

LIFELINES™

The Software Magazine™

\$3.00

July 1982

Volume III, No. 2 (ISSN 0279-2575, USPS 597-830)

On MicroPlan

Ada: Who Is She?

A Review Of VEDIT

**Microcommunications
Software — The Basics**

**More of Ward Christensen's
8080 Assembler Tutorial**

SB-80, A Comparative Analysis

A New Z80 Programming Tutorial

TIMTM III

The Non-Programming Approach to Data Base Management

Data Base Management

Data management packages were created to save time and money in the development of software solutions to information problems. Many have been designed to accomplish just that, although most have only the programmer in mind. Sure they would save time in the long run, but what of the initial investment in time and effort required to learn the new language? What about the non-programmers in the world who would like an easy yet powerful applications generator? The solution is one of the most highly acclaimed software packages of our time, T.I.M. III.

What is T.I.M.?

T.I.M. is **Total Information Management**. Programmers love it due to its original solutions to classic data management problems. Non-programmers adore it since they can use it to achieve the same results as with other more complicated programming-like packages.

What Makes T.I.M. So Simple to Use?

We at Innovative Software, Inc. designed T.I.M. from day one with the end user in mind. Maybe he is a programmer who doesn't have time to learn a new language. Or perhaps a neophyte who fears coding pads and lines numbered by tens. We felt that a data management package should be able to be used by anyone from a systems analyst to a secretary. That's why T.I.M. takes a full *menu-driven* approach, uses multiple *HELP* screens, and has a manual that sets a new standard in documentation.

The Manual

Many people believe that the manual is just as important as the software itself, a view that we at Innovative Software, Inc. tend to share. The manual for T.I.M. is divided into two sections, the Reference section and the Primer. The Reference section describes all of T.I.M.'s commands and subcommands. This is done in English, not in technical terms or in our own language. Even if you have

never seen a computer before in your life, you'll be able to read and understand our manual immediately. The second section is a primer which goes through several examples for you, again in plain English. These true-to-life examples take the beginner by the hand, and instructs him what to do and when. You will be able to see for yourself that T.I.M.'s only limitation is the imagination of the user.

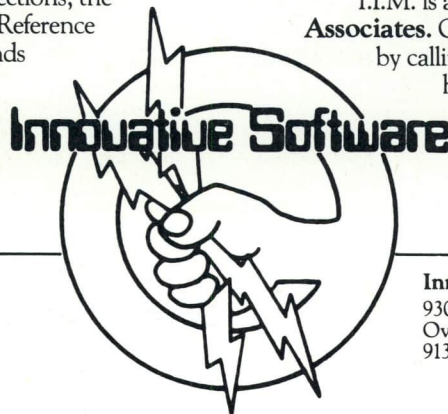
Features of T.I.M.

T.I.M. has all of the features one has come to expect from a data management package, as well as many new ones. For example, a *word processing* interface that allows you to merge information from a T.I.M. file with letters or other documents created by a word processor. Now you can automatically send personalized letters to hundreds or thousands—quickly and easily. T.I.M.'s *Select* command enables you to pull specific information from a file. For example. "All customers who live in a certain ZIP code, whose last name begins with the letter A to L, whose balance due is less than \$50.00." A sophisticated *report generator* and even a *list generator* are also included.

How powerful is T.I.M.? With a maximum record size of 2400 characters and the ability to keep up to forty fields sorted properly at all times, T.I.M. is powerful enough to handle just about any application. T.I.M. can handle over 32,000 records per file, and two files can be linked together for reports if your application requires a many-to-one relationship. T.I.M. also includes all of the same editing commands as your word processor, thus making data entry and editing a snap. You can also pull selected records from one file to place them into another. Files may be restructured to add or subtract fields and/or change field lengths or types. T.I.M. even has its own utility for backing up hard disks onto floppies.

Where to Find T.I.M.

T.I.M. is available from **Lifeboat Associates**. Or you may purchase from us direct by calling 913/383-1089. Either way you will have the finest data management program available.

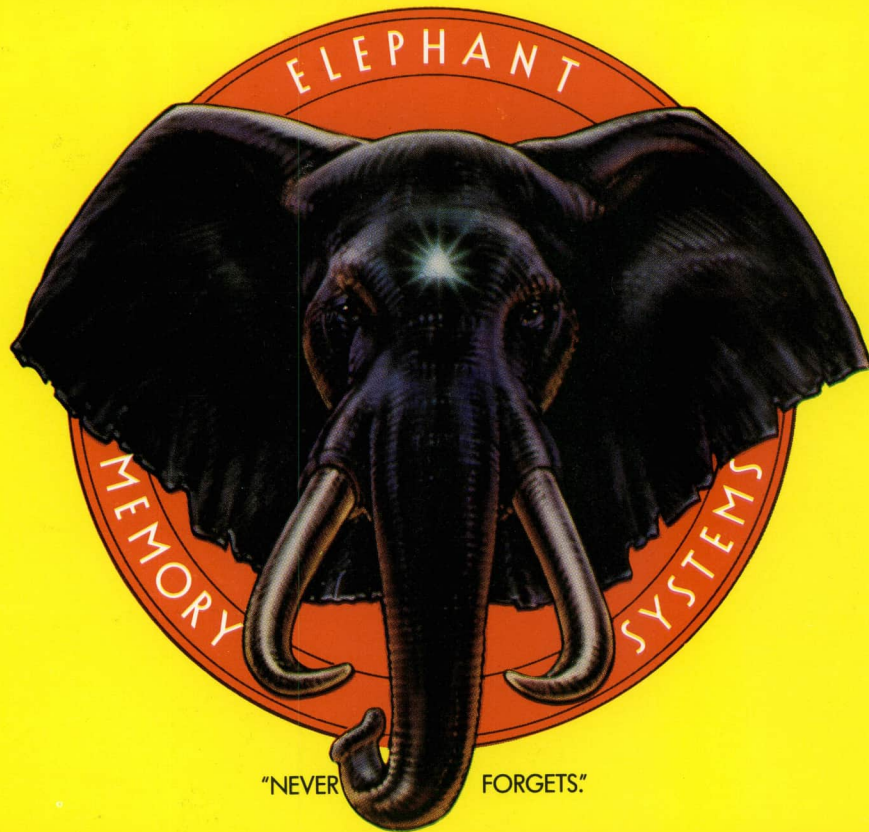


Available for CP/M,* and
IBM PC DOS.**
CP/M version—\$695. IBM PC version—\$495.

Innovative Software, Inc.
9300 W. 110th Street, Suite 380
Overland Park, Kansas 66210 USA
913/383-1089

TIM is a Trademark of Innovative Software, Inc.
*CP/M and MP/M are Trademarks of Digital Research
**Trademarks of IBM

REMEMBER:



MORE THAN JUST ANOTHER PRETTY FACE.

Says who? Says ANSI.

Specifically, subcommittee X3B8 of the American National Standards Institute (ANSI) says so. The fact is all Elephant™ floppies meet or exceed the specs required to meet or exceed all their standards.

But just who is "subcommittee X3B8" to issue such pronouncements?

They're a group of people representing a large, well-balanced cross section of disciplines—from academia, government agencies, and the computer industry. People from places like IBM, Hewlett-Packard, 3M, Lawrence Livermore Labs, The U.S. Department of Defense, Honeywell and The Association of Computer Programmers and Analysts. In short, it's a bunch of high-caliber nitpickers whose mission, it seems, in order to make better disks for consumers, is also to

make life miserable for everyone in the disk-making business.

How? By gathering together periodically (often, one suspects, under the full moon) to concoct more and more rules to increase the quality of flexible disks. Their most recent rule book runs over 20 single-spaced pages—listing, and insisting upon—hundreds upon hundreds of standards a disk must meet in order to be blessed by ANSI. (And thereby be taken seriously by people who take disks seriously.)

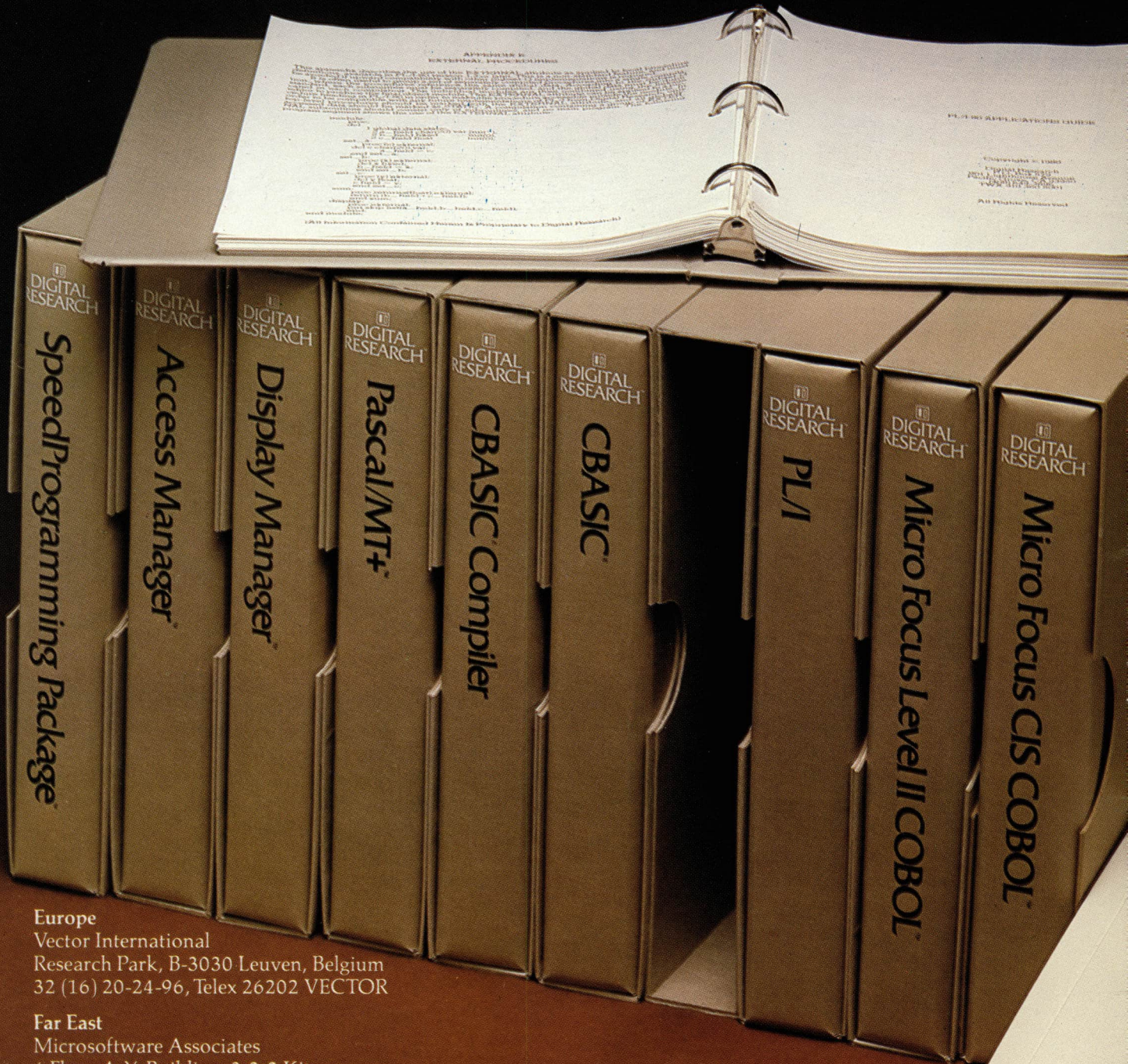
In fact, if you'd like a copy of this formidable document, for free, just let us know and we'll send you one. Because once you know what it takes to make an Elephant for ANSI . . .

We think you'll want us to make some Elephants for you.

ELEPHANT™ HEAVY DUTY DISKS.

Distributed Exclusively by Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021
Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.

PRODUCTIVITY AT YOUR FINGERTIPS.



Europe

Vector International
Research Park, B-3030 Leuven, Belgium
32 (16) 20-24-96, Telex 26202 VECTOR

Far East

Microsoft Associates
6 Floor A.Y. Building, 3-2-2 Kitaoyama
Minato-ku, Tokyo 107, Japan
03-403-2120, Telex 2426875 MSA

tools for more powerful programming.

The creators of CP/M® bring you the most powerful library of languages and utilities on the market. Developed to increase programmer productivity and to produce top quality applications, our languages allow you to work efficiently with both 8 and 16-bit microcomputers. Digital Research utilities reduce coding and designing effort for speedier programming and higher productivity. Our languages and utilities comprise the finest collection of microcomputer

programming tools available anywhere.

When you're designing microcomputer software use proven Digital Research languages and utilities. You can count on the highest standards in software and support from the company that made CP/M the most accepted microcomputer operating system. For information, call or write Digital Research (408) 649-5500 or (408) 649-3896, 160 Central Avenue, Pacific Grove, California 93950.

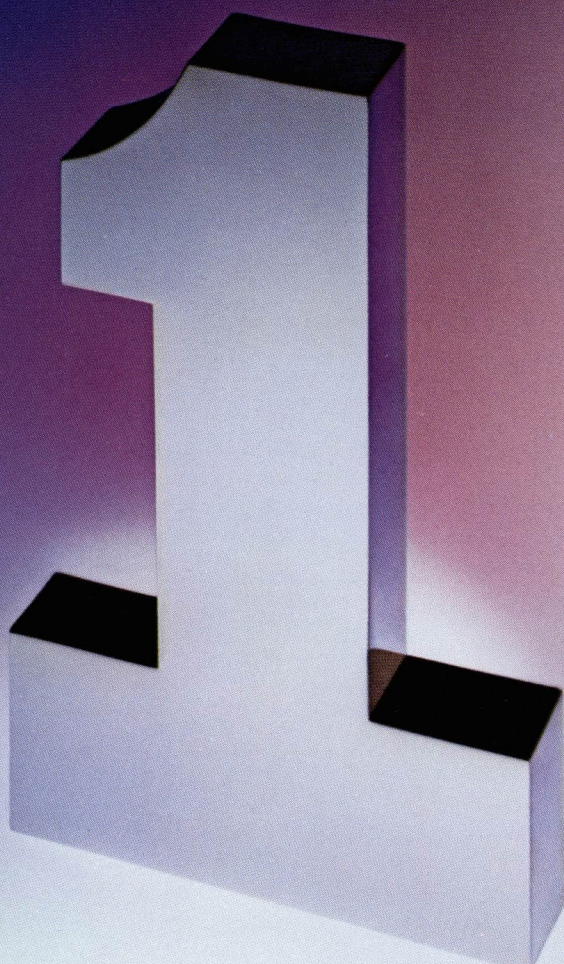


DIGITAL RESEARCH™

The creators of CP/M.™



How to get the most out of your IBM Personal Computer in one easy step.



