 194 is an update of the earlier PC-SIG vol. 50, and the C source code is supplied. Also on the same disk is a program called FOGFIND, which "determines the clarity of text." I haven't tried that one, but it sounds very much like a typical grammar checker except for the fact that it is free.

To round out vol. 194 is an integration routine written in a pair of languages: C and Pascal. A comparison provides an interesting look at the programmer's approaches to the two languages and their suitability for the task. Without starting a big debate, it is probably obvious which code is more efficient. Support and example files for the integrators are provided in both languages.

DVED, a screen editor on PC-SIG vol. 191 has been adapted for use with a mouse. Also on the volume are a few graphics utilities for most routine graphic applications, all written in BASIC. One program, EASEL, allows good graphics manipulation through a light pen and a color display.

A BASIC preprocessor called RASCAL is on PC-SIG vol. 162. It works equally well on MBASIC. A number of utilities are on the disk for BASIC programmers. The same disk has a screen-oriented text editor called RED with documentation, but I didn't check this one.

An IBM macro assembler tutorial is on PC-SIG vol. 166. More than adequate documentation is contained in five files and two disassemblers, with documentation. These should be in every assembler programmer's library, as they work well.

Finally, in the mathematics and statistics department again, PC-SIG vol. 180 features several dozen programs. They all are driven by a master menu that calls the programs by the rather cryptic names of S# or M# where # varies from 1 to 26. Completely in BASIC, they are not as fast as they would be if assembled, but they work very well on tested data.

Stats programs include binomial, Poisson, chi square, F, and normal distributions, with geometric, linear, multiple linear, exponential, and Nth order regression. Integration, matrix manipulation, roots, conversions, and simultaneous equations are a few of the mathematics routines. None have a great impact; they could be programmed by a reasonably competent programmer. However, having them all available on one menu is a nice alternative to hours at the keyboard.

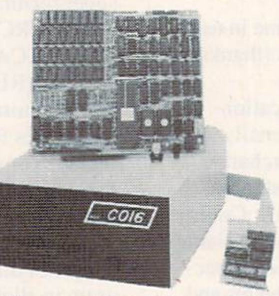
After the Apple and the IBM, we're left with machines that don't have quite the elegant history these two have or the mystique that surrounds them. In my opinion, the more mundane machines are much more interesting.

In the next few months we'll return to an old favorite, CP/M, and check its latest polish from Richard Conn—ZCPR3. Also, UNIX users (Xenix, etc.) can start gearing up for an all too brief look at the public domain world accessible to them.

Useful addresses: SIG/M is at P.O. Box 2085, Clifton, N.J. 07015-2085. CP/MUG is at 1651 Third Ave., New York, N.Y. 10028. PC-SIG is at 1556 Halford Ave., Suite 130, Santa Clara, Calif. 95051, (408) 730-9291. Elliam Associates is at 24000 Bessemer St. Woodland Hills, Calif. 91367, (818) 348-4278. The Public Domain Exchange is at 673 Hermitage La., San Jose, Calif. 95134, (408) 942-0309. ■

A POWERFUL 68000 DEVELOPMENT ENVIRONMENT FOR YOUR Z80 SYSTEM

CO1668 ATTACHED RESOURCE PROCESSOR



- 68000 Assembler
- C Compiler
- Forth
- Fortran 77

- Pascal
- BASIC-PLUS
- CBASIC
- APL. 68000

6 MHZ 68000 CP/M-68K 768K RAM
4 x 16081 MATH CO-PROCESSORS CPM80 RAM DISK

Develop exciting 68000 applications on your current Z80 based CPM system using powerful mini-frame like 32 bit programming languages. And then, execute them at speeds that will shame many $100K plus *mini*computer systems.

The **CO1668 ATTACHED RESOURCE PROCESSOR** offers a Z80 CPM system owner a very low cost and logical approach to 68000 development. You have already spent a small fortune on 8 bit diskette drives, terminals, printers, cards cages, power supplies, software, etc. The CO1668 will allow you to enjoy the vastly more powerful 68000 processing environment, while preserving that investment.

CO1668 ATTACHED RESOURCE PROCESSOR SPECIAL FEATURES:

- 68000 running at 6 Mhz
- 256K to 768K RAM (user partitioned between CPU and RAM Disk usage)
- Up to four 16081 math co-processors
- Real time clock, 8 level interrupt controller & proprietory I/O bus
- Available in tabletop cabinet
- Delivered w/ sources , logics, & monolithic program development software
- Easily installed on *ANY* Z80 CPM system
- CP/M-68K and DRI's *new* UNIX V7 compatible C complier (w/ floating point math) - standard feature
- *Can be used as 768K CPM80 RAM Disk*
- Optional Memory parity
- No programming or hardware design required for installation
- Optional 12 month warrantee

- PRICES START AS LOW AS $899.00 for a CO1668 with 256K RAM, CPM68K, C Complier, Sources, Prints, 200 page User Manual, Z80 Interface, and 68000 System Development Software.

For further information about this revolutionary product or our Intel 8086 Co-Processor, please send $1 [no checks please] or call:

HSC INC.

**Hallock Systems Company, Inc.
262 East Main Street
Frankfort, New York 13340
(315) 895-7426**

RESELLER AND OEM INQUIRIES INVITED.

CIRCLE 31 ON READER SERVICE CARD

ARTIFICIAL INTELLIGENCE

Programming:
Learn Fast,
Experiment,
Prototype

with the nonprocedural language chosen by Japan

PROLOG-86™

implements the "standard", features described in Clocksin and Mellish, has tutorials and sample programs to learn from that include:

- **an Expert System**
- **a Natural Language Interface**

1 or 2 pages of PROLOG is often equivalent to 10 or 15 pages in "C" or PASCAL. It is a different way of thinking. Become familiar in **one evening**, comfortable in days.

CONTEST: "Artificial Intelligence Concepts". All entries that teach key concepts and are clear will get recognition. The best will win $1,000. Submit by 2/28/85.

AVAILABILITY: PROLOG-86 runs on MSDOS, PCDOS or CPM-86 machines. We provide most formats. The Intro price is $125.

Full Refund if not satisfied during first 30 days.

617-659-1571

Solution Systems™
335-L Washington Street
Norwell, MA 02061

CIRCLE 81 ON READER SERVICE CARD

C Helper™

FIRST-AID FOR C PROGRAMS

Save time and frustration when analyzing and manipulating C programs. Use C HELPER's UNIX-like utilities which include:

DIFF and **CMP** – for "intelligent" file comparisons.
XREF – cross references variables by function and line.
C Flow Chart – shows what functions call each other.
C Beautifier – make source more regular and readable.
GREP – search for sophisticated patterns in text.

There are several other utilities that help with converting from one C compiler to another and with printing programs.

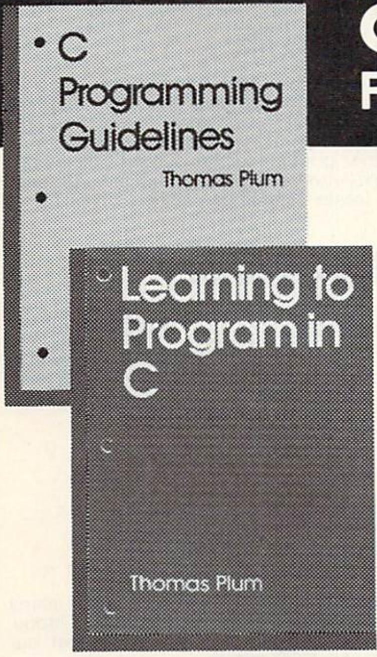
C Helper is written in portable C and includes both full source code and executable files for $135 for MS-DOS, CPM-80 or CPM-86. Use VISA, Master Card or COD.

Call: **617-659-1571**

Solution Systems™
335-L Washington Street
Norwell, MA 02061

CIRCLE 82 ON READER SERVICE CARD

C LANGUAGE PROGRAMMING
From Plum Hall...the experts in C training



Learning to Program in C 372 pp., 7½" x 10", Price $25.00
A practical, step-by-step guide for everyone acquainted with computers who wants to master this powerful "implementer's language" Inside, you will learn how to write portable programs for the full spectrum of processors, micro, mini and mainframe

C Programming Guidelines 140 pp., 7½" x 10", Price $25.00
A compilation of standards for consistant style and usage of C language. Arranged in manual page format for easy reference, it presents time-tested rules for program readability and portability.

FREE
C LANGUAGE POCKET GUIDE!
A handy C language programming pocket guide is yours *free* when you order either (or both) of the manuals above. A full 14 pages of valuable C language information!

PLUM HALL
1 Spruce Av, Cardiff NJ 08232
The experts in C and UNIX™ training.
Phone orders: 609-927-3770

Please send me: ________ information on C and UNIX Training Seminars
________ copies of Learning to Program in C @ $25.00/copy
________ copies of C Programming Guidelines @ $25.00/copy
NJ residents add 6% sales tax.

NAME ____________
COMPANY ____________
ADDRESS ____________
CITY/STATE/ZIP ____________
☐ Check ☐ American Express ☐ Master Card ☐ Visa
CARD # ________ EXP. DATE ____ / ____ Signature ____________

UNIX is a trademark of AT&T Bell Laboratories

CIRCLE 50 ON READER SERVICE CARD

U N I X   S O F T W A R E

UniPress Product UPDATE

AMSTERDAM COMPILER KIT FOR UNIX™

- Package of C and Pascal compilers, cross compilers and assemblers. The Kit was developed at the Vrije University of Amsterdam and has been used in Europe for years.
- Available for a wide variety of host and target machines: Unix 68000, VAX and PDP-11 and can produce code for any of those machines, plus the 8086.
- Collection of programs designed to simplify the task of producing portable compilers and interpreters.
- The Kit contains complete internals documentation describing how to make modifications needed to add a new program language or a new target machine.
- Cross assemblers are also provided for 8080, Z80, Z8000, 8086, 6800, 6809, 68000, 6502, and PDP-11. Cross interpreters, for testing, are also included.
- The Kit consists of 8 components: preprocessor, front end, peephole optimizer, global optimizer, back end, target machine optimizer, universal assembler, utility package.
- The Kit includes all sources and documentation for VAX/Unix 4.1/4.2, PDP-11/V7, and MC68000 implementations.

Source
Price: Full system $9950
Educational Institution $995

Source requires an AT&T source compiler license

Binary
Full system $4995
(contact us for specific availability)

OEM terms available • Much more Unix software, too! • Call or write for more information.

UniPress Software, Inc.
2025 Lincoln Highway, Edison, NJ 08817
201-985-8000 • Order Desk: 800-222-0550 (outside NJ)
Telex 709418 • Mastercard and Visa

Lattice is a registered trademark of Lattice, Inc. Unix is a trademark of Bell Laboratories. MS-DOS is a trademark of Microsoft.

CIRCLE 38 ON READER SERVICE CARD

NEW FOR THE MACINTOSH

PRESENTING THE MEGAMAX C COMPILER

FEATURING:
• IN-LINE ASSEMBLY • ONE PASS COMPILATION • SUPPORT OF DYNAMIC OVERLAYS • FULL ACCESS OF MACINTOSH TOOLBOX ROUTINES • AND MUCH MORE...

DEVELOPMENT SYSTEM PACKAGE INCLUDES:
• FULL-SCALE IMPLEMENTATION (K&R) C COMPILER • THE STANDARD C LIBRARY • ROM ROUTINES LIBRARY • LINKER • LIBRARIAN AND DOCUMENTATION...

DEALER AND USER GROUP INQUIRES INVITED

$299.95

FOR MORE INFORMATION OR TO ORDER CALL OR WRITE:
BOX 851521, DEPT. 1
RICHARDSON, TX 75085-1521

Megamax, Inc.
(214) 987-4931

MACINTOSH IS A REGISTERED TRADEMARK OF APPLE COMPUTER INC

CIRCLE 13 ON READER SERVICE CARD

WRITE

The Writer's Really Incredible Text Editor lives up to its name! It's designed for creative and report writing and carefully protects your text. Includes many features missing from WordStar, such as sorted directory listings, fast scrolling, and trial printing to the screen. All editing commands are single-letter and easily changed. Detailed manual included. Dealer inquiries invited. WRITE is $239.00.

BDS's C Compiler

This is the compiler you need for learning the C language and for writing utilities and programs of all sizes and complexities. We offer version 1.5a, which comes with a symbolic debugger and example programs. Our price is (postpaid) $130.00.

Tandon Spare Parts Kits

One door latch included, only $32.50.
With two door latches $37.50.
Door latches sold separately for $7.00.

All US orders are postpaid. We ship from stock on many formats, including: 8", Apple, Osborne, KayPro, Otrona, Epson, Morrow, Lobo, Zenith, Xerox. Please request our new catalog. We welcome COD orders.

Workman & Associates
112 Marion Avenue
Pasadena, CA 91106
(818) 796-4401

WORKMAN & ASSOCIATES USER-FRIENDLY SOFTWARE

CIRCLE 68 ON READER SERVICE CARD

EXOTIC LANGUAGE OF THE MONTH CLUB

Transaction Application Language

By Serg Koren and Pierre Provencher

A relatively recent advance in computer systems has been the development of fault-tolerant computers. One such system is the TANDEM machine—a multiprocessor, multiuser transaction-oriented minicomputer, which the manufacturer claims experiences minimal downtime. To take advantage of the NON-STOP (TANDEM trademark) capabilities of the system, TANDEM developed and implemented a high-level, block-structured language known as TAL (Transaction Application Language).