Contact Lifeboat Associates We're the only software publisher you'll ever need for easy-to-use, reliable programs created for your **IBM Personal Computer**.

Our growing line of proven software for the IBM Personal Computer with DOS currently includes:

Emulator/86™, the CP/M-86™ emulator, that lets you use CP/M-86 software under DOS. This emulator allows you to fully integrate and mix programs, and to run CP/M-86 programs quicker with faster program loading than with CP/M-86 itself. All this without having to learn a new set of utilities and commands or end up with two incompatible sets of diskettes. Why spend hundreds of dollars on CP/M-86 when the CP/M-86 emulator is \$75.

EM80/86™, the emulator that allows execution of CP/M®-80 programs on the IBM PC under DOS with no hardware addition or change. Programs will run from 0 to 8 times slower than on a 4MHz Z80™. Run your existing software today for only \$200.

PMATE™, the text editor with single keystroke editing, expression evaluation, horizontal scrolling, powerful macro command definitions with conditional branching and other programming language constructs. The installation permits the customization of the keystroke definitions and so the user can set PMATE to mime a previously familiar editor. \$195.

Lattice C Compiler, full implementation of 'C' with library and I/O sub-routines which implement under DOS most standard I/O functions specified by Kernighan and Ritchie. UNIX™ Ver. 7 compatible; produces relocatable machine code in Intel's 8086 object module format for use with the linker supplied with DOS. \$500.

ASCOM™, the asynchronous communication facility for users who need to interact with remote time sharing services or local computers to transfer data files and programs. \$175.

T.I.M., the user-friendly, menu-driven Database Management System, creates files, inputs, edits, deletes data and produces a variety of reports. \$495.

UT86™, system utility designed to improve user friendliness of systems using DOS. Provides neatly formatted and sorted directories, interactive copy routines, or groups of files, formatted file print-outs and more. \$180.

For full information about how Lifeboat can help you get the most out of your IBM Personal Computer (and other computers using MS™-DOS), or how you can be added to our Mail List for either 8- or 16-Bit software, contact Lifeboat Associates. — The #1 resource for your IBM Personal Computer.

Prices and specifications subject to change without notice
Prices FOB New York.
Shipping and handling and C.O.D. charges extra.

Emulator/86 is a trademark of Lifeboat Associates.
MS is a trademark of Microsoft, Inc.
EM80/86, ASCOM, UT86 are trademarks of DMA, Inc.
PMATE is a trademark of Phoenix Software Asso. Ltd.
Z80 is a trademark of Zilog, Inc.
UNIX is a trademark of Bell Laboratories
CP/M is registered and CP/M-86 a trademark of Digital Research, Inc.
Copyright © 1982, by Lifeboat Associates.

Lifeboat Associates 

the standard for fully supported software
1651 Third Avenue, NY, NY 10028 (212) 860-0300
TWX: 710-581-2524 (LBSOFT NYK) Telex: 640693 (LBSOFT NYK)

Copyright © 1982, by Lifelines Publishing Corporation. No portion of this publication may be reproduced without the written permission of the publisher. The single issue price is \$3.00 for copies sent to destinations in the U.S., Canada, or Mexico. The single issue price for copies sent to all other countries is \$4.30. All checks should be made payable to Lifelines Publishing Corporation. Foreign checks must be in U.S. dollars, drawn on a U.S. bank; checks, money orders, VISA, and MasterCard are acceptable. All orders must be pre-paid. Please send all correspondence to the Publisher at the below address.

Lifelines (ISSN 0279-2575, USPS 597-830) is published monthly at a subscription price of \$24 for twelve issues, when destined for the U.S., Canada, or Mexico, \$50 when destined for any other country. Second-class postage paid at New York, New York. POSTMASTER, please send changes of address to Lifelines Publishing Corporation, 1651 Third Ave., New York, N.Y. 10028.

Lifelines - TM Lifelines Publishing Corp.
 The Software Magazine - TM Lifelines Publishing Corp.
 SB-80, SB-86 - TMs Lifeboat Associates.
 The Apple - TM Apple Computer, Inc.
 BASIC-80, MBASIC, MS, SoftCard, COBOL-80 - TMs Microsoft, Inc.
 CB80, CBASIC2, PL/I-80, SID-86, CP/M-86, Pascal MT+ - TMs, CP/M registered TM - Digital Research, Inc.
 The CP/M Users Group is not affiliated with Digital Research, Inc.
 dBASE II - TM Ashton-Tate.
 HP-125 - TM Hewlett-Packard.
 MailMerge, WordStar - TMs MicroPro International Corp.
 MicroPlan - TM Chang Laboratories.
 Pascal/Z - TM Ithaca Intersystems.
 PLAN80 - TM Business Planning Systems, Inc.
 PMATE, PLINK-II - TMs Phoenix Software Associates, Ltd.
 T.I.M. - TM Innovative Software, Inc.
 VEDIT - TM CompuView Products, Inc.
 Z80 - TM Zilog Corporation.
 Program names are generally TMs of their authors or owners.

OKIDATA

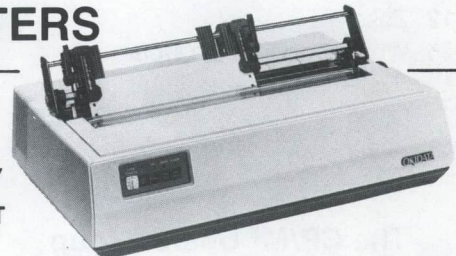
100% DUTY CYCLE PRINTERS

**SETTING NEW STANDARDS IN QUALITY
APART FROM THE REST**

MODEL	ML80	ML82A	ML83A	ML84	2350
Columns:	80	80	136	136	136
Print Speed: (cps)	80	120	120	200	350
Bidirectional/Short Line Seeking:	—	✓	✓	✓	✓
Throughput: (lpm)					
20 Char/line	86	187	173	266	500
40 Char/line	51	123	117	184	340
80 Char/line	28	73	71	114	210
136 Char/line	—	—	46	74	136
Head Life	— 200 million characters —				500 million
Graphics Option:	Block	60x66	60x66	72x72	72x72
RS 232:	Opt.	Std.	Std.	Opt.	Opt.
Tractor Feed:	Opt.	Opt.	Std.	Std.	Std.
Friction Feed:	✓	✓	✓	✓	—
Pin Feed:	✓	✓	—	—	—
Super Scripts • Sub-Scripts • Underline:	—	—	—	—	✓
Colors:	—	—	—	—	2

Immediate Delivery • Technical Assistance • Leasing • Maintenance • Interface Cables • Ribbons

DISTRIBUTED by: **GRAYDON-SHERMAN, INC.**
 (212) 289-3199 (201) 467-1401 * TWX #710-983-4375 (GRAYDON MAWD)



LIFELINES

The Software Magazine

July 1982

Volume III, No. 2

Editor-in-Chief: Edward H. Currie
Editor: Jane Mellin
Circulation/Customer Service: Patricia Matthews
Director of Communications: Bonita E. Taylor

Design/Production: K. Gartner
Typographers: Harold Black
Carole Mayer
Cover by K. Gartner

DEPARTMENTS

Opinion

- 9 The Pipeline
Wanna printer? Here's what to
look for
George Mitchell
- 12 Zoso
- 46 Why Standards On Operating
Systems Interfaces?
Jack Cowan
- 48 Letters

The CP/M® Users Group

- 27 Volume 82 Catalogue and Abstracts

Software Notes

- 23 Pascal/Z™ with Plink-II™
- 41 Getting Off Base
Michael Olfe
- 43 Pascal/MT +™ On The HP-125™
Al Bloch
- 44 MailMerge™, Do I Need It?
Robert P. VanNatta

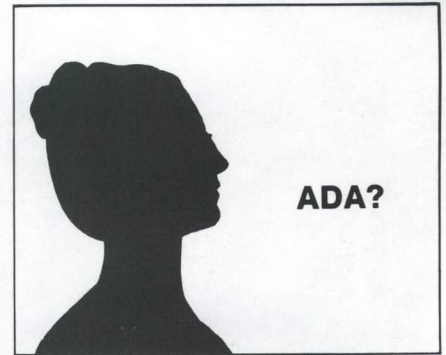
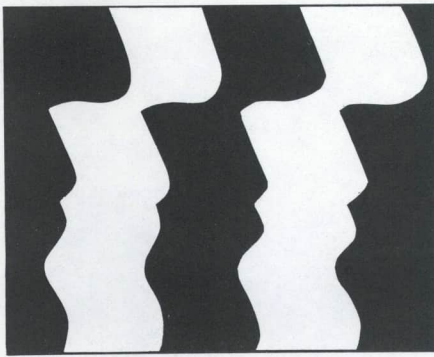
- 48 Bugs and Anomalies in dBase II™,
Version 2.3
Michael Olfe
- 52 Macros Of The Month
Michael Olfe

Product Status Reports

- 11 Books
Steve Patchen
- 49 New Products
- 51 New Versions
- 54 Version List

Miscellaneous

- 8 Index Available
- 11 Renew
- 23 Notice
- 42 Change of Address
- 47 OOPS!
- 52 Attention Dealers
- 52 Laughlines



FEATURES

15 Lifeboat Associates SB-80™, A Comparative Analysis

Michael J. Karas

A comparative method is used to introduce this new user-friendly operating system, contrasting it with CP/M-80. This article should give you a good idea of the philosophy behind SB-80.

22 Z80™ Programming Tutorial, Introduction

Kim West DeWindt

This tutorial expands on the 8080 tutorial series, covering the unique aspects of Z80 assembly language, describing its points of difference from the 8080 version. The author outlines plans for future segments, and solicits your questions about Z80 assembly language.

24 Full Screen Program Editors For CP/M-80: VEDIT™

Ward Christensen

This review assesses an ambitious full screen editor, comparing it with others on the market.

28 Ada: Who Is She?

Steve Patchen

The author discusses the role of Ada as a fourth generation language and its participation in the system development environment; this essay introduces reviews of two commercial microcomputer Adas currently available.

30 An Overview Of MicroPlan™

Raymond J. Sonoff

Part of an ongoing series, this article highlights the key features associated with Chang Laboratories' financial planning package.

32 8080 Assembler Programming Tutorial: Subroutines, Part 2

Ward Christensen

This highly acclaimed tutorial continues the examination of subroutines: MOVE, character I/O and standard equates, character input and hex input.

35 T.I.M.™ Data Base Software: First Impressions

Davis Foulger

This overview introduces a talented and versatile data base management system. A thorough evaluation will follow next month.

36 The Basics of Microcommunications Software

Davis Foulger

The structure and design principles behind microcommunications software are described step-by-step, in a continuing report on this growing field. This article will help you understand which features distinguish a primitive system from a more sophisticated one.

(continued next page)

.....	Sept81,15
.....	March81,18
.....	Aug80,15, Sept80,14
.....	July80,11
.....	Oct81,33
.....	Sept80,10
.....	April81,17
.....	March81,6
.....	Nov80,3
.....	Jan81,16
.....	Nov81,16
.....	Aug81,7
.....	Aug81,7
.....	July81,16
.....	Sept81,31
.....	Nov80,16
.....	Oct81,36
.....	May81,18
.....	Nov81,38
.....	Nov80,16
.....	June81,14
.....	Feb81,1, April81,15
.....	Sept81,24
.....	Oct81,8
.....	Nov81,5
.....	Dec81,22
.....	Aug80,3, Oct80,4
.....	Sept81,1
.....	July81,1
.....	Jan81,1
.....	June80,1
.....	Oct81,1
.....	Jan81,11, Sept81,1
.....	Nov81,1
.....	June80,5, July80,10, Aug80,16, Sept80,17, Oct80,18
.....	Nov80,17, Dec80,17, Jan81,14, Feb81,16, March81,20, April81,21
.....	June81,22, July81,22, Aug81,30, Sept81,38, Oct81,38, Nov81,39
.....	Feb81,1
.....	July80,1
.....	Aug80,1
.....	Dec80,1
.....	March81,1

Available this month, the second installment of the 1982 index to Lifelines.

Organized by subject and title, the 1982 Index is being published quarterly and is still available for only \$2.50.

Also available for an additional \$2.50 is our comprehensive index to Lifelines articles from June 80 to December 81. For only \$5, you can be up to date on Lifelines.

All orders must be prepaid by check, MasterCard, or Visa. Checks must be in U.S. Dollars, drawn on a U.S. bank. Write for your index, or call (212) 722-1700.

Wanna printer? Here's what to look for

To meet growing hardcopy needs, printer manufacturers are responding with a new crop of low-cost dot-matrix printers. Though low price tags are alluring to some, manufacturers are finding that reliability and serviceability overshadow the cost issue.

Because prices have already hit the under-\$300 level for some low-end designs and are under \$2000 for high-performance models, manufacturers are scrambling to meet serviceability and reliability requirements at these pricing levels, by reducing the overall complexity of printer mechanics and electronics.

Three Basic designs

Mechanically, horizontal motion control assemblies are becoming much simpler. According to Tritell Inc.'s v.p. Don McCombs, the horizontal motion of the printer is the most critical and the most exacting to design, especially when you're trying to achieve high-performance with less complex techniques. Currently, three basic technologies are employed for horizontal motion control:

- AC motor with double helix head guide. This is the least expensive and causes full excursion of the head assembly, thus precluding line seeking logic.
- Stepper motor. This is considered the most reliable and easiest technology to implement, permitting exact horizontal stepping of the head, with very little electronics required.
- DC-servo. Typically used for high performance, high-speed designs. Very accurate and very expensive. Requires feedback from a tachometer and extensive electronics.

Most printer designers employ stepper

motors for simplicity and reduction in control electronics. Head movement is accomplished using either a belt drive or wire and pulley arrangement. Either design is reliable and for the most part easy to replace in case of breakdown.

The basic stepper idea is used by Tritell in the printer mechanisms it has designed for such manufacturers as: Dataroyal, HI-G, and Infoscrite, to name a few. The entire Tritell design is simple and employs a single piece structural foam chassis, horizontal guide journals for the head assembly using a stepper motor and a wire and pulley arrangement. The electronics include print hammer drivers, character generators, and paper movement electronics.

The Tritell design has achieved a great deal of favor among printer manufacturers and has set the pace for simple reliable designs. Moreover, even though the basic design is not complex, it does permit the addition of the characteristics that the printer manufacturer has determined to be the most desirable.

Dataroyal, for example, has expanded on the basic design and offers a number of features in their IPS-5000 series. The 5000-C, for example, clips along at 165 cps, and is priced at \$1075(100) for the 80-column version and \$1110(100) for 136-column models.

The Dataroyal models come standard with a 9 × 9 dot matrix font, graphics, and a 500-character FIFO buffer. Since communication protocol is important, all printers in the 5000 series employ an XON/XOFF protocol. Adding an additional 2K buffer costs \$100 for the 80-column models and \$75 for the 136-column units.

Another manufacturer employing the Tritell design is Infoscrite in their series 500 and 1000 dot-matrix printers. The Model 1000 is a 200 cps printer with 136-column output. Like all Tritell designs, the printer offers bidirectional printing, vertical tabbing, and line seeking logic. The 1000 is priced as low as \$910(1000).

Besides the basic printer functions, Infoscrite has added value by offering such features as underlining, superscripts and subscripts as standard functions, along with the ability to print correspondence quality printing at 10-cpi, a capability being sought after in multi-purpose dot-matrix printers.

What may become the premier Tritell designed printers are being offered by Hi-G, a fast-growing manufacturer of electronic components. Working in concert with Tritell, Hi-G is offering the models 9/80 and 9/132 dot-matrix printers.

Although the basic printer design (chassis, head shuttle and carriage) are Tritell's, Hi-G supplied the head. The head is a 9-wire type with a modified clapper. The new design allows each wire in the head to be operated at 1000 Hz continuously making it ideal for 200 cps operation for 7 × 9 matrix characters, or 165 cps for 9 × 9 characters.

In addition, Hi-G is using a transformer of their own design, that will perform under power levels as low as 75V. This could make it almost brown-out proof (according to the Oct., 1981 issue of *Printout*, a Newtonville, MA-based newsletter to the printer industry).

Interestingly, Hi-G is planning to take on the well-ensconced Epson in the low-end (under \$1000) arena. The 9/80 is priced at \$995 and is expected to be well under \$600(500), while the 9/132 is planned for \$1100 single quantity.

Even though the Hi-G printers offer fairly standard features - 150 cps at 10 cpi bidirectional, 75 cps double-pass correspondence mode, and a paper slew rate of 4-in./sec - the real value is planned in the reliability.

A case of specsmanship

Following a very similar tack is Integral Data. According to Carl Peters, the manufacturer's reliability engineer, they test each printer with external

(continued next page)

equipment and check up on timing modes; they also look for parts failures that will never show up in a self-test diagnostic.

Peters and others contend that it's very easy for a manufacturer to specify certain operational parameters, that can in fact be met if tested in carefully controlled conditions. However, in real operation the printer may not measure up. Tritell's v.p. Don McCombs explains that if you know enough about printers, it's a fairly easy matter to develop a test program that makes a mechanism really look good.

To avoid finger pointing, manufacturers like Hi-G, Integral Data, and others are looking for failures. Integral Data's Peters says that his company not only tests before shipping, but carefully watches printers in the field to gather data on exactly what is failing and when - degradation of the MTBF based on parts that can wear out.

But what does MTBF mean? Tritell's McCombs isn't convinced he understands what manufacturers are saying about MTBF. Typically, MTBF can be calculated based on an 'n' sample of printers operating over a specified time period, printing an exact line length with a known number of characters, line feeds, and head firing cycles. Based on this information, an MTBF can be cited. However, warns Peters, it may look good in advertisements, but what's really meaningful is data from failure analysis.

You probably should ask what the exact conditions were during testing, what type of failures were encountered at what rates, and what the manufacturer has done to solve the problem.

The failure data becomes extremely important, since it determines exactly what impact on cost of ownership the printer will have. By garnering such failure information, you'll be able to spare inventories and lower service costs.

Coupled directly with MTBF is serviceability. This determines how easily you can service the printer in case of a failure. Most believe that the more gears, bells and whistles a printer has, the more difficult it is to service. Consequently, manufacturers are employing fewer complex and costly designs in favor of the simpler direct approaches

previously outlined.

On-line Microcenters Inc.'s president Dennis Mandell explains that his company deals in very high volumes, so they must have guaranteed reliability. Consequently, they have over 40 points that they check before committing to carrying a printer. And they try very hard to match the correct printer to the application. Someone who is using a printer for hobby purposes, for example, may only need a low-duty cycle printer since the duty cycle will be minimal. But for the business application, a tougher machine designed for continuous use must be specced, explains Mandell.

Byte Industries president David Pava agrees. He points out that cost is number 4 on the list while reliability and serviceability fit the number 1 and 2 slots. Pava says that no matter who is buying a printer for what purpose, they want quality and are willing to pay for it. With low-cost (under \$1000) printers, you can find almost any feature you want. Because of so much overlap and specsmanship, what quality really is can become pretty blurred, notes Pava.

Considered the benchmark of reliability and quality is the Texas Instruments Model 810. This \$1645 (single quantity) printer doesn't offer any earth-shattering technology, but does have a long track record of reliability and ease of service. What you get is a printer that handles data rates from 110 to 9600 baud, has a 132-column wide adjustable carriage, a 10 cpi pitch, 256-character buffer and a medium print speed of 150 cps. In addition, you can add options such as expanded and compressed printing (8.25- or 16.5-cpi), international character sets, and higher impact print heads for up to nine copies.

One of the features that makes the TI 810 an ideal buy is that it employs real standards on the serial interface that can be easily configured, via dip switches, to meet the idiosyncrasies of your interfaces.

The ability to interface and provide maximum throughput are also important issues to OEMs. Unfortunately, not all RS-232C is defined the same way. Consequently, interfacing the printer may not be as straightforward as promised. The problem may be ex-

acerbated with parallel interfaces. Even though a manufacturer may offer 'Centronics' type interfaces, the key word is type and may not follow the guidelines laid down by Centronics for a parallel interface.

Like MTBF, throughput is ill-defined and can mean whatever the manufacturer thinks it means. However, actual throughput is related to exactly how fast the characters are being put down on the paper in relationship to the speed the data is being sent to the printer. Overly fast data rates working in conjunction with small buffers can be more of a hindrance than a help. Even with bidirectional logic-seeking mechanisms, a printer which is handshaking too frequently can severely impact overall system throughput.

To improve throughput of printers, manufacturers are employing a number of techniques. Specifically, buffers are getting larger, with some makers offering optional buffer sizes (in certain cases as high as 6K-characters), multiple communications protocols (XON/XOFF, ETX/ACK, and none). Moreover, head designs are offering faster throughputs by operating at maximum speeds (1000 Hz), and the use of fans has obviated the necessity of either stopping or slowing the printing to allow cooling. Still, many believe that real throughput must be measured at specific print densities at fixed data rates and should be measured as the time between characters - the non-print time. The overall throughput, in most cases, would be less than that often specified.

Interestingly, Anadex's marketing v.p. Ken Matthews is the first to point out the pitfalls of relying on specs supplied by printer manufacturers. He thinks that the real test of a printer's specifications is to put it in a real environment, and see if it does the job. He also believes that if you're committing real dollars for as important a peripheral as a printer, you should develop a check list of those functions you consider important, and gauge each printer by those guidelines - and not rely solely on what the manufacturer says.

Software makes the printer

With the basic foundation of printers being the same - chassis, shuttle mech-

anism and carriage, head, and cooling – the real functional differences come in the added software implemented via onboard firmware.

The firmware available allows a printer to handle various communication protocols, print different character sets, and handle paper in various ways.

The firmware works in concert with the electronics – the engine – to provide the expected font and format versatility. Moreover, the attributes of the firmware permit such things as downloading of control sequences, special character sets, or even redefining the firing sequence for special graphics applications.

This ability to download to the printer has become almost standard across the board, even on printers that don't appear to offer the feature. You can usually perform downloading by setting up the correct escape sequences.

The Centronics Model 352, for example, priced at \$1795, is based on a design licensed from Brother of Japan, and sports a number of standard features such as: 200 cps bidirectional logic seeking operation, a 7 × 8 dot matrix font with 9th wire underline, ability to handle cut sheet for continuous form, demand form capability, 96-character ASCII set and 7 international ASCII sets, replaceable printhead, subscript/superscript capability, and self-diagnostics on power up..

In addition, the 40-lb printer handles data rates from 50 to 19.2K baud and handles XON/XOFF, reverse channel, and DTR protocols; it offers RS423 via an optional RS449 adapter. Moreover, the Model 353, at \$2495, offers all the same features but adds an operator-programmable control panel with liquid crystal display, and has the capability of accepting a user-defined character set from the host computer.

Should your needs differ from what Centronics is offering, changing the printer's characteristics is simply a matter of changing the PROM.

(Editor's Note: This is only the beginning! Next month, George will continue to discuss the options available to those shopping for printers.)

Books

Books

BUSINESS SYSTEMS FOR MICRO COMPUTERS: Concept, Design and Implementation

by William D. Haueisen and James L. Camp
published by Prentice-Hall, Inc.,
Englewood Cliffs, NJ 07632, 1982

This book is a landmark on the subject of microcomputers, being the first written by professionals in business management. The emphasis on management principles is evident throughout the book, which is intended to be readable by a wide range of people concerned with developing systems for small computers. The authors state that the book is designed as a text for a course in business management, as a supplementary text for a course in systems design, as a resource to the businessman and as a tool for the system designer developing general business integrated system products.

Organized around the design process used by the authors, the work is divided into three major parts. The first part consists of preliminary preparations and concepts, and the development of a model business; the second part includes the detailed design of each of the business subsystems. The last portion discusses the integration of the system into the business. Selection of a computer is considered in the first section. The steps of the design are explained and then a model for a sporting goods wholesaler is constructed. The second segment takes each of the business subsystems and presents the forms it uses. Then the file structures are formed and the input screens are derived. The system procedures are modelled in flow charts and discussed.

Though the systems were intended to be implemented in BASIC, no source listings are provided. The last section of the book covers the implementation and testing of the system and user training; here also are found chapters on profit management, system security and the potential use of microcomputers in distributed systems.

Many operators of small businesses have little or no formal training in business management, so the concepts of profit management and management by objective might be new to a lot of businessmen. This book gives a practical introduction to the use of these tools. The authors suggest that the design in the book could be modified to suit many businesses and save many hours of decision-making about details already provided. Even though the structure of the sub-systems strongly reflects the intent to implement them in BASIC, only minor modifications would be required to implement the system in other languages.

The chapter which discusses the selection of a computer fails to explore in detail the relation of the implemented system to the mass storage media capacity. I would like to emphasize that the computer should ideally be chosen last. This is usually not practical, but every effort should be made to accurately estimate the actual mass storage requirements before purchasing the computer. Buying a machine which is too small to do the job is a common mistake of the first time purchaser. At best a small machine will cause errors because of excessive handling and changing of diskettes. At worst, the system will not be able to access important information it needs during many parts of the processing and will cause sacrifice of security and other vital functions.

I highly recommend this book to both businessmen and system designers for microcomputer business applications.

Renew

If your subscription began last August, remember to get in touch with us before your subscription expires. If you procrastinate, you'll surely miss some of the "goodies" we have planned for you. A helpful hint: if you utilize *Lifelines/The Software Magazine* as a business aid, your subscription is tax-deductible. So stop what you're doing for a moment and take the time to fill out the renewal form you've received in the mail. Please send your check right away. Or you can get out your VISA or MasterCard and call *Lifelines/The Software Magazine* Subscription Dept. at (212) 722-1700. The address is: 1651 Third Ave., New York, N.Y. 10028.

May 21, 1982

Hello, at long last...

Where were we? Oh, now I remember. A while back, (in early March if you must know), I submitted something akin to the column which follows. I'd like to report on some neat trip that kept me away since then, but that's not the case. I'd be equally happy excusing my absence with the inside scoop on some super R & D project that had drained me of my usual energies, but that's not the case either. What did ensue were protracted and tasteful discussions about the need for me to tread more lightly on certain eggs (or some such equally scrambled metaphor). Anyway, for your belated amusement, here's the rewrite:

As I'm sure you've all read hundreds of times, the single chip microprocessor heralded the age of the home computer. I'd like to share a few observations about this phenomenon.

Consider the family which springs for a home computer so that everyone can get in on the marvels of modern technology and learn a bit about programming in the bargain. Some months later, only the kids have made measurable progress with the family computer. Mom and Dad have long since abandoned their efforts to learn BASIC. (Ironically, many parental types become terminally confused when they encounter the DIM statement.) Do you know any families like this? I sure do.

Many people are discovering to their considerable dismay that there's more to computer programming than buying books. Now, I'm sure most of you can figure out why the computer industry downplays this obvious fact. The promise of simple computer solutions, which even a child can understand, has sold lots of machines.

As I see it, the wherewithal to under-

stand computers and use them creatively is mostly an accident of birth. It's like painting, music or sports; if you've got that certain special aptitude, the sky's the limit, otherwise...

To graphically illustrate how different people excel in different endeavors, we could assemble an all-star squad of computer experts and pit them against the Oakland Raiders in a test of overall adaptation to life's exigencies. It might go like this: Round one would be a debate about computers with one point to be awarded to the winning team; Round two would feature some full contact football, again for one point. The probable outcome of these two rounds would be a 1-1 tie, proving that in the grander scheme of things, everyone's more or less equal. Which half of this 'Battle of the Stars' would you prefer to watch? My seats are on the fifty yard line.

Until the mid-seventies, the point I'm trying to make (about computers) was deservedly moot. Before then, computers were huge, super expensive and rather unfriendly to boot. To learn about computers you had to attend scheduled classes at someone else's convenience, only to proceed at the plodding pace of your less gifted classmates. Now it's become ever so much easier. You can succeed or fail in the comfort of your own home. Undoubtedly, some who fail under the new system will miss being able to blame it all on a teacher's incompetence.

I have never liked loose leaf (ring) binders. These fiendish contraptions drove me nuts during my school days and they still do. Even with light to moderate use, they are brutal on paper. This explains why some folks make a handsome living selling gummed hole reinforcers. Unless these blasted loose leaf binders contain a certain thickness of pages, they do not lend themselves to stacking or storage in crowded bookshelves (except for the ever-popular circular shelves).

Perhaps with the most honorable and ambitious intentions, someone will sell you an expandable manual with less than half as many pages as the binder can hold. I call these ergonomic mismatches 'wedgies'. Whereas a few wedgies pose no special problem, many wedgies do. If you own a computer and you don't already have a roomful of wedgies, you will soon enough. In theory, wedgies should eventually assume the shape of real books, thus becoming self-correcting annoyances. Hah!