TAL is used by TANDEM to develop all of its system software and by programmers to write applications where performance is a consideration or where system-level access is needed. TAL is proof that communication and system module applications do not need to be developed at the machine-code level. This article will touch briefly on TAL's main points.

TAL has many similarities to Pascal and ALGOL. It has procedures, subprocedures, and functions and is strongly-typed— although not as strongly as Pascal. TAL also has a standard I/O and function library as well as the ability to access the bit level.

TAL comments are delimited by !. Like Pascal, TAL permits local and global variables and has the same *BEGIN END* structure. A TAL identifier is up to 31 characters long and starts with a letter. The ^ is considered a letter permitting readable identifiers:

```
int line^buffer^of^terminal[0:39];
```

TAL's assignment operator is := and = represents a logical test. Assignment can be used after a variable declaration, making initialization easy. (Strangely, literals use = for assignment). Using apostrophes around the assignment statement transforms assignment into a move of multiple elements. Assignment can also be chained.

```
array1':=' array2 for 4;
a:=b:=c:=d;
```

One of the unique features of the language allows the user to interface non-standard hardware or to temporarily override the original sysgen configuration of a piece of hardware.

TAL is compiled in one pass and generates true TANDEM machine code (a custom processor board, not a CPU). The language allows calls to COBOL, FORTRAN, and library routines. In addition, TAL routines are callable by both COBOL and FORTRAN.

Procedures (known as PROCs) in TAL have their own local data area and can be called recursively. Like most other procedure-oriented languages, TAL's PROCs can pass or receive parameters. Parameters can be passed as a reference or a value although arrays are always passed by reference. A procedure or a function can also be passed as a parameter. A typical PROC shell is presented in Listing 1, with a typical call being,

```
CALL Sampleproc(A,B);
```

In addition, PROCs can return a value which makes the PROC equivalent to functions in other languages. To do this the procedure is given a type, and a *RETURN* statement is included in the body of the procedure which will return a value of that type. The procedure in Listing 2 can be called with

```
Result := Typedproc;
```

Procedures may also be assigned attributes. For example, the attribute RESIDENT indicates that the portion of code for this procedure should not be swapped out of memory. The procedure where execution begins is known as the main PROC

```
PROC Sampleproc(Param1,Param2);

    INT Param1,
        Param2;
BEGIN

    INT Localvariable1,
        Localvariable2;

       <code here>

END;       ! end of proc
```

Listing 1.

```
INT PROC Typedproc;          !without parameters
BEGIN
     <code>
     RETURN Intvalue;        !where Intvalue is global here
     <more code>
     RETURN Intvalue;        !you can have more than 1 RETURN
END;
```

Listing 2.

Thunder Software

• **The THUNDER C Compiler** - Operates under the APPLE Pascal 1.1 operating system. Create fast native 6502 programs to run as stand alone programs or as subroutines to Pascal programs. A major subset of the C defined by K & R. Includes a 24 page users guide, newsletters. Macro preprocessor, runs on APPLE][,][+, //e, //c. Source code for libraries is included. **Only $49.95**
• **ASSYST: The Assembler System** - A complete 6502 editor/assembler and lister for APPLE DOS 3.3. Menu driven, excellent error trapping, 24 p. users guide, demo programs, source code for all programs! Great for beginners. **Only $23.50**
• **THUNDER XREF** - A cross reference utility for APPLE Pascal 1.1. XREF generates cross references for each procedure. Source code and documentation provided. **Only $19.95**

Thunder Software POB 31501 Houston Tx 77231 713-728-5501
Include $3.00 shipping. COD, VISA and MASTERCARD accepted

CIRCLE 65 ON READER SERVICE CARD

MEMO: *C Programmers*

QUIT WORKING SO HARD.

These people have quit working so hard: IBM, Honeywell, Control Data, GE, Lotus, Hospitals, Universities & Government Aerospace.

THE GREENLEAF FUNCTIONS™

THE library of C FUNCTIONS that probably has just what you need . . . **TODAY!**

. . . already has what you're working to re-invent
. . . already has over 200 functions for the IBM PC, XT, AT, and compatibles
. . . already complete . . . already tested . . . on the shelf
. . . already has demo programs and source code
. . . already compatible with all popular compilers
. . . already supports all memory models, DOS 1.1, 2.0, 2.1
. . . already optimized (parts in assembler) for speed and density
. . . already in use by thousands of customers worldwide
. . . already available from stock (your dealer probably has it)
. . . It's called the **GREENLEAF FUNCTIONS.**

Sorry you didn't know this sooner? Just order a copy and then take a break — we did the hard work. Already.

THE GREENLEAF FUNCTIONS GENERAL LIBRARY: Over 200 functions in C and assembler. Strength in DOS, video, string, printer, async, and system interface. All DOS 1 and 2 functions are in assembler for speed. All video capabilities of PC supported. All printer functions. 65 string functions. Extensive time and date. Directory searches. Polled mode async. (If you want interrupt driven, ask us about the **Greenleaf Comm Library**.) Function key support. Diagnostics. Rainbow Color Text series. Much, much more. **The Greenleaf Functions.** Simply the finest C library (and the most extensive). All ready for you. From Greenleaf Software.

. . . **Specify compiler** when ordering. Add $7.00 each for UPS second-day air. MasterCard, VISA, check, or P.O.

◆ Compilers:	◆ General Libraries. . . . $175
CI C86 . . . . $349	(Lattice, Microsoft, Mark Williams, CI C86)
Lattice $395	◆ DeSmet C. $150
Mark Williams . . . $475	◆ Comm Library $160

GREENLEAF SOFTWARE©

GREENLEAF SOFTWARE, INC.
2101 HICKORY DRIVE • CARROLLTON, TX 75006 • (214) 446-8641

CIRCLE 44 ON READER SERVICE CARD

and is designated by the attribute *MAIN* following the PROC name:

```
PROC Starthere MAIN;
```

In addition to one procedure calling another, TAL allows PROCs to contain and call SUBPROCs. Just as with PROCs, SUBPROCs can pass parameters and be typed. Similarly, SUBPROCs have their own local data space and attributes.

TAL has the standard loop constructs found in other languages: the *FOR*, *DO UNTIL*, and *WHILE DO*. The *FOR* loop can be ascending (*TO*) or descending (*DOWNTO*). A primitive *CASE* is also provided. The *CASE* statement works with an index variable that starts at zero. *OTHERWISE* can be used to assure that a value greater than the last case index specified will have predictable results.

```
CASE indexvariable of
    begin
    a:=a+1; !indexvariable = 0
    b:=b+1; !indexvariable = 1
    otherwise c:=c+1;
                    !indexvariable > 1
    end;
```

Although it is confusing and not structured, TAL allows the use of *DEFINE*. The *DEFINE* assigns a portion of program source text to an identifier.

```
DEFINE total= count:=count+1 #;
    . . . . .
    if new^customer then total;
```

One unique feature of the TAL *IF* statement is that it can be used in an assignment statement:

```
Result := IF Thisistrue THEN 3 ELSE 0;
```

which will assign a value of 3 if the variable *Thisistrue* is logically true. If *Thisistrue* is logically false, a value of 0 will be assigned. As long as an expression can be evaluated as logically true or false, it can be imbedded within an *IF*.

```
IF (X := Y - Z) THEN CALL Myproc;
```

will evaluate the expression within parentheses. If the resultant value of X is greater than zero then the expression is true and the *CALL* will be executed.

TAL handles six data types, single-word (16 bit), double-word, strings (8-bit bytes), fixed point (64 bit), and reals (64 or 32 bit). Individual data types can be converted to any other through built-in type transfer functions. Data declaration can be specified as direct or indirect addressing (Listing 3). A TANDEM word is 16 bits.

A dot before an identifier indicates standard indirect addressing. Indirect addressing was introduced because of a

TANDEM restriction on memory addressing by a program. Without using extended addressing, 64K work is available for the data area of a TAL program. Only the first 32K is byte addressable. Standard indirect addressing provides access to the 64K while extended addressing (increasing the address to 32 bits from 16 bits) permits the user to define a larger memory segment.

For extended addressing it is the programmer's responsibility to manage memory through *ALLOCATESEGMENT* and *DEALLOCATESEGMENT* calls. The individual address of a variable can be accessed by prefixing the @ operator to the identifier.

Any data variable (except literals) can be specified as a one-dimensional array by specifying bounds. Multiple dimensional arrays are not permitted. Any variable can be indexed as an array. A pointer is an indirect addressing variable with no dimension. The compiler does not build run-time boundary checks even if bounds are specified.

```
int .pointer; !is a standard pointer
int .ext top; !is an extended
              !address pointer
```

Variables and arrays can be defined as read only and then are coded in-line instead of in the data area.

```
string
  .three^letters[0:2] = 'P' := [''abc''];
```

TAL also allows the user to perform block moves of data through the use of the move statement. For example, consider the code segment presented in Listing 4.

This example is a bit strange but none the less sometimes useful. Again, you can only move data of the same type although, as in the last example, the system does not check variable bounds.

Numerous data conversion functions are available to transform one data type to another. The built-in function *CODE* enables entry of machine assembler in the program and can be useful to write such things as a trap handling routine.

After an I/O or a GUARDIAN call, the condition code of the operation can easily be checked using an *IF* statement with no test variable.

```
if < then call error;
!the condition code is less than 0
```

Bit manipulations can only be done on integers and are useful for bit maps, etc. The examples presented in Listing 5 show the major statement types. TANDEM numbers the bits from the high-order bit (bit 0 is the high-order bit). A test on bit

```
literal
    true = 1,                              !definition of
    false = 0;                             ! constants
int
    counter:=0,                            !direct integer
   .array^of^ten^elements[0:9];            !indirect array
int(32)
    dblword;                               !double integer
string
   .a^to^c:=["abc"];                       !string declaration
```

Listing 3.

```
INT A,
    C;
STRING B[0:3];  !a 4 byte string array

C := 1;
B ':=' "TEST";  !would "move" the 4 bytes into the array
A ':=' C FOR 2; !would "move" the 2 words starting at variable
                !C into the memory starting at variable A.
                !A would then have the value 1 and C would have
                !the value of "TE".
```

Listing 4.

VEDIT PLUS

Easy to Use Word Processor with the Power of BASIC

VEDIT has been acclaimed for the last four years the industry standard in text editing.

InfoWorld Software Report Card

VEDIT 1.36

	Poor	Fair	Good	Excellent
Performance	□	□	□	☑
Documentation	□	□	□	☑
Ease of Use	□	□	☑	□
Error Handling	□	□	□	☑

Now there's VEDIT PLUS.

VEDIT PLUS is easy to learn, yet can do things far beyond ordinary word processors:

- Sort a mailing list
- Perform arithmetic computations
- Compare files
- Edit mulitple files of unlimited size

The power of VEDIT PLUS is equal to a high-level language - it gives you:

- If-then-else decision making
- Pattern matching
- User prompts and input
- An optional Z80 to 8086 translator

Expect a lot from VEDIT PLUS. It's from CompuView - nationally recognized for user support.

For a demo disk, more information or the dealer nearest you call toll free:

1-800-327-5895

VEDIT PLUS is available for practically every microcomputer . . . $225

CompuView
PRODUCTS, INC.

1955 Pauline, Ann Arbor, Mi 48103
(313) 996-1299 • (800) 327-5895

CIRCLE 40 ON READER SERVICE CARD

15 is an easy way to determine if the value is even or odd.

```
word1.<8:9>:=word2.<10:11>;
if word.<15> then ...
```

TAL allows the software developer to use address pointers to either byte or word data as the following shows:

```
INT .Indirectvariable[0:5];
              !six word array
STRING .Stringpointer :=
      @Indirectvariable '<<' 1;
```

The @ indicates "address of" and '<<' 1 does a logical left bit shift of 1 bit. This effectively multiplies the integer address by 2 in order to get the address of the string. This is one way of dealing with integers as if they were strings. The statement

```
Stringpointer[1] := "A";
```

would set the rightmost byte of the first word of *Indirectvariable* to the letter A.

Since TAL is used to write all of TANDEM's compilers and interpreters, a string search facility is provided in the form of a *SCAN* statement. *SCAN* tests a string of any length until either a target character or an ASCII null is encountered. The address of the target character is returned as a result of the *SCAN*. Whereas *SCAN* searches strings from left to right, a similar command, *RSCAN*, does so from right to left. The form of the syntax for *SCAN* is:

```
SCAN <string array> WHILE (or
    UNTIL) <target character> ->
    <next address>
```

In all TAL has 19 different types of statements, none of these dealing with I/O. All I/O is handled through a standard library of PROCs. A typical I/O statement is *READ*, and appears as a *CALL* in the body of a user PROC. For example:

```
CALL READ(filenum,filebuf,
    readcount,countread);
```

is typical. Filenum is the file to be read, filebuf is an integer array into which data is to be read, readcount is the number of bytes to read, and countread is returned by the *READ PROC* and indicates the actual number of bytes read. All other I/O is handled similarly.

TAL also allows the user to code applications that communicate with each other by means of inter-process messages. Library routines are provided to handle these. *AWAITIO* is a PROC that either awaits an I/O completion, specifies a time-out period for an I/O, or checks for a completion of an I/O. This PROC is used to code applications where processing need not wait for I/Os, where I/Os must be funneled, where I/O timing is critical, etc.

Library routines are provided for creation, deletion, and system management

```
A.<1> := 1;  ! set bit one of variable A
A := A.<3:6>;! set the value of A to the value of bits 3 thru 6
             ! bit extraction
IF A.<4> THEN RETURN; ! bit test
```

Listing 5.

RP/M T.M.

By the author of Hayden's "CP/M Revealed."

New resident console processor RCP and new resident disk operating system RDOS replace CCP and BDOS without TPA size change.

User 0 files common to all users; user number visible in system prompt; file first extent size and user assignment displayed by DIR; cross-drive command file search; paged TYPE display with selectable page size. SUBMIT runs on any drive with multiple command files conditionally invoked by CALL. Automatic disk flaw processing isolates unuseable sectors. For high capacity disk systems RDOS can provide instantaneous directory access and delete redundant nondismountable disk logins. RPMPIP utility copies files, optionally prompts for confirmation during copy-all, compares files, archives large files to multiple floppy disks. RPMGEN and GETRPM self-install RP/M on any computer currently running CP/M®2.2. Source program assembly listings of RCP and RDOS appear in the RP/M user's manual.

RP/M manual with RPMGEN.COM and GETRPM.COM plus our RPMPIP.COM and other RP/M utilities on 8" SSSD $75. Shipping $5 ($10 nonUS). MC, VISA.

118 SW First St. - Box G
Warrenton, OR. 97146

Micro Methods, Inc.

(503) 861-1765

CIRCLE 52 ON READER SERVICE CARD

OPT-TECH SORT™

SORT/MERGE program for IBM-PC & XT

Now also sorts dBASE II files!

- Written in assembly language for **high performance**
  Example: 4,000 records of 128 bytes sorted to give key & pointer file in 30 seconds. **COMPARE!**
- Sort ascending or descending on up to nine fields
- Ten input files may be sorted or merged at one time
- Handles variable and fixed length records
- Supports all common data types
- Filesize limited only by your disk space
- Dynamically allocates memory and work files
- Output file can be full records, keys or pointers
- Can be run from keyboard or as a batch command
- Can be called as a subroutine to many languages
- Easy to use, includes on-line help feature
- Full documentation — sized like your PC manuals
- **$99** —VISA, M/C, Check, Money Order, COD, or PO
  Quantity discounts and OEM licensing available

To order or to receive additional information write or call:

OPT-TECH DATA PROCESSING
P.O. Box 2167 Humble, Texas 77347
(713) 454-7428
Requires DOS, 64K and One Disk Drive

CIRCLE 67 ON READER SERVICE CARD

of other processes, files, etc. If additional efficiency or capability is necessary, TAL allows for actual TANDEM assembly code to reside within TAL PROCs. Device control is another major TANDEM function—everything from laboratory equipment to automatic tellers in banking (a common application).

SETMODE is the TAL library PROC that allows the user to temporarily override the device characteristics originally specified during sysgen. This procedure allows the user to code an application such that any device used by the application becomes virtually port-independent.

For example, if a data communications line is normally sysgened as being 9600 baud with 7 data bits, an application could alter this to 300 baud with 8 data bits for special communications needs. *SETMODE* can handle printers, terminals, communications lines, etc. and lets the user alter a wide range of characteristics.

Tal becomes a really powerful tool when associated with the GUARDIAN operating system. File access, process control and checkpointing are not defined in the language itself, but the calls needed are sourced in the TAL program. This gives serious advantages: only the needed procedures are sourced in by the programmer, the guardian routines are not reserved TAL identifiers, and the program stays structured since those calls are treated as procedures.

As is the case with most languages, TAL is still evolving to accommodate new uses, hardware, etc. Although TAL is usable only on TANDEMs, it is a highly structured and powerful language with a wide following. Part of the strength of TAL lies in the simplicity of its statement types and part in the wide range of standard library procedures provided. ■

Serg Koren is a professional programmer with Smith-Kline Clinical Laboratories and has been using TAL for six years.

Pierre Provencher graduated with a degree in computer science from the Univ. of Montreal, Que. He has been consulting for online full tolerance systems for five years.

DSD 80

The Dynamic Screen Debugger for CP/M

DSD-80 is the most advanced debugging program available for CP/M-80 and compatible operating systems. Spend less time debugging and more time programming because DSD is the only utility to offer all of the most important features in one package.

Satisfaction is guaranteed or a full refund!

Full featured simulator provides for:
- Execution only within boundaries
- Write protected memory
- Stack overflow protection
- Stack underflow protection
- Real time subroutines
- Breakpoint on specified memory or register values

Full screen display includes:
- 2 memory displays
- Register display
- Stack display
- Instruction display

Fully upward compatible with DDT

Single keystroke commands for:
- Stepping instructions
- Subroutine execution
- Scrolling memory displays

Full symbol support including:
- Loading symbol files
- Interactive symbol definitions

Endorsed by Leor Zolman — author of BDS C

Uses less than 15k of memory

On line help information

Clearly written 50 page manual

Free updates for one year

Only $195.00

SOFTADVANCES

P.O. Box 49473 Austin, TX 78765 (512) 478-4763

Available on 8" SSSD & various 5.25" formats
IBM PC version available soon!
Visa & MC accepted

CIRCLE 77 ON READER SERVICE CARD

UNPARALLELED **PERFORMANCE** and **PORTABILITY** in an **ISAM** PACKAGE at an **UNBEATABLE** PRICE

c-tree™
BY FAIRCOM

2606 Johnson Drive
Columbia MO 65203

The company that introduced micros to B-Trees in 1979 and created ACCESS MANAGER™ for Digital Research, now redefines the market for high performance, B-Tree based file handlers. With c-tree™ you get:

- complete C source code written to K & R standards of portability
- high level, multi-key ISAM routines and low level B-Tree functions
- routines that work with single-user and network systems
- no royalties on application programs

$395 COMPLETE

Specify format:
8" CP/M® 5¼" PC-DOS 8" RT-11

for VISA, MC or COD orders, call
1-314-445-6833

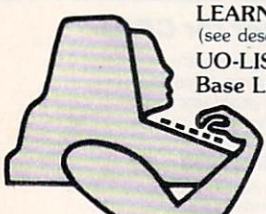
Access Manager and CP/M are trademarks of Digital Research, Inc.
c-tree and the circular disc logo are trademarks of FairCom

©1984 Faircom

CIRCLE 29 ON READER SERVICE CARD

(LISP) FOR A.I.

UO-LISP Programming Environment
The Powerful Implementation of LISP for MICRO COMPUTERS



LEARN LISP System (LLS.1)
(see description below) *$39.95*

UO-LISP Programming Environment Base Line System (BLS.1) *$49.95*

Includes: Interpreter, Compiler, Structure Editor, Extended Numbers, Trace, Pretty Print, various Utilities, and Manual with Usage Examples. **(BLS.1)** expands to support full system and products described below.

UO-LISP Programming Environment: The Usual LISP Interpreter Functions, Data Types and Extensions, Structure & Screen Editors, Compiler, Optimizer, LISP & Assembly Code Intermixing, Compiled Code Library Loader, I/O Support, Macros, Debug Tools, Sort & Merge, On-Line Help, Other Utility Packages, Hardware and Operating System Access, Session Freeze and Restart, Manual with Examples expands to over 350 pages. Other UO-LISP products include: LISPTEX text formatter, LITTLE META translator writing system, RLISP high level language, NLARGE algebra system. Prices vary with configurations beyond (BLS.1) please *send for FREE catalog.*
LEARN LISP System (LLS.1): Complete with LISP Tutorial Guide, Editor Tutorial Guide, System Manual with Examples, Full LISP Interpreter, On-Line Help and other Utilities. LEARN LISP fundamentals and programming techniques rapidly and effectively. This system does not permit expansion to include the compiler and other products listed above.
LISP Tutorial Support (LTS.1): Includes LISP and Structure Editor Tutorial Guides, On-line Help, and History Loop. This option adds a valuable learning tool to the UO-LISP Programming Environment (BLS.1). *Order (LTS.1) for* **$19.95.**
REQUIRES: UO-LISP Products run on most Z80 computers with CP/M, TRSDOS or TRSDOS compatible operating systems. The 8086 version available soon.
TO ORDER: Send Name, Address, Phone No., Computer Type, Disk Format Type, Package Price, 6.5% Tax (CA residents only), Ship & Handle fee of $3.00 inside U.S. & CN, $10 outside U.S., Check, Money Order, VISA and MasterCard accepted. With Credit Card include exp. date. Other configurations and products are ordered thru our *FREE catalog.*

Northwest Computer Algorithms
P.O. Box 90995, Long Beach, CA 90809 (213) 426-1893

CIRCLE 46 ON READER SERVICE CARD

C Source Code

RED

Full Screen Text Editor
IBM PC, Kaypro, CP/M 80 and CP/M 68K systems.

- RED is fast! RED uses all of your terminal's special functions for best screen response. RED handles files as large as your disk automatically and quickly.
- RED is easy to use for writers or programmers. RED's commands are in plain English.
- RED comes with complete source code in standard C. RED has been ported to mainframes, minis and micros.
- RED comes with a Reference Card and a Reference Manual that provides everything you need to use RED immediately.
- RED is unconditionally guaranteed. If for any reason you are not satisfied with RED your money will be refunded promptly.

RED: $95

Manual: $10

edward k. ream

Call or write today for for more information:
Edward K. Ream
1850 Summit Avenue
Madison, WI 53705
(608) 231-2952

To order:
Either the BDS C compiler or the Aztec CII compiler is required for CP/M80 systems. Digital Research C compiler v1.1 is required for CP/M 68K systems. No compiler is required for IBM or Kaypro systems.

Specify both the machine desired (IBM, Kaypro or CP/M) and the disk format described (8 inch CP/M single density or exact type of 5¼ inch disk).

Send a check or money order for $95 ($105 U.S. for foreign orders). Sorry, I do NOT accept phone, credit card, or COD orders. Please do not send purchase orders unless a check is included. Your order will be mailed to you within one week.

Dealer inquiries invited.

Fortran Scientific Subroutine Package

Contains Approx. 100 Fortran Subroutines Covering:

1. Matrix Storage and Operations	7. Time Series
2. Correlation and Regression	8. Nonparametric Statistics
3. Design Analysis	9. Distribution Functions
4. Discriminant Analysis	10. Linear Analysis
5. Factor Analysis	11. Polynomial Solutions
6. Eigen Analysis	12. Data Screening

Sources Included. Microsoft 3.2 compatible.
$295.00

FORLIB-PLUS™

Contains three assembly coded LIBRARIES plus support, FORTRAN coded subroutines and DEMO programs.

The three LIBRARIES contain support for GRAPHICS, COMMUNICATION, and FILE HANDLING/DISK SUPPORT. An additional feature within the graphics library is the capability of one fortran program calling another and passing data to it. Within the communication library, there are routines which will permit interrupt driven, buffered data to be received. With this capability, 9600 BAUD communication is possible. The file handling library contains all the required software to be DOS 3.0 PATHNAME compatible.

$69.95

Strings & Things ™
Character Manipulation and Much More!
$69.95

ALPHA COMPUTER SERVICE
P.O. Bos 2517
Cypress,CA 90630 **(714) 894-6808**
California residents, please add 6% sales tax

CIRCLE 2 ON READER SERVICE CARD

Announcing a

TOTAL PARSER GENERATOR

<GOAL> :: = <RAPID> <COMPILER> <DESIGN>

SLICE YOUR COMPILER DEVELOPMENT TIME

An LR(1) parser generator and several sample compilers, all in Pascal for your microcomputer.

- Generates parser, lexical analyzer and skeleton semantics
- Universal, state-of-the art error recovery system
- Adaptable to other languages
- Interactive debugging support
- Thorough documentation
- TURBO PASCAL™ INCLUDED FREE OF CHARGE
- Includes mini-Pascal compiler, assembler, simulator in SOURCE

SPECIAL INTRODUCTORY OFFER $1995

QPARSER™ runs on IBM PC/DOS in Turbo Pascal. Parser generator in object form, all else in source. QPARSER takes a grammar and generates a correct, complete, high-performance compiler with skeleton semantics in Pascal source. Easy to add full semantics for YOUR application. Excellent for industrial and academic use. An accompanying textbook (SRA publishers) available in 1985. Training can be arranged. **Demo disk available for $50.**

Educational and quantity discounts available. Check, money order, Mastercard, Visa. California residents add 6.5% sales tax.

WRITE OR CALL FOR FREE BROCHURE.
Technical details: call 408/255-5574. Immediate delivery. CALL TODAY!

QCAD SYSTEMS, INC.
1164 Hyde Ave., San Jose, CA 95129
TOLL FREE: 800-538-9787
(California residents call 408/255-5574)

™ Turbo Pascal is a registered trademark of Borland International.

CIRCLE 23 ON READER SERVICE CARD

Product reviews on CompuServe and the COMPUTER LANGUAGE Bulletin Board Service

Trio System's C-Index+	Spruce Technology's FirsTime
System Guild's CGraph	Borland's SideKick
Creative Solution's MacForth	Hendrix's Small-Tools
Waltz LISP	Nevada COBOL
CONIX	Harvard Softwork's Forth
Bendorf's Professional Programming Environment	

BetterBASIC

Hardware Requirements: One or more disk drives, 128K of memory (192K or more recommended) for IBM PC or 100% compatibles: DOS 1.0—2.1
Price: $199, $99 for 8087 math module, $250 for run-time system
Available from: Summit Software Technology Inc., 40 Grove St., Wellesley, Mass. 02181, (617) 235-0729
Support: Free updates and support by staff

BetterBASIC is a serious attempt to make a structured language out of BASIC. Some unusual control structures have been added, and the result is a language that feels like a cross between BASIC and Pascal with some of the strengths and weaknesses of each.

Overall, BetterBASIC is fast, modular, extensible, and flexible. It is also somewhat nonstandard in its syntax, more complicated to configure, requires much more memory than the standard Microsoft BASIC supplied with the IBM PC, and has less explanatory error messages.