Believe it or not, I have turned this pet peeve of mine into a quasi-serious research project during which I managed to get several vendors to explain why their documentation comes in loose leaf binders. Their answers were all so similar that one character, W: (for 'Wedgemonger'), speaks for them all.

Z: Why do you use these nasty things?

W: Because when we update the manual, you can just stick the new pages right in place.

Z: Somewhere off in the distant future that just might happen but right now I'm looking at a quarter inch of pages in a binder meant to be two inches thick. Hell, viewed from the top, it looks more like a log splitting tool than a book.

W: You don't have to use our binder if you don't like it.

Z: Well, I don't like it but I still had to pay for it. If you are so sure that you'll be providing more and more addenda pages and if you remain so dead set against archival binding, why not just offer your manual as a flat stack of prepunched pages. That way everyone could bind the manuals as they please and you could pass the savings along. (Mentioning 'savings' to W: damn near causes him to suffer cardiac arrest.) At first glance, I can see how this

little docudrama might make me sound like a nitpicker, while making W: sound like the voice of gentle reason, politely explaining sound business judgment to a slow thinker. Don't be misled!

Here's the real story: especially in the case of software, complete manual overhauls are a fact of life. Typically, just a few supplemental pages are ever issued before a manual becomes obsolete (and therefore no longer supported with printed addenda). To remain current, you will have to purchase a revised manual. You began with one product and one wedgie; soon you will have one product with two wedgies and that may not be the end of it. Often, as your wedgie collection grows unabated, considerable expense is involved. I'm reminded of obstetrical fees...

So much for the obvious facts. I haven't even begun to ponder the subtle nuances. For example, why do most addenda pages come without punched holes?

Recently I had a chance to sit down with the proprietor of one of the most disreputable computer stores in the Northeast. Here's yet another mini-drama which more or less conveys what was said. This time, the featured players will be yours truly and C: (for [that notorious] computer store or possibly the crook who runs it - take your pick). Let me set the scene...

Most computer stores are respectable enterprises but some are not. It doesn't really matter to me; I happen to prefer reputable mail order firms. I have had nothing but good experiences dealing with Jade, WW Component Supply, Ancie and Godbout to name just a few (see note below). Actually Godbout (Compupro) isn't really a mail order firm but, since I am so fond of their entire product line, I took the liberty of mentioning them anyway.

NOTE: Had this column appeared in its original incarnation, the last paragraph would have included a sincere endorsement of Priority One. Unfortunately, since then, I have experienced a minor problem at their hands and a friend of mine has had an even bigger problem. Recently, I bought a pair of Tandon drives from Priority One along with a ten buck 'manual' - (Priority One part number:

TNDTM8M). This 'manual' neglected to suggest which of 256 possible arrangements of an eight position shorting plug would provide the user selectable options required by my disk controller. I called Tandon with every intention of castigating them for such worthless documentation. I spoke with Jay Tyzzer, Tandon's 8" Product Manager, who stopped me in mid-sentence as soon as the nature of my complaint became evident. This is what Mr. Tyzzer said, "Priority One, right? You won't believe how many calls we've gotten on this. What they sold you is a preliminary spec sheet. The next time we print one, it will have prominent copyright notices so that nobody will be tempted to dupe it and sell it as our manual". At this point, it was pretty clear that Tandon owed me nothing; nonetheless, Mr. Tyzzer courteously provided me with all the information missing from my ten dollar spec sheet and he suggested that I purchase the real manual direct from Tandon. (I did, it costs twenty five dollars, and it's a complete and professional document - P.S. I love my Tandons.) I called Priority One's toll free order number to complain about the ten dollar gouging and I was instructed to call another number at my own expense. No thanks, amigos; you keep my ten bucks and I'll tell the world about it.

In a separate incident, a good friend bought two Mitsubishi drives from Priority One (the ones which they advertise as being better than Qumes or Shugarts). His Mitsubishis did not work (except as read-only devices). When he complained (again, the toll free number was off-limits) he was told about a factory recall, due to some design oversight. He was given an RMA (Returned Merchandise Authorization) number and instructed to return the non-functional merchandise. This involved almost fifty bucks worth of shipping and insurance which Priority One would not pay for. My friend was told to expect a new set of drives within two weeks. Three weeks later he called and was told that at least two more weeks of waiting were assured. My friend had an immediate need for two double sided 8" drives, so he asked if he could apply the money he had spent toward a pair of Qume DT-8's. "Sure", he was told, but since his original order had been processed (less than one month), the heavy demand for Qume DT-8's had resulted in

a price increase of \$100 per drive and that, circumstances notwithstanding, the new price would not be adjusted downward. That was three weeks ago and he's still waiting.

Oh my gosh! I almost forgot the second playlet.

C: Mostly everyone appreciates retail computer stores. You don't. What are your reasons?

Z: It's your particular operation I dislike the most but, to answer the question as you posed it, there are several. Right off the bat, retail anything plus sales tax had better be something special to interest me. Seriously, if I had to pick the main reason why I don't like stores like yours, it would be that you rarely if ever get high enough markups to carry the kind of merchandise which I would choose.

C: Why do you think this is the case?

Z: Computers are just like any other consumer product. Perhaps people who open computer stores begin with an educated grasp of which items are the best and a commitment to sell nothing less. If these budding entrepreneurs don't quickly disabuse themselves of these lofty ideals, they'll go broke. The same kind of juggernaut mentality that wants to put a car in every garage, even if that means we all have to pedal, is very much in evidence, exploiting the public's computer ignorance for all it's worth. Right now, it's worth plenty.

C: Isn't public acceptance a more reliable barometer of value and quality than anything else?

Z: With no warning that I can recall, I was suddenly expected to accept Detroit's decision that vent windows were out and that plastic grilles were in. You must realize that when the taste, experience and judgment of designers and engineers takes a back seat to the dictates of business and marketing types, value becomes a charade and quality is no longer an issue.

C: I see. How does this relate to your opinion of my store?

Z: Simple; you sell what's most profit-
(continued next page)

able. It was bad enough when you just pushed the machines with the biggest profit margins. Now, many manufacturers of such equipment have decided to gild their wilted lilies with their idea of OEM software and you make even more. Times have changed I guess. When people sold patent medicines it was charlatanism, when they sold worthless land it was fraud but when you sell some underconfigured toy as a solution to someone's business needs, it's merely an education.

C: What about after sale support? No matter what you may think, our customers consider this invaluable.

Z: I'm sure they do... in the rare cases when they ever see any.

C: You said something about it being unlikely that you could find your 'dream computer' in a typical computer store. I'm curious about just what that machine would be.

Z: Fair enough, I'll tell you. Terminals are very subjective items but, for what it's worth, I would be happy with almost any terminal from Lear Siegler, TAB or Zenith. Printer preference is also highly subjective. My favorites (in no special order) are made by Anadex, Lear Siegler, NEC, Datasouth and Texas Instruments. (I'll probably like the Siemens ink jet printer once I get to try one.) Eight inch double density floppies or better with an appropriate controller would suffice for storage. Here, I'll skip specific recommendations because software makes all the difference. Whatever disk devices I might choose, they would have to be housed in their own enclosure with a discrete heavy duty power supply. That leaves us with the computer. I insist upon an S-100 system with twenty or more slots, a good cooling fan and a front panel. The mainframe would come from TEI, the front panel from WW Component Supply. Almost everything else, (especially the memory), would be Godbout products. There's one more important and inflexible requirement. Eight bit machines are capable of addressing 64K of RAM. I will not accept one byte less. No matter what 'good' reasons some designers think they might have for committing any part of a 64K mem-

ory map to ROM, I'll never be interested.

(Another dialogue winds down).

I wish that C: had more stamina. I was just getting warmed up. This is one topic I could really pursue almost indefinitely. Even if he didn't want to hear more, maybe you do. I'll volunteer this much more; just for fun, walk into the computer store of your choice and price a complete system, then announce that another store has quoted a lower price. Don't be surprised if you hear one or more of the following:

"The other dealer sells out-of-warranty or obsolete models".

"If you buy from us, we'll throw in lots of [costly and illegally copied] software".

"Agreed, we do charge a bit more, but that barely covers the expenses we incur subjecting every component we sell to intensive power-on testing (the dreaded 'burn-in')".

This last one is a gem. To the untrained ear, it sounds like a concerned merchant taking pains to ensure that your substantial investment will procure a reliable product. Sometimes this is true, often it is not! Permit me to translate. You have chosen a factory tested and inspected product which the dealer receives in a carefully sealed carton. Factory testing and inspection is far more thorough and sophisticated than anything retail outlets can offer (except in the case of the one notorious manufacturer who ships so many D.O.A. products that their dealers should test every one - or even better, refuse to sell them). Oh yes, the burn-in; the obvious scam is that it never gets done. In a store like C:'s that's the best you could hope for. Otherwise, the dealer has opened up your carton and is testing your purchase as promised. While you are eagerly awaiting delivery, the system you paid list price for is hard at work as demo equipment. Go ahead, get impatient, show up a couple of days before delivery is promised. Watch the truants from the local high school playing space wars on your new machine. Watch as they yank the joy sticks from their flimsy housings. Watch them dribble food and ganja ashes in between the keys. Watch the grubby fingers dance across your CRT, leaving a layer of ineradicable filth in

their wake. Burn-in or just plain burn? You decide; I already have.

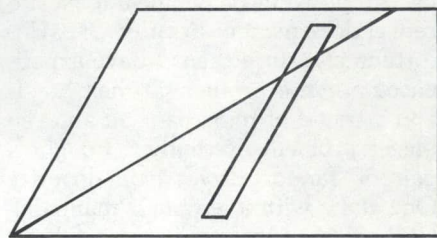
If you have stayed with me this far, it should be plain where I stand on retail computer outlets. In this regard, you may safely expect things to deteriorate even further. Selling computers in department stores is the dream of several manufacturers. What better way to isolate themselves from after-sale support? What an abominable concept! I don't care for department stores one bit. Except for clearance sales, full list price plus tax is the order of the day. Furthermore, I have yet to see the vaguest hint that department stores employ the kind of people whose expertise reaches or exceeds toaster oven technology. The salespersons I encounter seem to be either surly high school dropouts or crotchety old biddies who care about little other than their painful bunions. It should be quite a treat listening to them field the technical inquiries.

I used to wonder when I would get a revised product that I liked less than its immediate predecessor. Now I know. I like WordStar 2.26 a whole lot more than I like WordStar 3.0. They're both fine products, but I definitely prefer the older version. In addition, the exorbitant cost of the WordStar 3.0 Customization Notes provides another good reason to remain rooted in the past. I know quite a few others who concur.

Finally, when I rewrote this column I was quite surprised to find at least three computer puns near the beginning. Personally, I don't like puns and I try not to write them, but nobody's perfect. If you happen to find puns amusing, start searching.

See you soon,
Zoso

(Editor's Note: These are Zoso's personal opinions. We welcome comments and replies.)



Lifeboat Associates' SB-80 A Comparative Analysis

Michael J. Karas

What is SB-80?

Lifeboat Associates has announced SB-80 as an operating system software product for eight bit microcomputer systems with 8080, 8085, or Z80 type microprocessors. The product is intended to provide a software development, utility, and applications execution environment in the normal manner expected for an operating system. SB-80 defines a generalized computer interface in two areas:

- First there is the interface to the computer hardware whereby the SB-80 operating system defines standardized methods of interfacing to mass storage devices, memory, and attached peripheral devices.
- The other standardized interface provided by SB-80 is the human/operator level interaction that allows commanded action, to the computer, from the operator console to initiate execution of various computer applications programs.

The purpose of SB-80 is to provide a high performance 8-bit microcomputer operating system with many features not previously found intrinsic to small computer system software, while at the same time not using *too* radical and new an approach; so the potential user's investment in development packages, utilities, or applications may be applied for use with SB-80. The user program compatibility implied for the SB-80 installation is that SB-80 has been designed around the general physical file structure and DOS (Disk Operating System) call standards placed into wide use and subsequent *de facto* acceptance when Digital Research Inc. introduced their CP/M 8-bit microcomputer system control program.

Why Another Operating System?

This article will attempt to provide enough information about SB-80 so that the reader may begin to appreciate the importance of its existence. In no way can, or should, this short analysis describe all features and possibilities offered but it will point out key advantages of using SB-80 over the other 8-bit operating systems. Information presented herein will use a comparative method to introduce SB-80 by contrasting it against the CP/M-80 control program that brought into wide acceptance the basic format for interfacing to 8-bit microcomputer hardware.

SB-80 Evaluation Environment

Lifeboat Associates kindly provided information and the software to allow evaluation of the SB-80 product. The SB-80 DOS (Disk Operating System) and CLI (Command Line In-

terpreter) were provided on an 8 inch floppy diskette in a format suitable for use with a 56K byte memory setup. Documentation and users' manuals were also provided upon an 8 inch diskette for me to print out. (Note that a user obtaining SB-80 by actual purchase would receive a hard copy version of the manual.)

I adapted the BIOS from an old CP/M-80 2.2 installation on an IMSAI 8080 with 56K bytes of memory to be compatible with SB-80 and changed the cold boot loader such that it would load the reserved system track images of SB-80 instead of the previous CP/M-80 environment. Through completion of this effort an SB-80 disk was made to boot up on the IMSAI with the Tarbell single density floppy disk controller. The software was then analyzed for performance under this environment.

Secondary attempts were made to generate bootable system images of SB-80 on two other computer systems. One system was a Callan Data Systems CD100M with the Monolithic Systems MSC8009A Z80 CPU and Micro Computer Systems 9205 Winchester Disk Controller connected to a 5 megabyte Winchester drive. The BIOS on this hardware configuration included drivers for a pair of minifloppy disk drives, in addition to the Winchester disk drivers. Being familiar with the BIOS requirements for SB-80 and having originally written all of the BIOS software for the Callan unit, I was able to have SB-80 up and running within about 3 hours. (Contrast that with nearly one and one half days work for the IMSAI 8080 version.)

The last system attempted was a Cromemco Z2 system with attached ICOM 3812 double density floppy disk drives. At the time this article was written SB-80 was not operational upon the 3812 system; the project required considerably more work because source code for the CP/M-80 BIOS, cold boot loader, and disk I/O drivers was not available. New coding was attempted for the BIOS and loader while I elected to adapt the I/O code resident within an EPROM to the SB-80 setup. Additional effort to achieve an operational SB-80 system on the 3812 will be documented as a letter to the editor in a later issue of this magazine.