Whether BetterBASIC is a language suited for program development depends very much on the job you want to do. For many if not most business applications BetterBASIC certainly merits a close look.

I was supplied with BetterBASIC version 1.0, which contained 16 files. Instead of CONSOLE.IBM, mentioned in the documentation, the review distribution had module WINDOWS.IBM. (CONSOLE.IBM allows use of the keyboard edit keys, COLOR, LOCATE, and CLS functions.)

The manual (on page 16—not on the page showing the configuration files) states that the WINDOWS module is a replacement for the CONSOLE module. It would have been nice to be able to use the CONSOLE.IBM module to see what the differences were.

BetterBASIC can be run without WINDOWS; however, the lack of this module prevents you from running the demonstration programs. If I tried to run them, I was told that the demonstration programs weren't programs, not that I lacked a required module. This error message is not only misleading—it's downright wrong.

I can't know how big the CONSOLE.IBM file was, but if it were much larger than about 10K, it wouldn't have fit on the single-sided DOS 1.1 distribution disk. Since the graphics module takes about 10K, it's possible that the CONSOLE.IBM module would have fit. Unfortunately, the graphics module requires the color graphics monitor; I don't have one and neither do most business users. Therefore the GRAPHICS module wasn't tested.

Once you've configured BetterBASIC for your computer, all that's necessary in order to run BetterBASIC is to type the single letter B and a return. BetterBASIC will load, read the B.CNF file, and load the modules you specified.

Loading BetterBASIC takes about 10 sec from a floppy disk— slower than IBM's BASIC but understandably so. There is a lot more to load.

Summit isn't kidding about BetterBASIC needing 192K of memory. If you have a 128K machine, you're going to have only about 15K of memory available even if you just use the mandatory modules and the FILE.DOS module. Since most machines are now being sold with 256K of memory, most users should have few problems.

The configuration file can contain, in addition to the list of modules you want, commands to set the status line on or off (requires CONSOLE.IBM to work so I couldn't test this); default numeric precision (4 to 24 digits, default is 8); stack size needed (default is 640 bytes); and the amount of memory to reserve for other purposes (default is 0).

The last command illustrates an important difference between BetterBASIC and IBM BASIC—BetterBASIC can use all available memory. Because of BetterBASIC's modular design and because BetterBASIC encourages modular programming, this added memory can be very useful in constructing programming applications.

Each module and subprogram, as well as each array used, cannot be larger than 64K bytes though. I suspect that restriction is imposed by the 8088/8086 chip for which BetterBASIC was written. That's too bad, but it's hardly Summit's fault. If anything, it's another argument against using chips that force segmented architecture.

The stack size may be increased if you're planning on doing much recursion. That's right: BetterBASIC supports recursion. Things keep looking better and better!

BetterBASIC will recognize and correctly execute nearly all of IBM BASIC's commands and statements. In addition, BetterBASIC has some useful new commands like *UPPER$* and *LOWER$* (convert a string to upper or lower case), *SPAN* (counts the number of characters in a string contained in the character set defined by another string), *WOR* (selects a 16-bit word at a memory location), *OFFSET* (returns the offset address of a given datum), and *SEG* (returns the segment address of a given datum). BetterBASIC also adds shift and pointer operators to BASIC.

BetterBASIC adds still more commands and functions, some of which I'll refer to in the course of this review.

Some of IBM BASIC's commands are not supported. *FRE(X)*, which returns the amount of available memory, has no equivalent in BetterBASIC. Nor is there any way to obtain the amount of memory available while running BetterBASIC. The manual suggests that *SIZE* is equivalent—it isn't. *SIZE* returns the space used by a variable, not the amount of memory available.

ERL isn't supported. *LINE INPUT* can't include a prompt. *TRON* and *TROFF* aren't supported. *WHILE . . . WEND* are replaced by *DO . . . REPEAT* or *DO . . . END DO*. Of these, the missing *TRON, TROFF,* and *ERL* are perhaps the most

C

SOFTWARE DEVELOPERS!

V - FILE
THE VIRTUAL MEMORY FILE MANAGER

Let V-FILE save precious development time & cost as you create efficient applications with the power of VIRTUAL MEMORY.

DON'T RE-INVENT THE WHEEL

Why spend weeks or months coding and debugging file and memory management systems when you can order V-FILE today. V-FILE is a library that you can link with your code to provide sophisticated virtual file and memory management — allowing you to concentrate on developing your application.

VIRTUAL DATA OBJECTS SUPPORTED!

Data is referenced by using VIRTUAL MEMORY DATA HANDLES. Your code doesn't need to know whether the data is actually on disk or in RAM. Swapping between disk and RAM and updating files on disk is handled automatically and transparently! Complex VIRTUAL DATA STRUCTURES can be created by linking with data handles instead of pointers.

CHECK THESE FEATURES!

- Multiple, independent swap buffers
- Multiple files per swap buffer
- Highly efficient swap algorithm
- Automatic file updating
- Data prefetching supported
- Data may be locked in memory
- Memory buffers may be flushed
- Makes full use of extended memory on IBM PC/AT
- SOURCE CODE AVAILABLE
- NO ROYALTIES REQUIRED

Supports Dos 2.00+ with Lattice & Microsoft C compilers
Supports Microsoft windows

V·FILE TM

$299

Contact:
MindBank, Inc.
4620 Henry Street
Pittsburgh, PA 15213
412/683-9800

VISA/MASTER CARD ACCEPTED

CIRCLE 63 ON READER SERVICE CARD

Eco-C Compiler
Release 3.0

We think Rel. 3.0 of the Eco-C Compiler is the fastest full C available for the Z80 environment. Consider the evidence:

Benchmarks*
(Seconds)

Benchmark	Eco-C	Aztec	Q/C
Seive	29	33	40
Fib	75	125	99
Deref	19	CNC	31
Matmult	42	115	N/A

**Times courtesy of Dr. David Clark*
CNC - Could Not Compile
N/A - Does not support floating point

We've also expanded the library (120 functions), the user's manual and compile-time switches (including multiple non-fatal error messages). The price is still $250.00 and includes Microsoft's MACRO 80. As an option, we will supply Eco-C with the SLR Systems assembler - linker - librarian for $295.00 (up to six times faster than MACRO 80).

For additional information, call or write:

ECOSOFT INC. (317) 255-6476
6413 N. College Ave. • Indianapolis, Indiana 46220

VISA MasterCard

CIRCLE 22 ON READER SERVICE CARD

YOUR CODE MAY BE WASTING ITS TIME!
THE PROFILER™ CAN HELP . . .

- **Statistical Execution Profiler**
- **Works with any language**
- **Completely configurable**
- **Up to 16 partitions in RAM/ROM**
- **Time critical code optimization**
- **Abnormal code behavior tracking**
- **Graphic presentation of results**
- **Easy to use menu interface**

THE PROFILER is a software package which gives you, the programmer, a powerful tool for locating time consuming functions in your code and allows you to performance tune your program. With the **THE PROFILER** you can determine where to optimize your code for maximum benefit, then measure the results of your efforts.

Using **THE PROFILER**, you can answer questions like:
Where is my program spending its time?
Why is my program so slow? What is it doing?
Is my progam I/O bound? CPU bound? Are data buffers large enough?
How much improvement did my changes make?

THE PROFILER is completely software based and consists of a system resident driver and a monitor program. The memory partitions can range from 1 byte to 1 megabyte in size and can be anywhere in the address space.

NO ADDITIONAL HARDWARE IS REQUIRED!

Requires an IBM PC or compatible system with a minimum 64k and one drive.

THE PROFILER is available for $175.00 from DWB Associates or ask your software dealer. To order or for more information, call or write DWB Associates. VISA/MC accepted. Dealers welcome.

IBM is a trademark of IBM Corp. MSDOS is a trademark of Microsoft Corp.
THE PROFILER is a trademark of DWB Associates.

dwb Associates
P.O. Box 5777
Beaverton, Oregon 97006
(503) 629-9645

CIRCLE 20 ON READER SERVICE CARD

serious; their absence will make programming more difficult. I strongly recommend that Summit include these commands in future releases of BetterBASIC.

The random file I/O commands are also different. BetterBASIC has no *FIELD* statement and no *CVI, CVS, CVD, MKI$, MKS$*, or *MKD$* functions. Here is a very important area where BetterBASIC shows quite intelligent design. Other language developers should profit from Summit's example.

Instead of using *FIELD* to allocate space in a file's I/O buffer, BetterBASIC uses a named structure to hold an accumulation of data. The design of the structure is similar to Pascal and C. An example given in the BetterBASIC manual will prove instructive:

```
STRUCTURE: Man
INTEGER: Age, Height, Weight,
    EmpNumber
REAL: Salary, SocSecNumber
STRING: FirstName, MiddleName,
    LastName
END STRUCTURE
```

Following this definition, you can declare variables of type Man named *Fred, Oscar*, and *Joe* and a 10-item array of records of structure Man:

```
Man: Fred, Oscar, Joe
Man ARRAY(10):People
```

A request to print Fred.Age will print the age field from the Man-type variable *Fred* in the array *People*. To write *Fred*'s record to a suitably-sized file at record Recnum, you could use the statement:

```
WRITE RECORD #1 Recnum Fred
```

An analogous *READ RECORD* command can be used to retrieve a record from a file into a variable in a named structure.

The only shortcoming I can see to this approach is that BetterBASIC's strings are limited to 16 bytes in length unless otherwise declared (declared strings can be up to 32K bytes in length). Certainly, the record approach to file I/O is more intuitive and more easily understood than the somewhat clumsy *FIELD* statement used in IBM's BASIC.

On the other hand, IBM could improve its approach substantially by allowing a *FIELD* statement to spread out over more than one line. Perhaps something like this would be appropriate:

```
FIELD #1, 2 AS AGE$, 2 AS HEIGHT$,
    2 AS WEIGHT$, 2 AS
    EMPNUMBER$
4 AS SALARY$, 4 AS
    SOCSECNUMBER$
25 AS FIRSTNAME$,25 AS
    MIDDLENAME$,25 AS
    LASTNAME$
FIELD END
```

Even if IBM's BASIC were to be modified in such as fashion, the result would not be as easy to use and understand as BetterBASIC's approach. Plus BetterBASIC's approach to reading and using string data (in or out of a record structure) permits the programmer to actually treat any string as though it were input data. This approach has another important strength which can only be compared in flexibility to C's formatted input operation.

BetterBASIC sure seemed a lot faster than IBM BASIC in operation. To see just how fast BetterBASIC was, I wrote some small benchmark programs. Except where noted, the IBM BASIC versions of the programs are identical to the line-numbered part of the BetterBASIC version. BetterBASIC supplied the indentation—a nice touch. The programs were typed in without any attention to formatting and without the un-numbered lines.

Test 1 is a simple empty loop. In BetterBASIC, here's the listing:

```
SOURCE
PROCS=0
INTEGER: I%
      5 PRINT TIME$
      10 FOR I%=1 TO 10000
      20 NEXT
      30 PRINT TIME$
ENDFILE
```

3,500 Programmers depend on us
to find, compare, evaluate products and for *solid value.*

THE PROGRAMMER'S SHOP serves serious microcomputer programmers . . . from giant institutions to small independents. *Specializing* helps us provide 100s of programming products . . . technical literature . . . specialized evaluations and more to help you find and evaluate. Other services like . . . special formats . . . rush delivery . . . payment options (POs, COD, credit cards, etc.) . . . newsletters . . . and reports *help you save time, money, and frustration and get* ***solid value.***

ARTIFICIAL INTELLIGENCE

EXSYS - Expert System building tool. Full RAM, Probability, Why, Intriguing, serious. PCDOS $200

GC LISP - "COMMON LISP", Help, tutorial, co-routines, compiled functions, thorough. PCDOS $475

TLC LISP - "LISP-machine"-like, all RAM, classes, turtle graphics, 8087 for CP/M-86, PCDOS or MSDOS $235

Expert System front-ends for PROLOG: APES ($275), ES/P ($1895)

Other solid alternatives include: IQ LISP ($155), MuLISP-86 ($250), WALTZ LISP for CPM ($159), MicroPROLOG ($275), PROLOG-86 ($125), more.

C PROGRAMMING

C SHARP Realtime Toolkit - well supported, source, thorough, portable, objects, state sys. $600

INSTANT C - Interactive development - Edit, Source Debug, run. Edit to Run - 3 Secs. MSDOS $500

"INTRODUCING C" - Interactive C to learn fast. 500 page tutorial, examples, graphics. PCDOS $95

MEGAMAX C - native Macintosh has fast compile, tight code, K&R, toolkit, .OBJ, DisASM MAC $295

SUPPORT PRODUCTS

BRIEF Programmer's Editor - undo, windows, powerful. PCDOS $195

PERISCOPE DEBUGGER - load after "bombs", symbolic. PCDOS $295

Call for a Catalog, literature on any product or a **free literature "Packet"** on: "AI", BASIC, C, COBOL, Debuggers, Editors, FORTH, FORTRAN, Libraries, PASCAL

CALL TOLL FREE 800-421-8006

THE PROGRAMMER'S SHOP™

The programmer's complete source for software, services and answers

128-L Rockland Street, Hanover, MA 02339 In Mass.: **800-442-8070** or 617-826-7531

CIRCLE 69 ON READER SERVICE CARD

PROGRAMMER'S DEVELOPMENT TOOLS

Over the past year we at Programmer's Connection have had the good fortune to meet and talk with hundreds of fellow programmers throughout the country. We want to extend our best wishes for a prosperous and productive New Year. And we invite you to call us for the software development tools you need in 1985.

When you call Programmer's Connection you will talk with an experienced programmer. Our people have many years of professional experience encompassing the design, programming, documentation and support of software systems ranging from large mainframes to personal computers. Hence our motto, "Programmers Serving Programmers."

Our philosophy is to offer knowledgeable and dependable service at the lowest possible price. Compare our ad with the competition . . . you'll see the difference. Call us today . . . you'll hear the difference.

Sincerely,

[illegible]

Michael P. Colarik
President,
Programmer's Connection

Best Wishes for 1985!

C LANGUAGE:	List	Ours
Computer Innovations C-86	395	309
DeSmet C Compiler with Debugger	159	145
Lattice C Compiler	500	299
Wizard C	450	409
Xenix Development System by SCO	1350	1099

***** Mark Williams C Compiler *****
Sale Priced at $429
This highly optimized compiler includes an extremely useful source-level debugger which can save you hours of programming time. A truly professional C development system with too many features to list.

OTHER LANGUAGES:		
8088 Assembler w/Z-80 Translator 2500 AD	100	89
BetterBASIC by Summit Software *Sale!*	200	149
Janus/ADA + Tools by R&R	700	499
Modula-2/86 by Logitech	495	439
STSC APL*Plus/PC *We Can Support You!*	595	499

*** Morgan Computing Professional BASIC ***
New low price $89
This structured BASIC semi-compiler with debugger supports large workspaces and many extensions. 8087 support $45.

UTILITIES:		
CodeSmith-86 by Visual Age	145	129
Communications Library by Greenleaf	160	139
C-Tree by Faircom	395	359
C To dBase by Computer Innovations	150	139
C Utility Library by Essential Soft.	149	129
ESP for C by Bellesoft	349	319
FirsTime for C by Spruce Tech	295	269
GraphiC from Scientific Endeavors	195	169
Greenleaf Functions Library	175	159
Halo *Color Graphics*	200	125
Panel *Screen Design/Editing* by Roundhill	295	234
Periscope *Debugger* by Data Base Decisions	295	269
Phact by Phact Associates	395	359
Plink-86 *Overlay Linkage Editor*	395	310
Pmate by Phoenix Software	225	175
Profiler by DWB & Associates	125	99
Screen Sculptor by Software Bottling	125	109
Windows For C by Creative Solutions	195	159

***** Scroll & Recall *****
$49
Save $20 on this DOS-resident Screen and Keyboard utility. S & R allows you to recall text that has scrolled off the screen. You can also recall and edit all of your previously entered DOS commands.

Prices are subject to change without notice.

Call for our New Catalog consisting of 200 + Programmer's Development Tools Exclusively for IBM PC's and Compatibles.

Account is charged when order is shipped.

VISA **1-800-336-1166** MasterCard

PC **Programmer's Connection**
136 Sunnyside Street
Hartville, Ohio 44632 (216) 877-3781 (In Ohio)
"**Programmers Serving Programmers**"

CIRCLE 62 ON READER SERVICE CARD

This first test ran in 5.2 sec under IBM BASIC and in 2.0 sec under BetterBASIC. A second test added a little more complication:

```
SOURCE
PROCS=0
INTEGER: I%,J%
    10 PRINT TIME$
    20 FOR I%=1 TO 10000
    25    J%=J%+1
    30 NEXT
    40 PRINT TIME$
ENDFILE
```

This test revealed an interesting difference between BetterBASIC and other versions of BASIC: BetterBASIC doesn't zero variables between runs. If you don't set the variable *J%* to zero between runs, eventually you'll get an overflow error in line 25. I don't know if this should be classified as a bug since a documented *CLEAR* function will zero all variables, but the difference should have been more clearly pointed out in the BetterBASIC manual.

Test 2 ran in 23.0 sec under IBM BASIC and 5.8 sec under BetterBASIC.

Test 3 added still more complications:

```
SOURCE
PROCS=0
INTEGER: I%,J%
    10 PRINT TIME$
    20 J%=0
    30 FOR I%=1 TO 10000
    40    J%=J%+1
    50    J%=J%*2
    60    J%=J%/2
    70    J%=J%-1
    80 NEXT
    90 PRINT TIME$
ENDFILE
```

BetterBASIC ran this in 19.0 sec. IBM BASIC took 100.6 sec, an advantage of over 5:1 for BetterBASIC. This series of tests completed my test of integer operations and simple looping for BetterBASIC.

The next series of tests covered real arithmetic operations. I immediately ran into a snag in BetterBASIC, something I can only ascribe to a Pascal-originated deficiency. Suppose you type in the immediate-mode statement:

```
INT 11/43
```

In IBM PC BASIC, you'll get the result .255814. In BetterBASIC, you'll get 0. However, if you use the statement:

```
PRINT 11/43.0
```

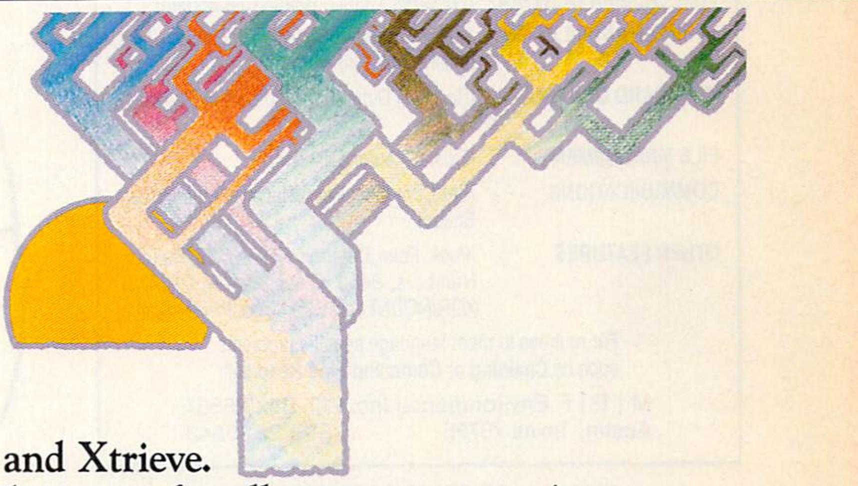
you'll get the correct result. This is contrary to all the rules I learned in every other version of BASIC I've used (over a dozen) and comparable in silliness only to Pascal. Pascal is famous for requiring added verbosity; part of BASIC's appeal

Benchmark speed comparison table

Test no.	IBM BASIC (interpreter)	BetterBASIC	IBM BASIC (compiled)
1	5.2	2.0	0.2
2	23.0	5.8	0.4
3	100.6	19.0	7.2
4	109.0	31.6	12.2
5	8.4	12.2/31.0[1]	2.2/6.8[1]
Average times	49.24	14.12	4.44

1. The result following the / is test 5 done with 16-digit precision.

Table 1.



Btrieve and Xtrieve. B-tree file access for all your programming languages. With the query tool your users demand.

Introducing a powerful data base combination for PC application developers. Btrieve™, the most sophisticated file access method for your IBM™ PC. In single user and network versions. And Xtrieve™, a new menu-driven query tool that gives you—and your users—fast access to information.

Btrieve: for professional programmers. Btrieve provides fast, flexible file management for all your application development. All your programming languages—BASIC, Pascal, Cobol, C. With multikey access to records. Automatic file recovery. Unlimited records per file. Duplicate, modifiable, and segmented keys.

With Btrieve, you can develop *better* applications *faster.*

Xtrieve: easy window interface. The ideal complement to Btrieve, Xtrieve is the non-programmer's interface. Xtrieve's full relational capabilities let users define a virtual table of data from multiple files. Then Xtrieve speeds them through query building with a series of easy-to-follow windows.

No command language. Xtrieve is completely menu-driven, so there's no need to memorize command language. Or special syntax. Everything you need is on the screen.

Xtrieve features a full range of restriction criteria. Online help messages. And interfaces to access information from Lotus 1-2-3™ and dBase II® files.

Network capabilities. Network versions of Btrieve and Xtrieve allow data sharing in PCnet™, NetWare™, EtherSeries™, MultiLink™, and OmniNet™ LANs.

For more information or to order, contact:

SoftCraft Inc.

P. O. Box 9802 #590 Austin, Texas 78766 (512) 346-8380

Suggested retail prices: Btrieve, $245; Xtrieve, $195. Btrieve/N (network), $595; Xtrieve/N, $395. Dealer inquiries welcome. Btrieve requires PC-DOS or MS™-DOS 1.X or 2.X; Xtrieve, PC-DOS or MS-DOS 2.X.

Btrieve and Xtrieve, IBM, 1-2-3, dBase II, PCnet, NetWare, EtherSeries, MultiLink, OmniNet, and MS are trademarks of SoftCraft Inc., International Business Machines, Lotus Development Corp., Ashton-Tate, Orchid Technology, Novell Data Systems, 3Com Corp., Davong Systems Inc., Corvus Systems, and MicroSoft Inc.

CIRCLE 78 ON READER SERVICE CARD

OVERCOME FORTRAN LIMITATIONS WITH

NO LIMIT FORTRAN

$89 Visa/MC

A library of 57 Assembler and FORTRAN routines transforms MS FORTRAN and other 8086/87/88 FORTRANs into the flexible, responsive, complete language needed for the microcomputer environment. Featuring:

EXTENSIVE GRAPHICS	(Get, Put, Paint, Color, Dot, Line, Box, Circle, Ellipse, Large Characters)
FULL SCREEN CONTROL	(Windows, Cursor, Read/Write Screen)
STRING MANIPULATION	(Match, Compare, Concatenate/Extract, Pack, Justify, Zero Fill)
KEYBOARD CONTROL	(Read Key During Execution, String Read With Edit)
FILE MANAGEMENT	(Exist?, Rename, Delete)
COMMUNICATIONS	(Set Com Line, Send/Receive, Line/Modem Status)
OTHER FEATURES	(Peek, Poke, Determine Time/Date, Random Numbers, Beep, Clear Screen, OR/AND/XOR/NOT/NEG of Byte/Word, Printer Status)

For routines to meet language specific problems such as **Chaining** or **Command Line Read** call:

M | E | F Environmental Inc. P.O. Box 26537
Austin, Texas 78755 (512) 251-5543

CIRCLE 90 ON READER SERVICE CARD

Scroll & Recall™

Screen and Keyboard Enhancement
for the IBM - PC, XT and Compatibles

Allows you to conveniently scroll back through data that has gone off the top of your display screen.

Allows you to easily recall and edit your previously entered DOS commands and data lines.

Very easy to use, fully documented. Compatible with all versions of DOS, monochrome & graphic displays.

$69 *- Visa, M/C, Check, COD, POs*
Phone orders accepted

Make Your Work Easier!

To Order or to Receive Additional Information, Write or Call:
Opt-Tech Data Processing
P.O. Box 2167 • Humble, Texas 77347
(713) 454-7428
Dealer Inquiries Welcome

CIRCLE 66 ON READER SERVICE CARD

SUPER FORTH 64®

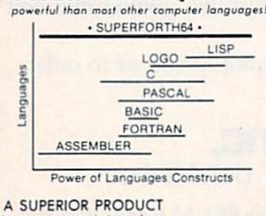
TOTAL CONTROL OVER YOUR COMMODORE-64™
USING ONLY *WORDS*
MAKING PROGRAMMING FAST, FUN AND EASY!

MORE THAN JUST A LANGUAGE...
A complete, fully-integrated program development system.
Home Use, Fast Games, Graphics, Data Acquisition, Business
Real Time Process Control, Communications, Robotics, Scientific, Artificial Intelligence

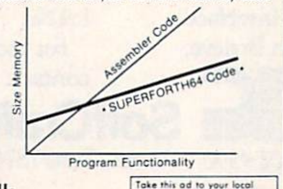
A Powerful Superset of MVPFORTH/FORTH 79 + Ext. for the beginner or professional

- 20 to 600 x faster than Basic
- 1/4 x the programming time
- Easy full control of all sound, hi res. graphics, color, sprite, plotting line & circle
- Controllable SPLIT-SCREEN Display
- Includes interactive interpreter & compiler
- Forth virtual memory
- Full cursor Screen Editor
- Provision for application program distribution without licensing
- FORTH equivalent Kernal Routines
- Conditional Macro Assembler
- Meets all Forth 79 standards+
- Source screens provided
- Compatible with the book "Starting Forth" by Leo Brodie
- Access to all I/O ports RS232, IEEE, including memory & interrupts
- ROMABLE code generator
- MUSIC-EDITOR
- SPRITE-EDITOR
- Access all C-64 peripherals including 4040 drive
- Single disk drive backup utility
- Disk & Cassette based. Disk included
- Full disk usage—680 Sectors
- Supports all Commodore file types and Forth Virtual disk
- Access to 20K RAM underneath ROM areas
- Vectored kernal words
- TRACE facility
- DECOMPILER facility
- Full String Handling
- ASCII error messages
- FLOATING POINT MATH SIN/COS & SQRT
- Conversational user defined Commands
- Tutorial examples provided, in extensive manual
- INTERRUPT routines provide easy control of hardware timers, alarms and devices
- USER Support

SUPER FORTH 64® *is more powerful than most other computer languages!*



SUPER FORTH 64® *compiled code becomes more compact than even assembly code!*



A SUPERIOR PRODUCT in every way! At a low price of only
$96

Call:
(415) 651-3160
PARSEC RESEARCH
Drawer 1776, Fremont, CA 94538

Take this ad to your local dealer, or B. Dalton Bookstore. Phone orders also accepted. Immediate delivery! Dealer inquiries invited. CA residents must include tax.

or C.O.D.