Analysis of the operation of SB-80 in the various hardware environments was performed to ensure that the new operating system was indeed file compatible, physical device I/O compatible, and compatible with the host disk sector de-blocking of similar CP/M-80 BIOS's. Testing has revealed that the advertised compatibility exists and the same files on various disk media may be manipulated both under SB-80 and CP/M-80. In addition, the testing on the Callan Unit has shown that the "much more complicated than normal" multiple controller BIOS is workable in the SB-80 environment.

(continued next page)

SB-80 Structure Comparison

The SB-80 operating system software package, in an installed configuration, consists of three modules as follows:

- 1) The BIOS (basic I/O system) contains all system hardware interface driver software. This module is analogous to the CP/M-80 BIOS and is reasonably compatible except as noted in a subsequent paragraph.
- 2) The DOS part of SB-80 is for the most part a file manager that accepts a standardized set of logical system interface conventions and converts them to the appropriate sequence of real I/O functions to be handled by the BIOS. The SB-80 DOS is functionally compatible with the CP/M-80 BDOS module except that the SB-80 DOS contains more functions and treats certain system functions in an expanded manner.
- 3) Human interface to the computer via the console device is performed by the Command Line Interpreter (CLI). Roughly equivalent to the CP/M-80 control program "CCP", the CLI accepts operator commands and takes action accordingly. Two types of commands, intrinsic (built-in) and transient (user programs) are possible. The user commands are typically the development packages, utilities, and applications program which the computer system is essentially intended to execute. Intrinsic commands allow operator control over everyday operations without the need for a disk full of simple-minded utilities. Examples of both types of functions are included in later sections.

System software for SB-80 is placed upon a "bootable system disk" with the DOS and BIOS on reserved system tracks, typically the first two tracks of the system disk. They are initially read into the upper part of system memory by a cold boot routine. Usually a shadow EPROM or system monitor routine reads a bootstrap program into memory from track 0 : sector 1. This small program then reads the rest of the reserved system tracks into memory, at the appropriate locations, and then jumps to the startup entry point of the BIOS for system initialization. The SB-80 DOS and BIOS are only read into the host computer memory at "cold boot" startup. Restart as a "warm boot" does not reload the DOS as in CP/M-80 where the BDOS is reloaded with each warm boot. In an SB-80 installation, the warm boot and/or cold boot routines may initiate execution of any user program desired, or a special reserved name file "SYSTEM.CLI" may be read into memory. "SYSTEM.CLI" is the SB-80 command line interpreter and would be used at this point to execute operator-typed commands.

For a 56k byte memory microcomputer system the chart in Table 1 shows the memory map layout for SB-80 as compared to a similar CP/M-80 2.2 installation. This map is the memory layout for the IMSAI 8080 system where the software was initially evaluated.

SB-80 I/O Software Differences

An SB-80 boot loader program, such as that on track 0 sector 1 of the evaluation IMSAI system disk, needed modification only to the extent that the load address for the system image

was different from that of the other operating system. I was able to get the SB-80 DOS and my Tarbell SB-80 BIOS to fit on the 52 available sectors of the reserved system tracks of a single density system diskette due to the small size of the Tarbell single density BIOS. Please note that many BIOS implementations, especially those for multiple disk controller setups, are very large and thus the DOS and BIOS may not fit on two single density 8-inch disk tracks. In these cases the bootable SB-80 system image must be placed upon the reserved tracks of a higher capacity media or else more disk tracks should be made available for the system image.

The structure of an SB-80 BIOS is "almost" the same as that required for a CP/M-80 Ver 2.2 implementation. Cold boot entry to the BIOS assumes that the bootstrap or PROM loader has brought the DOS and BIOS into memory. Cold boot processing in the BIOS then initializes all necessary peripheral I/O devices, such as UARTS, timers, and external printer electronics. The jump addresses in the base page of RAM, locations 0-1-2 and 5-6-7 are programmed for jumps to the warm boot entry to the BIOS and the entry point to the DOS (i.e., JMP C606H for 56k) respectively. Once this has been performed, the code may perform an autoloader of a user program or load the SB-80 CLI (command line interpreter) for direct execution. The loading of CLI is assumed to be from file "SYSTEM.CLI", assumed to be upon the system disk (and placed as the first file to optimize loading speed), and uses a coding sequence like the one below. (See Table 1 for a comparison of SB-80 and CP/M-80 memory layouts.)

```
dosent equ 0c606h      ;DOS entry for 56 K system
      mvi e,defdriv    ;set (e) for drive to load CLI from
      mvi c,137        ;function code to have DOS load CLI
      call dosent      ;call entry to SB-80 DOS to load
                      ;..returns with drive code in (c)
```

If at this juncture cold boot startup of the CLI is desired, the cold boot processing jumps directly to the base of the CLI. Entry at this point may autoloader an applications program for execution, if an autoloader command line has been placed into the CLI image. A jump to the CLI base + 03H will inhibit autoloader.

```
clibase equ 0b600H     ;cli entry for autoloader (56K)
clinal  equ clibase+3  ;cli entry inhibits autoloader
      jmp clibase      ;end cold boot processing
                      ;..and go to the CLI
```

Autoloader of a command file for immediate execution is very similar to the procedure used in CP/M-80. Through use of a debugger the file SYSTEM.CLI is read into memory at address 0100H. The debugger is then used to insert an autoloader command line into the file image at address 0108H. The length of the command line, up to a typical maximum of 127 or 79H bytes, is placed at address 0107H. The image is then resaved onto the system disk. The chart in Table 2 shows how the first part of the SYSTEM.CLI file appears both without an installed autoloader command and with an installed autoloader command. Also shown is the procedure to install the autoloader command and resave the SYSTEM.CLI file back onto the disk.

Autoloader installation with SB-80 is a lot easier than with CP/M-80 because only the CLI image file needs to be manipulated, as shown in Table 2. With CP/M-80 the autoloader command is installed, in the same manner illustrated in Table 2, in the base area of the bootable CCP image. Since the CCP

is on the system tracks the user implementing autoloading has to perform a SYSGEN operation to move a patched CCP/BDOS/BIOS image file to the system tracks.

Warm boot entry to the SB-80 BIOS does not reload the DOS portion of the operating system. The technical reason for this system function is twofold:

- 1) Warm boot in some applications can be faster since the 01400H byte DOS need not be reloaded and the BIOS is correspondingly smaller through elimination of the reload code.
- 2) SB-80 supports an online intrinsic printing spooler. Current status, file pointers, and the data area associated with the spooler functions are contained inside of the DOS area of memory. Reloading of this area of memory through each transient program exit and subsequent warm boot would clobber the currently valid spooler status, preventing spooler operation over warm boots.

The normal warm boot processing within the BIOS simply jumps to a code sequence like:

```
dosent equ 0c606h      ;DOS entry for 56 K system
logdriv equ 00004h    ;page 0 logged drive byte
clibase equ 0b600H   ;cli entry for autoloading (56K)
clinal  equ clibase+3 ;cli entry inhibits autoloading
        lda logdriv   ;get current logged drive
        push psw     ;save drive code
        mvi e,defdriv ;set (e) for drive to load cli from
        mvi c,137    ;function code to have DOS load CLI
        call dosent  ;call entry to SB-80 DOS to load
        pop  psw     ;set drive to log into (c)
        mov  c,a     ;set drive to log into (c)
        jmp  clinal  ;back to command line interpreter
                    ;..if no autoloading wanted on
                    ;warm boot
```

Note that certain parts of the above code sequence may be shared with the cold boot processing. The warm boot entry could also reset the page 0 jump addresses if desired, like many CP/M-80 BIOS warm boot procedures. However, reprogramming of the page 0 jumps is not really necessary unless the transient program has inadvertently clobbered page 0 of RAM. (In most cases if this has happened you might as well cold boot via the reset switch because Murphy's Law states that the rest of RAM is probably also gone!)

SB-80 File Compatibility

SB-80 is advertised to be file compatible with CP/M-80. The claim is applicable at the CP/M-80 Ver. 1.4 or CP/M-80 Ver. 2.2 User Area 00 with no file attributes set. SB-80 has some particular differences in the way files are handled, so that additional capability can be placed in the operating system. I have ascertained through use of SB-80 that the claim is indeed true. However, if a new SB-80 operating system user is converting from CP/M-80 it would not be wise to freely intermix file storage media between the operating systems. The old CP/M-80 disks will generally operate O.K. in the SB-80 environment, but SB-80 created files will drive your old CP/M-80 system crazy - if the CP/M-80 BDOS can even see the files in the directory. (*Editor's Note:* This is true when files over 8 megabytes in size are employed, or when SB-80 file attributes are used.)

The technical reasons for the file handling differences will be

explained after we present the justification for SB-80's new approach. The designers of SB-80 decided that the CP/M-80 maximum file size should be eliminated. This size for CP/M-80 is 8 megabytes, derived from 256 possible 16K byte extents for each file as supported by the CP/M-80 directory structure. Lifeboat Associates decided to use two byte extents to allow for 65535 16K byte extensions, putting the maximum file size at 4 gigabytes. A quote from the SB-80 user's manual states that "the user should feel comfortable with never running up against using the maximum file size because if his data base is that big he shouldn't be using a microcomputer".

SB-80 directory entries and/or system file control blocks contain data as shown to the left in Table 3. Corresponding formats for a file control block or directory entry under CP/M-80 are shown to the right. The contrasting use of certain bits/bytes results in CP/M-80's confusion with SB-80 files not kept under 8 megabytes, 16K or less per extent, within the user 0 area, and without any attributes set.

SB-80 Intrinsic Command Enhancements

The SB-80 command line interpreter contains a large number of built-in commands that allow operator use of the computer system and manipulation of the disk file data. In comparing it to CP/M-80 version 2.2 with the following well-known commands:

DIR, ERA, REN, SAVE, & USER

SB-80 marches in with the annotated list of commands given below:

- ATTR Set file attributes such as public, write protect, invisible or remove all attributes.
 - BATCH Define a file as console for batch processing as "A-CON filename" where a file contains letter, string or command sequences for processing in a job mode. (This replaces CP/M-80's SUBMIT and XSUB.)
 - BPINI Initialize the background print function to allow the resident print spooler to be used.
 - BPST Report status of background print function, i.e. initialized, dormant, error conditions queue full, etc.
 - CONT Continue suspended background printing if queued files currently being printed had previously been suspended.
 - DEQ Remove a file from the background-print queue. A single previously requested file for print may be taken inactive from printing.
 - DIR Directory of files. See example following list.
 - DIRA Directory with file attributes and owners. See example following intrinsic command list.
 - DIRI Directory including normally invisible files. See example below.
 - ERA Delete a file or wild card selection of files with oper-
- (continued next page)

ator prompt for delete confirmation.

ERA/N Delete a file or wild card selection of files without delete confirmation.

KILL Cancel background printing activity.

QUE Place a file on the background-print queue. Files are printed in the order they are placed upon the queue.

REN Rename a file to operator-specified name. Wild card selection is possible and SB-80 user friendly file name order is possible (see below).

SAVE Create a file from the memory image of the user program area (UPA). Useful for program patching and system generation work.

SBST Display status of the SB-80 system including file size limit flag, the case translation switch, the file control block compatibility switch, command file search system selected, and the current user number and SB-80 version number.

SRCAL Operator specifies that following command file searches (.COM files) are to be performed first from drive A: and then from the currently logged drive.

SRCLA Operator specifies that the following command file searches are to be performed logged drive first and then on the A: drive.

SRCLO Allows the operator to modify SB-80 program load scanning to the currently logged drive only line in CP/M-80.

STCLI Specify an alternate program and/or drive for loading at the next warm boot time instead of the normal SYSTEM.CLI file.

STCPM Set CP/M-80 file-size limitations. This is used to flag the SB-80 DOS so that it will not access files beyond 8 megabytes if the files already exist, and not permit creation of files beyond 8MB if being written. Forces the DOS to deal with extents, and random record numbers in the way that CP/M-80 handles them.

STIO Set I/O byte. In CP/M-80 "STAT" was used for this function. This byte is the well known Intel IOBYTE located in SB-80 page 0 at address 4. If the BIOS supports I/O device driver steering based upon the value of this byte, it is easy to use CLI to change it.

STSB8 Set SB-80 file size capabilities to normal SB-80 mode, in terms of dealing with extent bytes, and random access record numbers.

SUSP Suspend background printing without abort. (Phone rings and the printer is too loud or a new box of paper is needed.)

TROFF Turn off translation of lowercase to uppercase letters in SB-80 command lines to allow operator specification of file names and program entry parameters in the full keyboard character set.

TRON Turn on translation of lowercase to uppercase letters to prohibit lowercase character file names from getting on the disk. This is the default mode, as in CP/M-80.

TYPE Display a file at the console. File assumed to be ASCII text, but the CLI does strip upper bits like those commonly found in word processor files.

USER Change user numbers to allow single user access to another work area of the disk media. Similar in function to the CP/M-80 user function, the primary usefulness is in hard disk systems to allow segmentation of several system users' programs and data files.

User interface to the command line interpreter from an entered command editing standpoint is quite similar to CP/M-80. Implemented editing functions in SB-80 are backspace with erase, rubout/delete for character erase via echo, CTL-X to erase the whole current command, CTL-U to quit the current command line but not erasing it from the screen, and CTL-R to retype the current command line.

Other characters like tab, and linefeed are accepted by SB-80's CLI. Like CP/M-80, CTL-C initiates a warm reboot of the CLI and CTL-P is a printer echo toggle command that may be used at any time. I found one annoying thing regarding CTL-P, however. If an active print spooling queue is being printed, and CTL-P is activated, the console output goes to the printer interspersed with the file currently being printed. (Some people do put their printers in another room!)