® PARSEC RESEARCH (Established 1976)   Commodore 64 & VIC-20 TM of Commodore

CIRCLE 49 ON READER SERVICE CARD

QUALITY SOFTWARE AT REASONABLE PRICES

CP/M Software by
Poor Person Software

Poor Person's Spooler	**$49.95**
All the function of a hardware print buffer at a fraction of the cost. Keyboard control. Spools and prints simultaneously.	
Poor Person's Spread Sheet	**$29.95**
Flexible screen formats and BASIC-like language. Preprogrammed applications include Real Estate Evaluation.	
Poor Person's Spelling Checker	**$29.95**
Simple and fast! 33,000 word dictionary. Checks any CP/M text file.	
aMAZEing Game	**$29.95**
Arcade action for CP/M! Evade goblins and collect treasure.	
Crossword Game	**$39.95**
Teach spelling and build vocabulary. Fun and challenging.	
Mailing Label Printer	**$29.95**
Select and print labels in many formats.	
Window System	**$29.95**
Application control of independent virtual screens.	

All products require 56k CP/M 2.2 and are available on 8" IBM and 5" Northstar formats, other 5" formats add $5 handling charge. California residents include sales tax.

Poor Person Software
3721 Starr King Circle
Palo Alto, CA 94306
tel 415-493-3735

CP/M is a registered trademark of Digital Research

CIRCLE 51 ON READER SERVICE CARD

is its lack of strong typing for variables and constants. BetterBASIC has a *PRECISION* statement that allows the user to specify the default precision for arithmetic operations, and I expected that command to allow me to get the correct answer. It didn't. I had to supply the decimal point and the zero to get a real result.

This is unfortunate, but perhaps this is as good a place as any for a digression on programming using BetterBASIC.

The BetterBASIC manual abounds with program examples. In fact, the manual is excellent. The index could be a little more complete, but the 400+ pages of documentation supplied are clear, concise, and well-organized. I've read and written more pages of documentation than I care to think about. The BetterBASIC manual is among the very best language manuals I've seen.

Every single function, procedure, or command is documented and easily located in the manual. If it requires a particular module or modules for execution, the module or modules are listed. The purpose of each function, command, or statement is listed, so is its FORMAT (syntax), a list of possible arguments, at least one example of use, and some valuable comments about any differences from the IBM BASIC equivalent.

As an example, I just flipped the manual open to page 131. On that page was a description of the *DATA* procedure. Now, *DATA* is very simple, but BetterBASIC has implemented the *DATA* procedure so that not only integer, real, or string constants can be data. Integer, string, or real variables can also be data items! This allows run-time computed values to be included in the list of constants in a *DATA* statement. The BetterBASIC manual modestly notes that this capability can "greatly simplify many programming situations."

This is one example of many I could make. The point, here, is that the documentation is complete.

The difficulty in using BetterBASIC lies not in the manual and not in the documentation of each of the functions and statements. Rather, it lies in which functions and statements are included in the language. Here we get into difficulties in basic philosophy and orientation. What should BASIC be?

Should BASIC be a rationalized Pascal? A simplified C? Or should BASIC be just a better BASIC?

Again, let's take an example program from page 55 of the BetterBASIC manual. The procedure given is called *BELL*.

```
INTEGER N
PROCEDURE:Bell
INT ARG Count/OPT=N
10 DO Count TIMES
20    PRINT CHR$(7);
30    DO 1000:REPEAT 'Time delay
40 REPEAT
```

This example is for a procedure that can be called from another BASIC program. To invoke it, just include a line like this in the calling program:

```
100 Bell(X)
```

where *X* is the number of times you want the bell to be sounded. This is too verbose and too much like Pascal for my taste.

BetterBASIC uses the concept of a workspace rather than a single program. All the programs and procedures you need can be kept in the workspace at one time and be invoked by using their names. This approach is much like APL's workspace.

Let's go on to the next series of tests: real and string-handling benchmarks.

Test 4 was a test of the BASIC string operation *MID$*. Since BetterBASIC uses a fixed location for each string and the size of the string must be pre-allocated if longer than 16 bytes, I expected BetterBASIC to be quite a bit faster than IBM

For your IBM/PC

mbp COBOL: 4 times faster, and now with SORT & CHAIN.

$750.

mbp COBOL can be summed up in one word: fast.

Because it generates native machine language object code, the mbp COBOL Compiler executes IBM/PC* programs *at least* 4 times faster (see chart).

GIBSON MIX Benchmark Results
Calculated S-Profile
(Representative COBOL statement mix)

Execution time ratio

mbp COBOL	Level II** COBOL	R-M*** COBOL	Microsoft**** COBOL
1.00	4.08	5.98	6.18

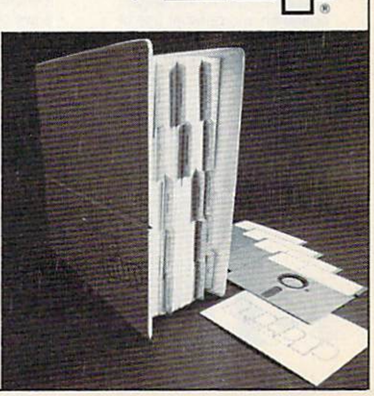
128K system with hard disk required. *IBM/PC is an IBM TM; **Level II is a Micro Focus TM; ***A Ryan-McFarland TM; ****A Microsoft TM.

Fast also describes our **new SORT**, which can sort four-thousand 128-byte records in less than 30 seconds. A callable subroutine or stand-alone, 9 SORT control fields can be specified. And our **new CHAIN** is both fast and secure, conveniently transferring control from one program to another, passing 255 parameters. Plus, **new extensions** to ACCEPT & DISPLAY verbs give better, faster interactive programming.

The complete COBOL. An Interactive Symbolic Debug Package included standard; Multi-Keyed ISAM Structure; listing options allow source & object code, map & cross-reference checking; GSA Certification to ANSI '74 Level II; mbp has it all.

It's no surprise companies like Bechtel, Chase, Citicorp, Connecticut Mutual, and Sikorsky choose mbp COBOL; make it your choice, too. mbp is available at Vanpak Software Centers, or direct. For complete information, write **mbp Software & Systems Technology, Inc.**, 7700 Edgewater Drive, Suite 360, Oakland, CA 94621, or phone 415/632-1555 —today.

mbp®



CIRCLE 39 ON READER SERVICE CARD

BASIC. I was not surprised. Here's test 4:

```
SOURCE
PROCS=0
STRING: TEST$[255]
INTEGER: I%,J%
   10 PRINT TIME$
   20 TEST$=STRING$(255,65)
   30 FOR I%=1 TO 100
   40    FOR J%=1 TO 255
   50       MID$(TEST$,J%,1)
              =CHR$(66)
   60    NEXT
   70 NEXT
   80 PRINT TIME$
ENDFILE
```

BetterBASIC ran this test in 31.6 sec; IBM BASIC took 109.0.

The next test pitted the two BASICs in real arithmetic. For compatibility, BetterBASIC's precision was set at 6 digits. The program:

```
SOURCE
PROCS=0
INTEGER: I%
REAL: J
      10 PRINT TIME$
      20 FOR I%=1 TO 100
      30    J=1
      40    J=J*1.5/6.3+2.411-7.38
      50    J=SIN(J)+COS(J)+TAN(J)
              +LOG(ABS(J))
      60 NEXT
      70 PRINT TIME$
ENDFILE
```

BetterBASIC ran this test in 12.2 sec compared with IBM BASIC's 8.4 sec. That's the first test where IBM BASIC was faster than BetterBASIC. However, BetterBASIC was able to run this test in 31 sec with precision set at 16; the IBM BASIC interpreter can't return results from transcendental functions like *SIN* accurate to more than 6 digits. BetterBASIC's precision can be set as high as 24 (BCD) digits.

Further notes about BetterBASIC: the BetterBASIC interpreter catches errors as they are entered. If you make a syntax error while entering a line, the error is flagged for you as soon as you depress the return key. This is very nice and a feature I wish IBM BASIC had.

One the other hand, you can't break a running BetterBASIC program. The CTRL-BREAK pair is ignored, as is CTRL-C. That's not so nice and is much like the operation of the IBM BASIC compiler.

For comparison purposes, I compiled the test programs with the IBM BASIC compiler, release 1.00 (Table 1).

Over the five benchmarks, BetterBASIC seems to be nearly four times as fast as IBM BASIC, but the IBM BASIC compiler is over three times (on some tests as much as 14) as fast as BetterBASIC. If speed is your prime concern, the IBM BASIC compiler should fill the bill better than BetterBASIC. Of course, BetterBASIC has advantages over the IBM BASIC interpreter and compiler other than those a sheer speed comparison would imply:

- ■ The syntax checking on entry is a nice feature, especially for beginning BASIC programmers
- ■ An excellent manual
- ■ Being able to use all available memory
- ■ The modularity of BASIC, its procedural orientation, and the modularity of the code you can write
- ■ A speed advantage over the IBM BASIC interpreter
- ■ The excellent record I/O facility
- ■ BCD arithmetic and high accuracy (up to 24 digits)
- ■ Added control and loop structures
- ■ Recursion is explicitly supported.

Disadvantages of BetterBASIC can also be summarized:

- ■ As an extra-cost optional BASIC, the IBM BASIC compiler has higher compatibility with standard IBM BASIC and greater speed than BetterBASIC
- ■ IBM BASIC loads itself and loads programs faster than BetterBASIC
- ■ BetterBASIC wants stronger typing of variables
- ■ The nonbreakable programs are unfriendly to beginners
- ■ The default string length of 16 bytes is too small
- ■ The verbose and Pascal-derived nature of the newer commands
- ■ The greater memory requirements

(Continued on next page)

CP/M-80 C Programmers . . .

Save time

. . . with the BDS C Compiler. Compile, link and execute *faster* than you ever thought possible!

If you're a C language programmer whose patience is wearing thin, who wants to spend your valuable time *programming* instead of twiddling your thumbs waiting for slow compilers, who just wants to work *fast*, then it's time you programmed with the BDS C Compiler.

BDS C is designed for CP/M-80 and provides users with quick, clean software development with emphasis on systems programming.

BDS C features include:

- Ultra-fast compilation, linkage and execution that produce directly executable 8080/Z80 CP/M command files.
- A comprehensive debugger that traces program execution and interactively displays both local and external variables by name and proper type.
- Dynamic overlays that allow for run-time segmentation of programs too large to fit into memory.
- A 120-function library written in both C and assembly language with full source code.

Plus . . .

- A thorough, easy-to-read, 181-page user's manual complete with tutorials, hints, error messages and an easy-to-use index — it's the perfect manual for the beginner *and* the seasoned professional.
- An attractive selection of sample programs, including MODEM-compatible telecommunications, CP/M system utilities, games and more.
- A nationwide BDS C User's Group ($10 membership fee — application included with package) that offers a newsletter, BDS C updates and access to public domain C utilities.

Reviewers everywhere have praised BDS C for its elegant operation and optimal use of CP/M resources. Above all, BDS C has been hailed for it's remarkable *speed*.

BYTE Magazine placed BDS C ahead of all other 8080/Z80 C compilers tested for fastest object-code execution with all available speed-up options in use. In addition, BDS C's speed of compilation was almost *twice* as fast as its closet competitor (benchmark for this test was the Sieve of Eratosthenes).

"I recommend both the language and the implementation by BDS very highly."
Tim Pugh, Jr.
in *Infoworld*

"Performance: *Excellent.*
Documentation: *Excellent.*
East of Use: *Excellent.*"
InfoWorld
Software Report Card

". . . a superior buy . . ."
Van Court Hare
in *Lifelines/The Software Magazine*

Don't waste another minute on a slow language processor. Order your BDS C Compiler today!

Complete Package (two 8" SSDD disks, 181-page manual): **$150**
Free shipping on prepaid orders inside USA.
VISA/MC, COD's, rush orders accepted.
Call for information on other disk formats.

BDS C is designed for use with CP/M-80 operating systems, version 2.2. or higher. It is not currently available for CP/M-86 or MS-DOS.

BDSoftware

BD Software, Inc.
P.O. Box 2368
Cambridge, MA 02238
(617) 576-3828

CIRCLE 5 ON READER SERVICE CARD

FORTRAN vs. FORTRAN 8x

(Continued from page 21)

although there are those who believe that FORTRAN is and should remain an assembler, and that most language enhancements should be via one of many preprocessors.

Martin: I don't think the experienced programmer will be hampered. Experienced FORTRAN programmers at my laboratory have invented all sorts of ingenious schemes involving macros, preprocessors, and subprogram libraries to achieve many of the capabilities that the proposed standard will provide. The problem with ingenious schemes is that each project has invented its own, and this hampers portability of programmers among the projects.

There seems to be some feeling that the X3J3 is perhaps a bit biased in favor of the academic community. What is the nature of X3J3?

Matheny: Biased? We have perhaps 40 strong-minded people on X3J3. And each of us has his or her own point view. Any bias of X3J3 is toward the large-scale, scientific user.

Martin: At the present time, X3J3 has 37 members that can be categorized as follows:

Hardware vendors	19
Software vendors	3
Users	9
Academics	5
Other	1

I would like to see a better ratio of users to hardware vendors on the committee. Certainly all major vendors should be represented but it is the users who should have the larger voice.

Meissner: X3J3 is an ANSI committee, and has to follow certain rules. For example, whenever anybody writes them a letter, it has to be answered. With FORTRAN 77, 2,400 questions were received and had to be answered. Of course, you could group those questions together; there is an awful lot of duplication among the questions.

In order to get a proposal through X3J3, you have to really become part of it. You have to get close enough to find

BetterBASIC

(Continued from previous page)

- Errors are reported by number; there are few explanatory error messages and at least some of them are misleading
- Variables aren't cleared between successive runs of a program
- There's no *ERL, TRON,* and *TROFF* functions for debugging programs.

BetterBASIC is an interesting attempt at creating a new version of BASIC. It is strong in a few areas and weak in a few others but overall appears to be a language that should entice Pascal programmers to return to BASIC. It should also appeal to BASIC programmers looking for an easy way to develop modular programs, business programmers requiring modular construction and BCD arithmetic, and all programmers desiring a more natural and efficient way of handling file I/O.

BetterBASIC is faster for most operations than the IBM BASIC interpreter, and its use of all available memory and support of program modularity ought to facilitate the construction and development of larger applications.

This language is well worth a look. I strongly recommend you investigate the package if any of the listed strengths appeal to you. Despite my stated misgivings about particular parts of BetterBASIC, this is definitely a good implementation of the BASIC language.

By Bruce W. Tonkin

Now with 8087* Floating Point Support

***The* C Interpreter: *Instant-C*™**

Programming in C has never been Faster.
Learning C will never be Easier.

Instant-C™ is an optimizing interpreter for the C language that can make programming in C three or more times faster than using old-fashioned compilers and loaders. The interpreter environment makes C as easy to use and learn as Basic. Yet ***Instant-C***™ is 20 to 50 times faster than interpreted Basic. This new interactive development environment gives you:

Instant Editing. The full-screen editor is built into ***Instant-C***™ for immediate use. You don't wait for a separate editor program to start up.

Instant Error Correction. You can check syntax in the editor. Each error message is displayed on the screen with the cursor set to the trouble spot, ready for your correction. Errors are reported clearly, by the editor, and only one at a time.

Instant Execution. ***Instant-C***™ uses no assembler or loader. You can execute your program as soon as you finish editing.

Instant Testing. You can immediately execute any C statement or function, set variables, or evaluate expressions. Your results are displayed automatically.

Instant Symbolic Debugging. Watch execution by single statement stepping. Debugging features are built-in; you don't need to recompile or reload using special options.

Instant Loading. Directly generates .EXE or .CMD files at your request to create stand-alone versions of your programs.

Instant Floating Point. Uses 8087* co-processor if present.

Instant Compatibility. Follows K & R standards. Comprehensive standard library provided, with source code.

Instant Satisfaction. Get more done, faster, with better results. ***Instant-C***™ is available now, and works under PC-DOS, MS-DOS*, and CP/M-86*.

Find out how ***Instant-C***™ is changing the way that programming is done. ***Instant-C***™ is $500. Call or write for more information.

Rational
Systems, Inc.

(617) 653-6194
P.O. Box 480
Natick, Mass. 01760

Trademarks: MS-DOS (Microsoft Corp.), 8087 (Intel Corp.), CP/M-86 (Digital Research, Inc.), Instant-C (Rational Systems, Inc.)

CIRCLE 56 ON READER SERVICE CARD

STRUCTURE FOR BASIC

With PROFESSIONAL PROGRAMMING ENVIRONMENT

BENDORF ASSOCIATES
6006 S. MAIN
P.O. BOX 5910
ROSWELL, NM 88201
505 347-5701
VISA/MASTERCARD

ELIMINATES LINE NUMBERS

ALLOWS MULTI-LINE CONDITIONALS

PCDOS/MSDOS

WORKS WITH BASICA INTERPRETER COMPILER

FULL ERROR LOGGING

PROGRAM LISTER

LABELED PROCEDURES

MACROS

SUB-ROUTINES

LIBRARIES

$49.95

CIRCLE 10 ON READER SERVICE CARD

New Release

CP/M ⟷ ISIS
for
PDS & MDS

ICX v.4 eXchanger now supports BOTH 8" MDS and 5-1/4" iPDS formats. Manipulation of ISIS-II files using your CP/M system was never easier.

ISE v.6 Emulator gives the CP/M-80 user access to all the ISIS-II languages and utilities.

Complete source (C and MAC asm) included with all packages

ICXMDS	$89
ICXPDS	$89
ISE	$89
ICX Toolkit (all 3)	$250

Copyrights: CP/M Digital Research, Inc.; ISIS-II and iPDS Intel Corp.

Western Wares

303-327-4898 • Box C • Norwood, CO 81423

CIRCLE 89 ON READER SERVICE CARD

A general purpose programming language for string and list processing and all forms of non-numerical computation.

SNOBOL4+ —the entire SNOBOL4 language with its superb pattern-matching facilities • Strings over 32,000 bytes in length • Integer and floating point using 8087 or supplied emulator • ASCII, binary, sequential, and random-access I/O • Assembly Language interface • Compile new code during program execution • Create SAVE files • Program and data space up to 300K bytes RAM

Have you tried SNOBOL4+?

MUSIC ANALYSIS • TEXT PROCESSING • SYMBOLIC MATHEMATICS • ARTIFICIAL INTELLIGENCE

COMPILER PROTOTYPING • CRYPTOGRAPHY • LINGUISTICS • LIST PROCESSING • GAMES •

With **ELIZA** & over 100 sample programs and functions

For all 8086/88 PC/MS-DOS or CP/M-86 systems, 128K minimum 5¼" DSDD, specify DOS/CPM format

Send check, VISA, M/C to:

$95
plus $3 s/h

Catspaw, Inc.
P.O. Box 1123 • Salida, CO 81201 • 303/539-3884

CIRCLE 9 ON READER SERVICE CARD

---

* * * EASY TO USE * * *
Macro Programs for

Spellbinder™

NOW for PC-DOS

Customized Tables -- RAM disk with .BAT file. Toggle extended character set and printer control.

Our old standby, DearJohn, mailing list management with utilities. Still $67.50 CP/M or PC-DOS

"NEW" "All You Wanted to know about Spellbinder but Couldn't Find Out". A programmer's notebook of major commands and how to use. $10.00 per copy

We have a long list of custom macros for use in the office and by the programmer. Send $1.00 for catalog.

COMPUTER RESOURCES of WAIMEA

P.O. Box 1569 Kamuela, Hawaii 96743
(808) 885-7905

CIRCLE 14 ON READER SERVICE CARD

ADVERTISE

in the April issue of

COMPUTER LANGUAGE

Special distribution at the West Coast Computer Faire

Reservation deadline:
February 6th
Call (415) 957 9353

BDS C v1.5
For *FAST*
Development

- Fastest compile to execute cycle of any CP/M 80® Compiler.
- Dynamic debugger.
- 180 page manual.

KS res. add 4% tax.
$120.00
Plus $2.50
Shipping & handling

TERMS: check, c.o.d., charge card.

(316) 431-0018

DMS
Dedicated Micro Systems, Inc.

P.O. Box 481 Chanute, KS 66720

CP/M ® trademark of Digital Research

CIRCLE 21 ON READER SERVICE CARD

C Programmers
B-Trees
For
$75.00 + 2.00 Postage
Source Code Included

The Softfocus B-Trees record indexing library will help you develop sophisticated application programs. With Softfocus B-Trees you get:

- high speed file handling for up to 16.7 million records per file
- customizable BDS or K & R standard C source code
- no royalties on applications programs
- support random and sequential file access
- includes example programs

Softfocus
1277 Pallatine Dr., Oakville, Ont. Canada L6H 1Z1 (416) 844-2610

CHARGEX VISA

CIRCLE 79 ON READER SERVICE CARD

CompuPro®
Users Group
Over 700 Members and Growing!

Join now to receive:
— Technical Newsletter
— Bulletin Board & RCPM
— Software Reviews
— New Product Announcements
— Hardware Help
— And More!

All for just $24/year!
Send check to:
C-PRO Users Group
P.O. Box 2146
Woodbridge, VA 22193

CIRCLE 34 ON READER SERVICE CARD

↓
→ *All* <u>LOMAS</u> <u>DATA</u> <u>PRODUCTS</u>' ←
16-Bit, S-100 Bus: Systems & Boards

*** * * *<u>FIREBALL DISCOUNTS</u>* * * ***
25% off List Price (UPS extra) for Cashier's Check or Money Order
+ + +
You save *MONEY* = We save *TIME*
+ + +
See → LOMAS ← ad in BYTE or request complete specifications

\ *<u>LIGHTNING SPEED</u>* /
80286-CPU & 80287-NDP: *$1458.75*

! ! *<u>THUNDERING PERFORMANCE</u>* ! !
80186 & I/O & 256K-DRAM & 5.25"/8" disk controller & CCP/M-86: *$1196.25*

Configured for *SanTec-S700* printer:
TeleVideo-GA970C* VDT: *$1121.75
Spellbinder-5.30* W/P: *$371.25

HIERATIC COMPUTER SYSTEMS
BOX 133; MEDFORD, MA 02155
(617) 683-6540

CIRCLE 87 ON READER SERVICE CARD

out what has gone before. There could be five years of minutes, including abbreviated discussions on all these topics. A lot, of course, goes on in hallways outside regular meetings. The question is why was it done this way or that way? What were the reasons or were there any reasons? It's very hard to keep track of what's happening when you're watching from the outside.

The best way to get a change in something you hear about is to go through someone on the committee who knows what has been going on. Or go to the meetings and take the opportunity to buttonhole people in the corridor and ask, "Hey, why has this feature been done that way? Is there support to do it that way? Is it a dead issue?" The committee is political as well as technical.

Martin: The committee seems to change by two members every other meeting, and that can mean two more people who have to get up to speed in what has gone before.

Before joining the committee, I would never have thought that language design could be successfully conducted by a committee. I have since changed my mind. The members and consultants have widely diverse backgrounds. This ensures that any accepted change of direction or any new feature that gets enough support to be adopted will meet the needs of many application areas and many user communities, not just one.

Any feature that is accepted gets a trial by fire. All the arguments, pro and con, are aired. It is a very democratic process. Unfortunately, it takes a lot of time. Design by committee is certainly not the most efficient method but it is thorough and painstaking. In the end I think it produces a superior product. The resulting language is not exactly what any one person on the committee would have wanted but it is acceptable to the majority or it would not be proposed as a standard.

Adams: This year, X3J3 is holding forums in most sections of the country to explain the design of the language and the new features. During the sessions, these issues are discussed. Before we release the standard, we hope to have spoken with and heard from the FORTRAN user community. Depending on the response, the list of changes will undoubtedly be modified.

FORTRAN is undergoing a major revision of a kind not attempted before. Whether it succeeds or fails, the results should prove quite interesting. Even if you are not a FORTRAN user, the results of this effort can be invaluable, if not instructive. Consider that every language in use, both formal and natural, requires some sort of standardization. And consider that there are standards committees for many programming languages extant.

If you are a FORTRAN user, you will undoubtedly be affected by the proposed changes. Your input to the committee can influence the direction that FORTRAN will take in the future, whether toward a system level or problem level orientation.

If you wish to keep current with the activities of the X3J3 committee or if you wish to make your comments known, you can write to: Dr. Loren Meissner, editor, ACM SIGPLAN FORTRAN Newsletter, Univ. of San Francisco, Ignatius Heights, San Francisco, Calif., 94117-1080. Letters may be published in the newsletter and will be passed on to X3J3.

If you wish to become a member of X3J3, you can contact Jeanne Martin, L-300, Lawrence Livermore National Laboratories, Livermore, Calif. 94550.

If you wish to make comments on the proposal, contact Andrew Johnson, MS 10C17-3, Prime Computer Inc., 500 Old Connecticut Path, Framingham, Mass., 01701.

GOOD NEWS!

C for the 6809 WAS NEVER BETTER!

INTROL-C/6809,Version1.5

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

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

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

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

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

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

For further information, please call or write.