Another feature of SB-80, (which seems to be mostly a confusion to a well-seasoned CP/M-80 user like this author), is the ability to have "USER FRIENDLY" bi-directional file names in multiple file name parameter CLI commands, and in most Lifeboat-supplied utilities. The concept is best presented as an example. The following two command lines perform equivalent functions under SB-80. The first command line is the format typical for the equivalent CP/M-80 command.

```
A>REN NEWNAME.FIL=OLDNAME.EXT
```

```
A>REN OLDNAME.EXT NEWNAME.FIL
```

The following examples will show some neat features of the SB-80 CLI commands for directory display, attribute setting and corresponding display:

```
A>DIR<cr>          <== Let's look at a partial disk
                    directory in the same format as
                    CP/M-80 Ver 2.2.
```

```
A: SB80      COM : BIOS56  ASM : DISKDEF  LIB : DU-V75  COM
A: CLI       HEX : COPY    COM : DOS      HEX : DUMP    COM
A: EJECT     COM : LIST    COM : PIP     COM : STAT    COM
```

```
A>DIRA<cr>        <== Look at same partial disk directory
                    with displayed file attributes and
                    user numbers.
```

```
D: FILENAME.EXT WPS  Usr  FILENAME.EXT WPS  Usr
A: SB80      COM    0   : BIOS56  ASM    0
A: DISKDEF   LIB    0   : DU-V75  COM    0
A: CLI       HEX    0   : COPY    COM    0
```

```
A>ATTR SB80.COM /S<cr> <== Let's change one file to an invisible
                        one with the built in attribute set
                        command.
```

```

A>DIR<cr>          <== Note how SB-80.COM is no longer seen
                   in the standard directory listing.

A: BIOS56  ASM : DISKDEF  LIB : DU-V75  COM : CLI    HEX
A: COPY    COM : DOS     HEX : DUMP   COM : EJECT  COM
A: LIST    COM : PIP     COM : STAT   COM : SYSTEM  CLI

A>DIRI<cr>        <== But all files including invisible
                   ones are displayed with this one.

D: FILENAME.EXT WPS  Usr  FILENAME.EXT WPS  Usr

A: SB80     COM  S  0 : BIOS56  ASM      0
A: DISKDEF  LIB  0 : DU-V75   COM      0
A: CLI      HEX  0 : COPY     COM      0

A>ATTR SB80.COM /X<cr> <== Reset SB80.COM back to normal.

A>DIR<cr>        <== .. and we see it again.

A: SB80     COM : BIOS56  ASM : DISKDEF  LIB : DU-V75  COM
A: CLI      HEX : COPY    COM : DOS     HEX : DUMP   COM
A: EJECT    COM : LIST    COM

```

The background print spooling features are among the capabilities I liked in SB-80 (despite the CTL-P gripe). The features are always present; six files may be queued up for print at a time, and current status may be monitored by CLI commands. And of course, the printing process can be suspended or aborted easily at any time while at the prompt level of CLI. The following examples show how to initiate a single file printing task:

```

A>BPST<cr>          <== Initial status monitor shows
Off                background print off.

A>BPINI<cr>         <== Initialize the spooler.

A>BPST<cr>          <== Now status is "no print in
Dormant            progress".

A>QUE DISKDEF.LIB<cr> <== Set this file on the queue for
                   printing. The printer started
                   chattering away at this point.

A>KILL<cr>          <== Abort print of that file as I
                   already have too many listings
                   it.

Status was :
Ready

```

Supplied SB-80 Utilities

In its distribution form, SB-80 is sold with a certain amount of utility software to enhance the usefulness of the package. The listing below summarizes the utility packages documented in the SB-80 user's guide.

SB-80 STANDARD DISTRIBUTION UTILITY PACKAGES

Hardware-Dependent Programs
Configured for Turnkey Systems by Lifeboat

SYSGEN Specially configured for SB-80 in that the reserved system tracks may be read or written to/from a file or a memory resident image. The SYSGEN also is claimed to move that essential file SYSTEM.CLI around as a new system disk is created.

COPYDISK This wonderful program (a version for my ICOM 3812 was included with the CP/M-80 Ver.1.4 that I bought from Lifeboat several

years ago), allows for a fast menu-driven selection of many copy options, including full disk copy up to an empty track and a verify (read only) of the source diskette.

FORMAT An essential utility for any disk-based computer system; it allows preparation of diskette media prior to use for file storage.

RESIZE A transient utility that allows the user to reconfigure the DOS, CLI, and image-resident, Lifeboat-supplied BIOS's to work with different amounts of memory. (*Editor's Note:* This feature will only be supplied with certain systems.)

CONFIG For systems where Lifeboat supplies the BIOS, the config program allows the user to set up his or her system for various terminal characteristics, printer types, and default CLI flags.

Non-Hardware-Dependent Transient
User Programs Supplied with SB-80

COPY A simple, menu-driven, multiple file, disk-to-disk file copy utility. The understanding is that the copy process with this utility is almost as fast as a full disk copy program.

DUMP Equivalent to CP/M-80's DUMP, this program allows the user to display file contents in hexadecimal and ASCII format at the console or on the printer.

LOAD A utility for converting Intel HEX format object code files into executable .COM files for execution under SB-80.

PIP A general purpose file mover program designed like CP/M-80's; it allows file movement between disks, user areas, files, and physical devices. With the evaluation version of SB-80 the PIP program provided appeared to be an independently-written utility from InfoSoft Inc. The PIP program accepts command formats that are uniform with the rest of the SB-80 system.

STAT This is a new Lifeboat-provided system status program that allows the user to monitor disk space, file characteristics, system configuration characteristics, input and output device assignments and so forth. During the evaluation of SB-80 I found the STAT program somewhat cumbersome to use because of the unwieldy 17K byte size. (For a simple STAT check for disk space the program takes far too long to load. It was even slow on the Callan unit with the 5 1/4 inch Winchester Drive.)

The differences between the Lifeboat philosophy and that of Digital Research seem to have a direct effect upon the type of utilities provided on a typical system disk. Lifeboat assumes the role of providing a more "finished and/or turnkey end user product" while CP/M-80 is more "roll your own" if pur-

(continued next page)

chased from the software house. Note that SB-80 for the end user would typically include copy, format, and finished BIOS programs - ready to run. Contrast this to CP/M-80, where the purchaser receives an editor, assembler, and loader and a manual that tells how to make a BIOS, COPY, and maybe a FORMAT program.

SB-80 DOS Interface Enhancements

Due to the ".COM file compatibility" that intrinsically exists between SB-80 and its grandparent model, all of the CP/M-80 BDOS calls to perform generalized file and logical device I/O are supported in SB-80. There are also a number of additional functions provided. For information purposes, the summary chart below itemizes the new DOS calls. Some of the standard DOS calls are treated in an expanded manner by SB-80. This greater level of interpretation exists primarily in the areas associated with file size, and random access record addressing.

SUMMARY OF NEW SB-80 DOS SYSTEM CALLS

Function # Purpose

128	Place a print file on the spooler queue. Equivalent to the CLI QUE command.
129	Reset a spooler queue file name. Like the CLI file de-queue function DEQ.
130	Abort all spooler queued files. Just like CLI KILL.
131	Suspend background print. (Like SUSP in CLI)
132	Resume background print.
133	Return spooler print status flag.
134	Set spooler print status flag.
135	Return pointer to base of the print spooler queue.
136	Initialize the spooler. Similar in function to the Command line BPINI command.
137	Initialize DOS and load SYSTEM.CLI from system disk.
138	Return current DMA address. Complement to the normal set DMA address function.
139	Get/Set SB-80 to CP/M-80 80 file size compatibility flag.
140	Direct call to BIOS function... Finally !!
141	Set user provided CLI file name for next warm boot.
142	Set CLI command load address for alien format COM files.
143	Set CLI exit to transient execution address for

alien format .COM files.

144 Load alien format file into previously defined load point.

SB-80 User Analysis Report

After a complete and detailed analysis of SB-80 for functionality, advertised compatibility to CP/M-80 Ver. 2.2, and usefulness of new features, certain comments related to the product, its implementation, and marketplace can be made. Please note that the opinions expressed here are those of the author and reflect my personal experience and expectations.

Use by an end user of SB-80 in a retrofit mode to upgrade an existing computer system should be typically discouraged. If a serious computer user on the technical side determines that he or she could make good use of the features described within this comparative study of SB-80, then the product should be considered on the basis of these questions:

- 1) Can the users of the upgrade system(s) deal with the small CP/M-80 incompatibility inconveniences if upgrade is only partial, or should SB-80 be installed completely with no backward looking at CP/M-80?
- 2) In a retrofit environment, does the simplistic level of operational discrepancy between the two systems warrant the level of confusion that could develop?
- 3) How much investment in presently purchased software can be directly transported to the SB-80 environment? For those programs and applications that cannot

SB-80 and CP/M-80 MEMORY LAYOUT COMPARISON

SB-80 Memory Address		CP/M-80 Ver 2.2 Address	
000H	Warm Boot Entry to restart "CLI" JMP 0DA03H	000H	Warm Boot Entry to reload CCP and exec JMP 0DA03H
0005H	Entry point to DOS JMP 0C606H	0005H	Entry point to BDOS JMP 0CC06H
0008H to 00FFH	Normal SB-80 Base Page FCB's and Default Disk Buffer	0008H to 00FFH	Normal CP/M-80 Base Page FCB's and Default Disk Buffer
0100H	User Program Area Base Address	0100H	User Program Area Base Address
B600H	Base of SYSTEM.CLI Command Line Interpreter when loaded	C400H	Base of CP/M-80 CCP Console Command Processor if loaded
C600H	Base of SB-80 DOS in resident position	CC00H	Base of CP/M-80 BDOS in resident position
DA00H	Base of BIOS for SB-80	DA00H	Base of BIOS for CP/M-80 Ver 2.2
E000H	Local PROM Monitor that has SB-80 load command. Load T0.S1	E000H	Local Prom Monitor that has CP/M-80 load command. Load T0:S1

Table 1

be transported, does there exist an equivalent set of programs and applications in the marketplace that are compatible with SB-80?

The use of SB-80 in a sophisticated end user application environment, such as an OEM purchasing hardware, integrated operating system, with the addition of an applications package, is highly recommended, based upon the product reviewed for this article. Simply said, if its use is divorced from the development arena where a CP/M-80 implementation has been used for some length of time, then SB-80 is the right product to use.

SB-80 AUTOLOAD INSTALLATION EXAMPLE

Note: Lower case characters typed by operator and <cr> indicates depression of the carriage return key.

```

A>ddt<cr>          <== Bring in a debugger with file
                    reading capability
-isystem.cli<cr>   <== Initialize an FCB to read file
                    "SYSTEM.CLI" from drive B:
-s5c<cr>          <== Read in the file image
005C 00 02<cr>
005D 53 .<cr>
-r<cr>            <== Note length at 4K bytes
NEXT PC          <== Look at image without
1100 0100        <== an autoload file name
-dl00,130<cr>

----- CLI Entry to allow autoload
----- CLI Entry to inhibit autoload checking
----- This byte = 0 indicates no
----- autoload
V          V          V
0100 C3 78 B7 C3 93 B7 7F 00 00 00 00 00 00 00 00 .x.....
0110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .
0120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .
0130 00 .

-iwordstar<cr>    <== Build ascii image of desired
                    autoload command
-m5d,65,108<cr>   <== Move from default fcb to the
                    CLI file image area
-sl07<cr>         <== Place command length into image
0107 00 08<cr>
0108 57 .<cr>

-Dl00,130<cr>    <== Dump the image out to see
                    installed command

----- CLI Entry to allow autoload
----- CLI Entry to inhibit autoload checking
----- This byte = 8 indicates command
----- for autoload is 8 char long
----- Start of command name
V          V          V  V
0100 C3 78 B7 C3 93 B7 7F 08 57 4F 52 44 53 54 41 52 .x.....WORDSTAR
0110 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .
0120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .
0130 00 .
~^C              <== Exit from debugger
A>save 16 b:system.cli<cr> <== Save the new autoload CLI image

```

Table 2

The initial release product performed well during evaluation and operated as could be expected. Some minor cosmetic flaws and a few operational bugs were encountered during the testing phase. These will have been communicated to technical personnel at Lifeboat Associates by the time this is printed. I hope in some small way to help make SB-80 a finer product.

Editor's Note: The authors report that this list of utility packages is now incomplete. Mr. Karas will note the additions in his update letter.

FILE CONTROL BLOCK COMPARISONS

byte #	SB-80 File Descriptor	CP/M-80 2.2 File Control Block
00	Drive locator/active flag. FCB-drive location Directory-00 active; 0e5H erased.	Drive locator/active flag. FCB-drive location. Directory-0e5h is erased 0-15 is user number
01 to 08	SB-80 file name in 7-bit upper and lower case characters. None of the upper bits used. If set not modified by SB-80 or SB-80 utilities.	CP/M-80 file name in Upper case 7-bit characters. Upper bits 01-04 are special application flags. Upper bits 05-08 reserved by software vendor.
09 to 11	SB-80 file name extent in 7-bit up/low case characters. Upper bits not used or modified.	CP/M-80 file name extent in Up case 7-bit characters. Up bit 09-Read/Only Up bit 10-System (Inv) Up bit 11-Not used
12	Low byte of extent numb	CP/M-80 file extent number.
13	High byte of extent num	CP/M-80 reserved S1 byte.
14	SB-80 Attribute Byte Bit 7 — Write protect Bit 6 — Public File Bit 5 — Invisible File Bit 4 — not used Bit 3-0 User number 0-15	CP/M-80 reserved S2 byte.
15	Record count this extent # 128 byte records	Record count this extent # 128 byte records
16 to 31	Disk block allocation for location of file on drive media. Single byte entries if disk < 256 blocks, double byte otherwise. Each block # allocates file size = to one disk group size. = < 16K bytes allocated	Disk block allocation for location of file on drive media. Single byte entries if disk < 256 blocks, double type otherwise. Each block # allocates file size = to one disk group size. = < 16K bytes allocated
32	FCB current record numb for this extent if sequential file access.	FCB current record number for this extent if sequential file access.
33 to 35	SB-80 Random access record number in low, middle & high byte order	CP/M-80 Random access record number. # 35- overflow if max file size exceeded # 33-34 low & high bytes

Table 3

Z80 Programming Tutorial, Introduction

Kim West DeWindt

Why A Z80 Tutorial?

This article responds to the many requests which have been stimulated by Ward Christensen's popular tutorial on the 8080. The 8080 is the precursor to the Z80 and its instruction set is a subset of the Z80's. Software written for the 8080 will run directly on a Z80 system (CP/M-80 is written for the 8080). However, the Z80 instruction set adds several powerful instruction groups that improve the speed and performance of a Z80 system over the performance of an 8080 system. Today, most CP/M-80 systems run on Z80 microcomputers, and the interested user may want to use the full power of her/his Z80 system. The tutorial on the 8080 is informative, but there are some gaps that need to be filled for the inquisitive Z80 user.

The intent of this tutorial, then, is to expand on the 8080 tutorial that has been running in *Lifelines/The Software Magazine*. I will spend some time discussing terms that are common to the Z80 and the 8080, but may have a slightly different meaning for the Z80. In addition, I will define those terms unique to the Z80. The architecture of the Z80 is similar to that of the 8080, so I will confine my discussions of Z80 architecture to hardware enhancements and additions that affect the programming of the Z80.

The majority of this article deals with the Z80 instruction set. Since the Z80 instruction set is a superset of the 8080, most of the discussion about the Z80 instruction set will center on the operation of instructions that go beyond the 8080. I will describe some details of the Zilog assembler mnemonics and conventions.

In the CP/M-80 environment, care must be taken when assembling and debugging Z80 opcodes. Assemblers blow up and debuggers become erratic when Z80 opcodes appear in the pro-

gram. I will show you how to deal with these problems, and enhance the performance of your system and software using the Z80 instruction set.

My Background

You're probably wondering who I am, and why I am writing about the Z80. I first ran into the Z80 in college, and I have been using it, along with a lot of other microprocessors, ever since. My latest project is a Z80B single board computer, Multibus@ compatible, that has 64K bytes of RAM, three serial ports and all kinds of other goodies on it. Since I work for a small company, I also wrote the monitor, CP/M-80 interface, and other assorted software for my board. As a result, I have become very familiar with the abilities, shortcomings and quirks of the Z80. It really is a nice little chip for an oldtimer (old by the current here today/gone tomorrow standards). By the end of the article, you should have a better understanding of the Z80 and be able to judge its advantages versus those of the 8080.

Summary

In this tutorial you will learn about:

- terminology and hardware that is different from and expands upon the 8080
- all Z80 instructions that are similar to the 8080 and unique Z80 instructions in great detail
- how to use Z80 instructions in 8080-based CP/M-80 systems
- some Z80 programming tips (including techniques to be wary of)
- how to write a basic Z80 program that takes advantage of the expanded instruction set

Outline

The tutorial has the following basic outline:

- 1-**Introduction:** What you are reading right now.
- 2-**Words:** An expansion of the 8080 tutorial terms including things that you need to know about the Z80, 8080 words that have different meanings in the Z80 universe, and general terms that were not covered in the list of 8080 terminology.
- 3-**Z80 Architecture:** A quick review highlighting additions that make the Z80 an 'improved' 8080.
- 4-**Zilog Mnemonics:** I'll cover these briefly so that you will understand the format of the instruction set in the rest of the tutorial.
- 5-**Moving Data:** How to move data within the Z80, and to and from memory. The Z80 has both 8 bit and 16 bit move instructions.
- 6-**Arithmetic:** 8 bit and 16 bit add, subtract, increment and decrement.
- 7-**Logical Operations:** AND, OR, XOR, and compare instructions.
- 8-**Exchange, Block Transfers, and Search:** One of the prime advantages of the Z80 over the 8080 is the ability of the Z80 to move chunks of data with a single instruction.
- 9-**Rotate and Shift:** The Z80 has more variations on this theme than the 8080.
- 10-**Bit Operations:** Another whole new area of control. The Z80 can test, set, and reset single bits within its registers and in memory.
- 11-**Program Control:** Jumps, relative jumps, calls and returns. Just wait

until you hear about DJNZ, the 'decrement, jump if not zero' instruction.

12-I/O Operations: Basic input and output, plus block I/O transfers and variable I/O ports.

13-CPU Control and Oddball Operations: Miscellaneous instructions that do not fall into any of the above categories.

14-Assembling Z80 Code: How to assemble Z80 code using CP/M-80 software. Includes brief discussion of public domain software that aids Z80 assembly.

15-Pitfalls and Debugging: One of the pitfalls of programming with the Z80 is trying to debug code that the host software (CP/M-80) cannot understand. (DDT gets very upset

when it tries to disassemble non 8080, i.e. Z80, opcodes.) I'll talk about software that overcomes this problem.

16-Subroutines: Useful subroutines that use some of the special features of the Z80. Block moves, searching memory for a specific byte or string of bytes and block I/O moves for the fast transfer of data to peripherals.

17-Interfacing to CP/M-80: Bringing it all together and closing the open loops.

References

Most of the information presented in this article is based on my own experience with the Z80. Hardware infor-

mation comes from the Zilog Data-book. The assembler mnemonics are detailed in the 'Z80-Assembly Language Programming Manual' also by Zilog. I have been looking for a first class book that delves into some of the theory of programming the Z80, but have yet to find one worth its binding. Anyone out there have any suggestions?

If, in the course of this article, you run across items you'd like to know more about, or have comments about the tutorial, jot them down and send them to me care of *Lifelines/The Software Magazine*. If they are items of general interest, I will append them to this series. Personal questions or complaints will be answered if you include a stamped self-addressed envelope with your query. Next month I will continue with a definition of the terms and buzzwords that surround the Z80.

Software Notes

Pascal/Z with Plink-II

The end result of the Pascal/Z system is a .REL file that is meant to be compatible with the Microsoft format. Unfortunately, multiple records for external symbol fixups are created, making these .REL files non-compatible. Plink-II will print duplicate symbol warning messages or die with error #190 when these files are linked.

The problem occurs when the compiler outputs macro calls that equate internally generated symbols of the form "Lnnnn" with the names of the actual routines they represent.

Ithaca Intersystems suggests the following procedure to correct the problem. The symbol equates can be eliminated by changing the macros called by the compiler. First, the EXTD macro must be changed in the MAIN.SRC and EMAIN.SRC files. Currently, it appears as:

```
EXTD:    MACRO    INTN,EXTN
          EXT      EXTN
INTN:    EQU      EXTN
          ENDMAC
```

This should be changed to read:

```
EXTD:    MACRO    INTN,EXTN
          EXT      EXTN
INTN:    JMP      EXTN
          ENDMAC
```

Second, the EXTR macro in the E\$MAIN.SRC file must be changed. It currently appears as:

```
EXTR:    macro    intn,extn
extn:    equ      intn
          entry   extn
          endmac
```

It should be changed to read:

```
EXTR:    macro    intn,extn
extn:    JMP      intn
          entry   extn
          endmac
```

Programs compiled after these changes have been made should link correctly, although they will be a few bytes larger.

Two modules in the LIB library have the problem described above: EXPFCT and NATLOG. These should be re-assembled and used to update the library.

Notice

The June issue was placed into the mail on May 26th. If you had any problem with the timeliness of this issue, please call our Subscription Department at (212) 722-1700, or write to *Lifelines/The Software Magazine* Subscription Department, 1651 Third Ave., New York, N.Y. 10028. We expect to place this issue, dated July 1982, into the mail around June 23rd. We will print each month the date of the previous issue's mailing and would appreciate your help in tracking the deliveries.

Full Screen Program Editors For CP/M-80: VEDIT

Ward Christensen

VEDIT (that's "VED-IT" not "VEE-EDIT"), is from Compu-View Products, Inc. They are a company of ambitious though naive people out to make a name for themselves by attempting to provide a quality full screen editor product for a variety of systems, including 8080 and Z80 (with both memory-mapped and CRT versions), as well as versions for CP/M-86, and the IBM PC-DOS.

Evaluation

DOCUMENTATION: The manual is 67 pages long. I tested Version 1.34, which has an additional 35 or so pages, primarily containing customization worksheets, changes which had not been incorporated into the main manual, etc. The documentation is quite complete, serving both the novice and the experienced user and including a nice one-page command summary.

SPEED: VEDIT is the slowest editor I have tried and is the only one I have seen which doesn't implement "preemptive scrolling". This means that the editor completes updating the screen after each scrolling key press, before looking at what additional scrolling keys you might have pressed.

For example, if you are repeatedly scrolling one line or one screen backwards on a terminal without line insert/delete, it completely redraws the screen over and over. My first reaction was "What in the world is it doing?" You may think you have time to go get a cup of coffee. To contrast: WordMaster, PMATE, and MINCE all preempt, thus interrupting the first scroll after having computed that it is futile, and very quickly showing you the final screen.

However, if you scroll the screen and then resist executing a scroll command until it stops (as a beginning user would do), you won't even notice this otherwise abominable characteristic.

VEDIT is very slow at executing commands. I edited a very large file, (my MAST.CAT), and deleted all "ASM" files from it. The file had 2600 lines, and there were 520 such lines. For this informal benchmark, I typed the edit command (e.g. VEDIT MAST.CAT) at the CP/M A↑ prompt. I started my stopwatch as I pressed return, and stopped the watch when the editing process was complete and I was back to the A↑ prompt. MINCE couldn't do this, as it lacks the suitable commands. WordMaster took about 1/2 minute, PMATE about 3/4 minute, and VEDIT just over two minutes.

ERGONOMICS: The default keyboard layout is "geometric". The control characters S, F, E, and C, are cursor left, right, up, and down. VEDIT customizability allows you to make the configuration "anything you want".

CONFIGURABILITY: There is a configuration program to set up VEDIT for one of many common terminals. Here is a list extracted from their configuration program (thus the abbreviations): ACT IV, Adds Reg 20, Adds Reg 30, ADM-3A, ANSI standard (VT100, Ann Arbor AMBASSADOR, etc.), Beehive (but not all of them), Datamedia, VT-52, Dyna 57, Hazel, H19, HP 2621, IBM 3101, Info 10, Int II-1, Int II-2, ISC 8063, Perk 550, VDB 8024, Soroc, Superbrain, CT82, Tele-ray, Televideo 920, 912, Televideo 950, 910, and Xerox 820.

The control sequences for these terminals are contained in an ASM file, as only DB's, so you don't have to know assembly programming. Strangely enough, it assembles not with the CP/M-80 assembler, but rather with the TDL assembler, although the differences are minor ("." instead of "\$" for current location counter, and ".END" instead of just "END"). In a phone call, I suggested it would be nice if it worked with the CP/M-80 assembler, they said "that's not a bad idea." Sigh.

For keyboard customization, VEDIT supplies the following functions which you may customize. These are taken right from the configuration program. I have added explanations for those that are not obvious: home (top of buffer), zend (bottom of buffer), cursor up, cursor down, cursor right, cursor left, back tab, tab cursor, zip (cursor to end of line), next line (cursor to start of next line), previous word, next word, previous paragraph, next paragraph, page up, page down, backspace, delete, erase to end of line, erase line, del previous word, del next word, undo, tab character, set insert mode, reset insert mode, switch insert mode, indent, undent, copy to text register, move to text register, insert text register, print text block, format paragraph, visual escape, visual exit, and restart editor.

The customizing nicely supports up to two lead-in escape sequences. I use ESC for one, and control-] for the other. This easily expands the allowable "control" keys.

The first time you do the customization, you use a "SET" file supplied. From then on, you may use your customized editor, making minor changes to it. For example, if you have made a mistake in one control key, you can change just that one. Note that this a lengthy process: you have to press return for every control function you want to leave as it was.

EASE OF LEARNING: The video-related commands are pretty straightforward. If you have customized them to your keyboard, you will nearly have completed the learning process just in setting them up.

The VEDIT command-mode is an extension of the CP/M-80 ED commands, which is convenient. There are a few minor differences ("Z" instead of "B", etc). The obscure ED "J"

command is not included, and the "V" command goes into visual mode, rather than presenting you with a line-numbered file. In general, it is easy to upgrade your thoughts from ED to VEDIT. The biggest difference is that you press two ESC characters to execute the command, rather than pressing return. Matter of fact, carriage return is an allowable character, so you don't have to "hokey it up" with ↑N as you do in ED or WordMaster. That's nice.

Objective Evaluation

A. Video Related Criteria

FULL SCREEN: VEDIT has two distinct modes: a full screen mode, and a command mode. In full screen mode, there are keys to move the cursor one line up or down; ahead or back one character, word, or paragraph; to go to the end of the line, to the beginning of the next line, or to the top or bottom of the buffer. Very large screens are also supported - up to 70 lines, via a special configuration file.

SCROLLING: There are keys to scroll up or down one line, or a screen full. As I already noted under "SPEED", scrolling does not preempt itself, which is really unbearable for me. This review is based, perhaps unfairly, upon only a very short time working with VEDIT, but every time I started working with it, the lack of preempting caused me to quit before this shortcoming drove me bananas.

A good feature of VEDIT is its support of terminal hardware line-insert and line-delete, making small scroll amounts very nice on the proper supporting hardware.

Forward file scrolling is transparent to the user. Backward scrolling, once the front of the file has been forced out to disk, is *not*. This was surprising, since WordMaster "showed the way" years ago. (The VEDIT author designed and implemented VEDIT without having ever tried WordMaster!) Thus VEDIT was started as an ED enhancement rather than an improvement on WordMaster. Don't get me wrong, they implemented some useful features WordMaster doesn't have, but missed some of its simple usability features. This is why I used the word "naive" when referring to the author company in my opening paragraph.

INSERT: VEDIT has an insert mode.

OVERTYPE: VEDIT has an overtyping mode.

UNDO-KEY: If you screw up a line, VEDIT has automatically saved it for you, and puts the original back upon your command. This feature is not as general as the PMATE undo, but still handles many mistakes. It doesn't let recovering from a total line delete, but does handle a mistaken erase to end of line.

REPEAT: *Nope!* (Sigh). Again, WordMaster showed the way, but CompuView Products weren't looking, I guess.

TEXT EDITING ABILITIES: VEDIT can handle word-wrap, so it is suitable for basic text editing. "EP 7 n" sets the right margin to "n". The left margin is set by an indent amount. "EP 7 0" turns off word wrap. Also, you may configure a key as "format paragraph".

VEDIT also has the ability to print, either a specific number of lines, or the entire buffer. It does not explicitly handle paging, etc.

B. Command-Related Criteria

MOVE: You may move to the top or bottom of the file (via "B" and "Z"), ahead or back by character (via "C" or "-C") or line (via "L" or "-L"). Like ED, "OL" moves to the front of the current line. A positive or negative number may precede these commands.

DELETE: VEDIT may delete characters (via "D") or kill lines (via "K"). A positive or negative number may precede the command, indicating the direction and number of characters or lines to delete.

INSERT: Arbitrary character strings may be inserted. The "T" command inserts the character string, following it up to a terminating ESC (which echoes as a "\$"). To insert a single character, Ichar\$ may be used, or, the "nEI" command. The latter allows inserting control characters.

TYPE: "nT" types "n" lines. Works just like ED.

CHANGE: As in ED, the S command changes one string to another.

MOVE and COPY: These are implemented very well: one control key marks either end of a block to be moved, another a block to be copied. You may mark either end first. A third key takes the marked block and inserts it.

In command mode, "nP" puts "n" lines into a special text buffer. The G command gets them back. There is no ability to append to the text buffer like WordMaster, PMATE, or MINCE can.

COMMAND STRINGS: These are similar to those implemented by ED. You may choose the characters you want to use for enclosing repeated commands, i.e., "[" and "]", or "<" and ">", etc. You must explicitly code a number of repeats, unlike ED, WordMaster and PMATE; those editors, if you code a repeated command, nicely default to "all occurrences".