CIRCLE 32 ON READER SERVICE CARD

ADVERTISER INDEX

	PAGE NO.	CIRCLE NO.
ACM	33	7
Alcor Systems	22	1
Alpha Computer Service	68	2
Atron	2	4
Awareco	52	3
BD Software	76	5
Bendorf & Associates	78	10
Blaise Computing, Inc.	11	11
Borland International	6,7	6
Business Utility Software	16	17
C - Line, Inc.	4	26
C Systems	54	16
C Ware	58	18
C-Pro Newsletter	78	34
Carousel Micro Tools	48	8
Catspaw, Inc.	78	9
Code Works (The)	15	42
CompuPro	Cover IV	12
Computer Resources of Waimea	78	14
Computing!	24	37
Compuview Products	65	40
DWB Associates	70	20
Datalight	14	19
Dedicated Micro Systems, Inc.	78	21
Earth Computer	80	24
Echelon, Inc.	17	43
Ecosoft	70	22
Essential Software, Inc.	22	28
FairCom	67	29
Forth Inc.	45	30
Greenleaf Software	64	44
HSC, Inc.	60	31
Harvard Softworks	46	47
Hieratic Computer Systems	78	87
Introl Corp.	79	32
Laboratory Microsystems, Inc.	58	35
Lahey Computer Systems, Inc.	21	53
Lattice Inc.	58	36
Lomas Data Products	34	54
mbp Software & Systems Technology	75	39
MEF Environmental	74	90
MEF Environmental	20	55
Megamax, Inc.	62	13
Mendocino Software Co., Inc.	22	48
Micro Methods	66	52
MicroWay	48	57
Miller Microcomputer Service	47	61
Mindbank, Inc.	70	63
Next Generation Systems	32	45
Northwest Computer Algori thms	68	46
Opt-Tech Data Processing	66	67
Opt-Tech Data Processing	74	66
Parsec	74	49
PlumHall	61	50
Poor Person Software	74	51
ProCode	54	60
Programmer's Connection	72	62
Programmer's Shop	71	69
Programmer's Shop	54	71
QCAD	68	23
RR Software	Cover II	58
Rqtional Systems, Inc.	77	56
Realia, Inc.	10	76
Ream, Edward	68	-
SLR Systems	18	59
Simpliway Products, Co.	16	70
Soft Advances	67	77
Soft Craft, Inc.	73	78
Softfocus	78	79
Software Horizons	53	25
Solution Systems	25	80
Solution Systems	61	81
Solution Systems	61	82
Spruce Technology Corp.	31	33
Stride Micro	Cover III	72
Summit Software	12	83
System Engineering Tools	15	64
Systems Guild	18	84
Thunder Software	64	65
UniPress	62	38
Western Wares	78	89
Whitesmiths, Ltd.	1	85
Wizard Systems	14	86
Wordtech Systems	8	41
Workman & Associates	62	68

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

THE WORLD'S FASTEST

S-100 Z-80 SLAVE PROCESSOR

TurboSlave I

- 8 Mhz Z-80H
- Data transfers to 1 mbyte/second
- S-100 IEEE-696 compatible
- 4k Monitor rom
- Low parts count
- No paddle boards
- 128k Ram with parity
- 2 RS-232 Ports. 50-38.k baud
- F.I.F.O. communications
- On board diagnostics
- Low power consumption
- TurboDOS compatible
- **GUARANTEED COMPATIBLE WITH ALL S-100 SYSTEMS RUNNING TURBODOS**

INTRODUCTORY PRICE $495

Includes TurboDOS drivers (a $100 value) and TurboSlave I with 128k ram.

EARTH COMPUTERS

P.O. Box 8067, Fountain Valley, CA 92728
TELEX: 910 997 6120 EARTH FV

FOR MORE INFORMATION AND QUANTITY DISCOUNTS
CALL: (714) 964-5784

Registered trademarks: Z-80H, Zilog Inc.; TurboDOS Software 2000, Inc.

***** IBM PC VERSION COMING SOON *****

CIRCLE 24 ON READER SERVICE CARD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 27346 PHILADELPHIA, PA USA

POSTAGE WILL BE PAID BY

COMPUTER LANGUAGE

P.O. BOX 11747
PHILADELPHIA, PA 19101

COMPUTER LANGUAGE

Subscribe to **COMPUTER LANGUAGE** today for only $24.00 — over 33% savings off the single copy price.

☐ Yes, start my Subscription to COMPUTER LANGUAGE today. The cost is only$24.00 for 1 year (12 issues).

☐ I want to increase my savings even more — send me 2 years (24 issues) of COMPUTER LANGUAGE for only $39.00.

☐ Payment enclosed ☐ Bill me

Name ______________________________

Company ______________________________

Address ______________________________

City, State, Zip ______________________________

Please allow 6-8 weeks for delivery of first issue. Foreign orders must be prepaid in U.S. funds. Canada orders $30.00 per year. Outside the U.S., $36.00/year for surface mail or $54.00/year for airmail.

BIJ5

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 27346 PHILADELPHIA, PA USA

POSTAGE WILL BE PAID BY

COMPUTER LANGUAGE

P.O. BOX 11747
PHILADELPHIA, PA 19101

COMPUTER LANGUAGE

Subscribe to **COMPUTER LANGUAGE** today for only $24.00 — over 33% savings off the single copy price.

☐ Yes, start my Subscription to COMPUTER LANGUAGE today. The cost is only$24.00 for 1 year (12 issues).

☐ I want to increase my savings even more — send me 2 years (24 issues) of COMPUTER LANGUAGE for only $39.00.

☐ Payment enclosed ☐ Bill me

Name ______________________________

Company ______________________________

Address ______________________________

City, State, Zip ______________________________

Please allow 6-8 weeks for delivery of first issue. Foreign orders must be prepaid in U.S. funds. Canada orders $30.00 per year. Outside the U.S., $36.00/year for surface mail or $54.00/year for airmail.

BIJ5

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 22481 SAN FRANCISCO, CA USA

POSTAGE WILL BE PAID BY

COMPUTER LANGUAGE

2443 FILLMORE STREET • SUITE 346
SAN FRANCISCO, CA 94115

READER SERVICE CARD

Name ______________________

Company ______________________

Address ______________________

City, State, Zip ______________________

Country ________ Telephone number ______________________

January issue. Not good if mailed after May 31, 1985.

Circle numbers for which you desire information.

1	11	21	31	41	51	61	71	81	91
2	12	22	32	42	52	62	72	82	92
3	13	23	33	43	53	63	73	83	93
4	14	24	34	44	54	64	74	84	94
5	15	25	35	45	55	65	75	85	95
6	16	26	36	46	56	66	76	86	96
7	17	27	37	47	57	67	77	87	97
8	18	28	38	48	58	68	78	88	98
9	19	29	39	49	59	69	79	89	99
10	20	30	40	50	60	70	80	90	100

Please complete these short questions:

1. I obtained this issue through:
   - ☐ Subscription
   - ☐ Computer Store
   - ☐ Retail outlet
   - ☐ Passed on by associate
   - ☐ Other ______________
2. Job Title ______________________
3. The 5 languages that I am most interested in reading about (list in order of importance).

____

____

Comments ______________________

____

____

Attn: Reader Service Dept. 1/5

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 22481 SAN FRANCISCO, CA USA

POSTAGE WILL BE PAID BY

COMPUTER LANGUAGE

2443 FILLMORE STREET • SUITE 346
SAN FRANCISCO, CA 94115

READER SERVICE CARD

Name ______________________

Company ______________________

Address ______________________

City, State, Zip ______________________

Country ________ Telephone number ______________________

January issue. Not good if mailed after May 31, 1985.

Circle numbers for which you desire information.

1 11 21 31 41 51 61 71 81 91
2 12 22 32 42 52 62 72 82 92
3 13 23 33 43 53 63 73 83 93
4 14 24 34 44 54 64 74 84 94
5 15 25 35 45 55 65 75 85 95
6 16 26 36 46 56 66 76 86 96
7 17 27 37 47 57 67 77 87 97
8 18 28 38 48 58 68 78 88 98
9 19 29 39 49 59 69 79 89 99
10 20 30 40 50 60 70 80 90 100

Please complete these short questions:

1. I obtained this issue through:
☐ Subscription ☐ Passed on by associate
☐ Computer Store ☐ Other ______________
☐ Retail outlet

2. Job Title ______________________

3. The 5 languages that I am most interested in reading about (list in order of importance).

____

____

Comments ______________________

____

____

Attn: Reader Service Dept. 1/5

"Despite the recent press notices, multiuser microcomputers aren't anything new!"

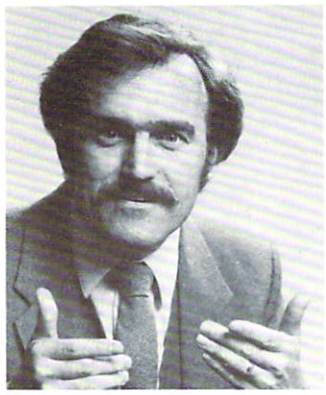
This is the first in a series of discussions with Rod Coleman, President of Stride Micro (formerly Sage Computer) on the 68000 multiuser market and its current environment.

Q: Why do you say that?

RC: "The technology to build a high performance multiuser system has been around for five years. And while some of the leaders in this industry have been pretending that micro multiuser didn't exist, we've been shipping complete systems for nearly three years. The benefits of multiuser are undeniable; it is more cost effective, and offers greater flexibility and utility. But until just recently, the marketing pressure to be compatible instead of being better, has blinded the industry."

Q: What do you mean?

RC: "Well, for example, the Motorola 68000 processor introduced 16/32-bit technology to the personal computer world a long time ago. It was fully capable of meeting high performance and multiuser design requirements in 1980. Instead of this trend taking off, most energy was spent promoting 8088/8086 products that were clearly inferior from a technical point of view. This phenomenon leads me to believe that they will soon rewrite the old proverb: 'Build a better mousetrap and the world will beat a path to your door,' but only if they can find the way through the marketing fog."



"A surprising feature is compatibility. Everybody talks about it, but nobody does anything about it."

Q: Are things changing now?

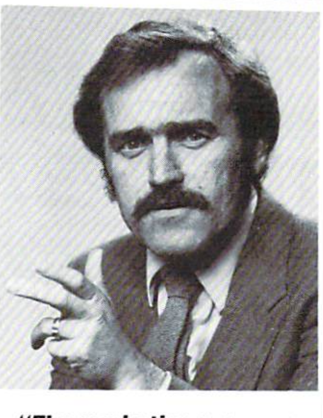
RC: "Yes and no. With the business world starting to take more and more interest in microcomputer solutions, the advantages of a solid multiuser system couldn't be kept hidden forever; companies like ours and a few others were beginning to make a dent. Instead of taking a fresh approach, some of the newest multiuser offerings will probably only give the technology an undeserved black eye! Multiuser is far more than the ability to plug in more terminals. It involves things like machine compatibility, fast processors, adequate memory, large storage capacities, backup features, networking, and operating system flexibility."

Q: Is this what makes the new Stride 400 Series different?

RC: "Exactly. That sounds self-serving, but it's true. Today a number of companies are introducing their first multiuser system. We've been building and shipping multiuser machines for almost three years. We know the pitfalls, we've fallen into some of them. But we have learned from our mistakes."

Q: Give me some examples.

RC: A hard disk is almost mandatory for any large multiuser installation. Yet, backing up a hard disk can be a nightmare if you only have floppies to work with. That's why we've added a tape backup option to all the larger Stride 400 Series machines. It's irresponsible for a manufacturer to market a multiuser system without such backup. Another good lesson was bus design. We started with one of our own designs, but learned that it's important not only to find a bus that is powerful, but also one that has good support and a strong future to serve tomorrow's needs. We think the VMEbus is the only design that meets both criteria and thus have made it a standard feature of every Stride 400 Series machine."



"The marketing pressure to be compatible instead of being better, has blinded the industry."

Q: What are some of the other unique features of the 400 Series?

RC: "A surprising feature is compatibility. Everybody talks about it, but nobody does anything about it. Our systems are completely compatible with each other from the 420 model starting at $2900, through the 440, on to the powerful 460 which tops out near $60,000. Each system can talk to the others via the standard built-in local area network. Go ahead and compare this with others in the industry. You'll find their little machines don't talk to their big ones, or that the networking and multiuser are incompatible, or that they have different processors or operating systems, and so on."

Q: When you were still known as Sage Computer, you had a reputation for performance, is that still the case with the new Stride 400 Series?

RC: "Certainly, that's our calling card: 'Performance By Design.' Our new systems are actually faster; our standard processor is a 10 MHz 68000 running with no wait states. That gives us a 25% increase over the Sage models. And, we have a 12 MHz processor as an option. Let me add that speed isn't the only way to judge performance. I think it is also measured in our flexibility. We support a dozen different operating systems, not just one. And our systems service a wide variety of applications from the garage software developer to the corporate consumer running high volume business applications."

Q: Isn't that the same thing all manufacturers say in their ads?

RC: "Sure it is. But to use another over used-term, 'shop around'. We like to think of our systems as 'full service 68000 supermicrocomputers.' Take a look at everyone else's literature and then compare. When you examine cost, performance, flexibility, and utility, we don't think there's anyone else in the race. Maybe that's why we've shipped and installed more multiuser 68000 systems than anyone else."

STRIDE MICRO

Formerly Sage Computer

For more information on Stride or the location of the nearest Stride Dealer call or write us today. We'll also send you a free copy of our 32 page product catalog.

Corporate Offices:
4905 Energy Way
Reno, NV 89502
(702) 322-6868

Regional Offices:
Boston: (617) 229-6868
Dallas: (214) 392-7070

CIRCLE 72 ON READER SERVICE CARD

CompuPro has now dramatically increased your microcomputing power and speed.

With our System 816/F™ supermicro with **CPU 286/287**™ board. The computer that gives you results. Fast.

Built to provide sophisticated computer users with the fastest 16-bit system available, the System 816/F is a multi-user computer so powerful it virtually has no supermicro peer.

The reason for such a strong statement? We configure the system around the **80286**—among the most powerful 16-bit processors available anywhere and one that's built for speed. The 286/287 board lets you run anything from the 8086/8088 family and includes the 80287 math processor and as much as 16 Kb of EPROM on-board.

But this board is only part of the story. CompuPro has included a long list of features that enhance this exclusive system even more. Like 1.5 Mb of our MDRIVE®/H—a solid-state disk with the capacity to dramatically increase the speed of the 286 processor even more . . . 512 Kb of 16-bit main memory expandable to 16 Mb . . . 1.2 Mb floppy disk and up to 80 Mb of hard disk storage . . . 12 serial ports . . . and much more.

And even though our System 816/F has set some industry standards, we still designed it to conform to the IEEE 696/S-100 bus standard. And virtually no one else can say that.

The time you save with CompuPro will save you money, too. Our System 816/F speeds up software development. So the quality and capacity of your programs is enhanced, and the value, maximized.

The CompuPro System 816/F. It's the essential system for knowledgeable users who want all the power and speed they can get. And best of all, we've shipped hundreds of them already.

CIRCLE 12 ON READER SERVICE CARD

The Essential Computer™

CompuPro®

A *GODBOUT* COMPANY

3506 Breakwater Court, Hayward, CA 94545

(415) 786-0909