C. File-Related Criteria

BACKUP: VEDIT works the same way as ED and WordMaster do. After a file is edited, the original is renamed to ".BAK", and a temporary file created by the editor is renamed to the original name of the file.

You may also separately specify the name of the file being read, and the file being written. In this case, if the file being written already exists, it is called a .BAK file first. This is a handy feature.

SAVE: The EA command is like the "H" command in ED or WordMaster; it finishes writing the edited file to the work file, then renames the original file to .BAK, and renames the temporary file to the original filename. It then starts editing that

(continued next page)

file again. It works the same way as leaving VEDIT, then re-editing the same file.

QUIT: EQ quits VEDIT. There is no "quit but stay in VEDIT", as the "O" command of ED or WordMaster permits.

READ: To insert a file from disk into the one you are editing, you may use "EGfilename[starting line, ending line]". Very nice. It doesn't default to "entire file" if the line range is omitted.

WRITE: "EWfilename" opens "filename" for writing, creating a .BAK if appropriate. "nW" then writes "n" lines.

DIRECTORY: No.

OTHER: VEDIT has a few features that don't fit the above categories. It can log in a new disk (in case the disk you are working on fills, and you want to change disks). It can erase files on disk.

Statistics

VEDIT sells for \$145, available directly from CompuView, or from various software houses and takes 10K of memory.

Room For Improvement

(1) VEDIT should preempt the display when you have pressed another scrolling key.

(2) It could be easier to customize. I'd like to see a file (either an ASM file, or a data file). Then, if you don't like something, you can easily edit it and change it. Also, if you receive an updated version of VEDIT, you wouldn't have to do the customization from scratch.

(3) VEDIT lacks bidirectional file scrolling, which it should have. If you edit only small files, this won't bother you.

(4) A repeat key is badly needed - at least a simple one like WordMaster that allows 4, 16, 64 times, or better yet, an elegant one like MINCE has.

(5) The "scroll down" key, if pressed repeatedly, does take you to the last screen of the file, but *not* to the bottom of the file. I didn't like having to press all those "line scroll down" keys after using "page scroll down", just to get past the bottom of the file to insert more material.

(6) Line-insert: ask yourself why you would want a "line insert" command. I'm sure your answer will be "to type some new lines into". At least, that's how I use them. Well, VEDIT's line-insert puts the inserted line *above* the cursor line, so typing ten line-inserts leaves you having to do ten cursor-ups. Did CompuView use ED to write VEDIT?

(7) Line-insert re-draws the entire screen rather than using hardware line-insert, even when configured for it.

(8) The first character of each line is not even used. I guess they took the easy way out (or at least, the "strange way out")

and left it for a "long line continuation character" instead of simply adding it to the beginning of continued lines.

(9) Nit-picking: For repeating commands, you shouldn't have to specify "#" to mean "repeat forever". [command] should repeat forever, as it does in WordMaster or PMATE. Even "m..." in ED repeats - you don't have to say "#m...".

(10) More nit-picking: EGfilename[start line,end line] should default to the whole file, instead of forcing the user to key in EGfilename[1,9999].

I'm tired of complaining. I think I'll go on to more serious matters.

Bugs

Flat "bugs" are pretty rare in most commercial software these days. However, I uncovered several in VEDIT after a few hours use:

(1) Holding down "delete line" at the top of a small file, eventually makes VEDIT "go away", instead of simply clearing your buffer. This is one of several oddities I observed using VEDIT with a CRT having auto-repeat keys.

(2) When presented with a file containing "high bits" (à la WordStar), it, well, aah, umm, barfed! The screen was a hodge podge of reverse video splotches, with my file strewn about. Character delete deleted the wrong characters, etc.

(3) Very long lines give it conniption fits. (That, whether or not it is spelled right, is Wisconsin-ese for "doesn't work".) Note that this *is* a documented shortcoming. However, ED, WordMaster, PMATE, and MINCE all handle very long lines admirably.

Conclusions

The CompuView folks have implemented an editor with many much-needed features, such as the UNDO key and total keyboard customization, and provide versions for a wide variety of systems. Most VEDIT users I have talked to are happy with it. (I can only assume they never saw WordMaster or PMATE or MINCE or...) But I feel that VEDIT has a long way to go.

Next, that 35-buck editor. It is called PRECISE, and is patterned after the old (but excellent) Processor Technology full screen editor. In the queue are Micro-Emacs, Mr EDit, a packaged version of the full screen editor written in "C" from Dr. Dobbs' Journal, and others. I do not at this time plan to review any "Z80 only" editors, as I prefer software which does not limit itself to a subset of CP/M-80 machines.

Volume 82, Catalogue and Abstracts

CP/M Users Group

Volume 82

DESCRIPTION: North Star BIOS routines by Steve Bogolub.

NUMBER	SIZE	NAME	COMMENTS
	3K	CATALOG.082 ABSTRACT.082	Contents of CPMUG Vol. 82 Volume 82 Abstract
82.1	53K	MBIOS56.ASM	BIOS with PMMI MODEM drivers
82.2	55K	NBIOS56.ASM	North Star BIOS
82.3	16K	NSBIOS.DOC	DOC on above
82.4	35K	NSCOPY.ASM	Disk format and copy program
82.5	12K	NSGEN.ASM	SYSGEN program
82.6	21K	SDNBIOS.ASM	Single density N* BIOS
82.7	2K	SDNBOOT.ASM	Single density N* BOOT
82.8	11K	SDNSGEN.ASM	Single density N* SYSGEN
82.9	8K	SDNUSER.ASM	Single density N* user area
	5K	U-G-FORM.LIB	CPMUG contribution form
	2K	CRCK.COM	CRC check program.
	1K	FILES.CRC	CRC of files on disk

Abstracts

This volume consists entirely of Steve Bogolub's North Star BIOS routines. They are copyrighted, but available freely for non-commercial use.

NBIOS56.ASM

This is a BIOS for North Star MDS-AD2 double density 5.25" floppy disk units 1 and 2 as CP/M-80 drives A: AND B: and a Jade Double D Rev C 8" floppy disk units 0 and 1 as CP/M-80 drives C: and D:.

NSCOPY.ASM

This is a utility to allow the operator to

copy, format, or validate North Star double density disks. Copying can be done between any two North Star drives, and can include all tracks, or just the system (0 and 1) tracks or data tracks (2 - n).

NSGEN.ASM

Provides the functions of the Digital Research standard CP/M-80 program 'SYSGEN', but for a North Star double density version of CP/M-80.

MBIOS56.ASM

This BIOS is specialized for remote modem dialup use. It incorporates elements of Dave Jaffe's BYE program. If the memory location "MODEM"

contains a value of zero when the cold boot routine "init" is entered, a special set of console vectors is patched over the genned-in set to handle the PMMI modem instead of the normal horizon drivers. Variable baud rate, and other features, are supported.

SDNBIOS.ASM

BIOS containing routines to support North Star MDC-A4 single density 5.25" floppy disk units 1 and 2 as CP/M-80 drives A: and B:. Console and list routines are located in a separate module loaded by the bootstrap routine into high memory at address USER. That value is assumed to be on a 1K boundary, and the user routines are restricted to 2 pages. The BIOS is so large, MOVCPM should be run for two less than the desired system size.

SDNSGEN.ASM

This program provides the functions of the Digital Research standard CP/M-80 program 'SYSGEN', but for a North Star single density version of CP/M-80.

SDNUSER.ASM

These routines provide console and list functions for CP/M-80 2.2, using the hardware of the Martin Research Machine. The routines are loaded into the non-contiguous 1K of high ram by the bootstrap routine.

SDNBOOT.ASM

This routine is read off the disk from track zero, sector four by the North Star single density BOOT PROM. It is responsible for reading in the BIOS and USER routines from the disk. It must then call the user init routine, then jump to the BIOS cold boot entry point.

Ward Christensen

Ada: Who Is She?

Steve Patchen

Ada is a new programming language which has been under development since the mid 70's. It is sponsored by the Department of Defense and is formally described by MIL-STD-1815. I have included a list of books and articles about Ada for those readers who wish to learn more details. In this article I am going to concentrate upon the role of Ada as a fourth generation language and on its participation in the system development environment. I should point out, however, that the intended purpose of Ada is to enhance development of what are called embedded systems. Embedded systems are those which control the entire computer environment. An embedded system would be, in a business environment, both the operating system and the application (designed for applications such as missiles require, no doubt). This approach is not necessarily contrary to use in business systems, because integration of the operating system and the application is desirable for operation integrity.

Two compilers for subsets of Ada have recently been introduced for the CP/M-80 operating system. They are Supersoft's Ada and RR Software's Janus. *Lifelines/The Software Magazine* will be covering these two versions in more detail later. For the moment I am only concerned with the current comparative extent of their subsets. The manuals for both compilers show an intention to add more Ada features. The Supersoft Ada implements less than 50% of Ada and Janus implements a little more than 50%. Janus also provides a few non-standard additional features. Neither compiler has the separate compilation feature yet.

Ada is a very ambitious language. That a reliable compiler is possible for its full definition has not yet been proven, even on mainframes. The more complex a language is, the more difficult it is to write a reliable compiler for it. The

alternative to a complex language which has everything is a simple one which contains only those features needed by every application. This type of language then requires versions with extensions for many applications. BASIC and Pascal are examples of this second type of language. Unfortunately, when the method for extension is not provided in the original definition of the language, every compiler writer tends to go his own way when providing the needed extensions. Thus there are more than 200 versions of BASIC and most of them are not compatible. Pascal is on its way to a similar fate. This problem affects the portability of applications written for these languages. Earlier languages like FORTRAN maintain popularity because they are defined by a standard and programs enjoy good portability across machine environments.

Ada, on the other hand, seems to be following in the footsteps of IBM's PL/I. PL/I exemplified a similar grandiose scheme to design a language for all purposes. It too has the problems of complexity which have led to the introduction of several subset languages over the years. (Digital Research's PL/I-80 is an implementation of the G subset for the language.) The Department of Defense has so far resisted the inclusion of the definition of any subsets of Ada in its specification. We will no doubt be seeing compilers emerge with a random sampling of subsets of Ada. Many will not bear the name Ada because DOD controls the use of this name. In many ways Ada suffers from the design goal of trying to include everything on everybody's wish list.

There are conflicting requirements in the design of programming languages. The language must be complex enough to do complex jobs in real life. Yet it must be simple enough for ease of learning and use. Simplicity usually depends on readability and a formal

description in terms of the language's syntax and its semantics. That is, programs should be understandable by a programmer other than the original author. Also, the program created in the language should be portable to other machines which have a compiler for it. For lack of unnecessary complexity and improvement of reliability and readability, Pascal takes the best approach. However, the only way of avoiding the problem of infinite versions and extensions is to provide the means of extending the language within the language itself.

The goal of extensible languages is the improvement of the efficiency of programming and the clarity of the programmer's products by facilities which mold the language to the application's requirements. Such languages consist of a base language with a complete but minimum set of primitive facilities and extension mechanisms; these allow the definition of new language features in terms of the language primitives. There are two types of extension mechanisms: semantic extensions and syntactic extensions. Semantic extensions introduce new objects to the language (such as data types or operations) and syntactic extensions create new notations for existing or user-defined mechanisms. The subroutine is a common form of semantic extension. The record data type and user-defined data types in Pascal and Ada are semantic extensions. Syntactic extensions tend to slow down compilation considerably and, as a result, none of the popular languages have the ability to extend their syntax. If syntax extension were defined in a language like Pascal the extensions could be carried to different machine environments with the programs, permitting portability of the programs. Until someone invents a way to write an efficient extensible compiler we will have to continue to struggle with these design trade-offs. Why couldn't a compiler compile its extensions first and use the result to com-

pile program units? Languages like FORTH are capable of syntactic extension but implementations of a FORTH compiler for a more popular language run very slowly. Compiler writing languages have been proposed before (see McKeeman).

In the realm of microcomputers there already exist a great many software development tools. Many of these tools complement each other and some were even designed to work together. Digital Research has a tool set for its PL/I-80 compiler which includes a linking loader and relocatable librarian, and an independent debugger and editor. Microsoft's MBASIC has a complementary M80 compiler and similar library facilities. In addition, database management systems like MDBS and MICROSEED are implemented as relocatable libraries of routines to be linked with applications developed in Microsoft or Digital Research environments using existing linking loaders.

Ada attempts to integrate some of these functions into the language definition. It introduces the concept of the PACKAGE. The PACKAGE is a unit of code and/or data which can be compiled separately and automatically linked to later compilations which reference the package when they are compiled. This integrated approach allows data type checking to be performed by the compiler of references made, against the actual package data types.

However, Ada is still not a complete system development environment. Debugging facilities tend to be machine-dependent and are thus left up to the implementor to define. The implementor might or might not choose to include debugging features in the compiler. But there is a submerged facility called EXCEPTIONS which lets the programmer control error handling.

Although the data structures defined in Ada are extensive, they are still primitive and do not include any physical database management level definitions. Database management still requires comprehensive extension of objects of the language. Project documentation management and program editing are also independent of the language definition. The PACKAGE separate compilation technique implies the need for a

sophisticated editor possessing knowledge of the library in source and compiled forms. The order of compilation of PACKAGES might be hard to chart during program modification without some sort of hierarchical management of the complete project environment.

Last month, while developing some criteria to place software development products into perspective, I discussed some structures which are characteristic of VHLL's and fourth generation languages. Ada has many examples of these structures. The concept of the RECORD was introduced by Pascal and is used by Ada. Records are groups of possibly dissimilar data types, even other records. This structure allows the construction of hierarchies and networks of data structures within a system. This does make it convenient to define data structures of a high level of abstraction. Such structures can be allocated to a package which is used like a COMMON or GLOBAL data structure (i.e., manipulated by high level routines). Ada's ability to overload functions and routines with the same name but with different data types for parameters provides abstraction; it permits similar functions, involving different objects, to read the same way to the programmer. These functions and data structures take on the characteristics of aggregate operators. Indeed, within the limits of the previously-defined operators for the language, other operations can be made to read with the same syntax; these operators can be overloaded with the new functional definitions. There is even an aggregate syntax for dealing with instances of records and array type contents. In addition, the GENERIC feature allows macro-like substitution; it permits parametric substitution into routine and function masks, so variations of commonly-used structures can be created. This feature could make application building techniques convenient to implement.

Ada also has implicit control structures to ease the programmer's burden for error handling. It has built-in functions which return the attributes of data types and it includes facilities to perform parallel processing and handle the *rendezvous* among separately running TASKS. Machine-dependent control structures can be implemented by the declaration of PRAGMAS. These are

like compiler switches. Ada has pointer capability, employing access data types which can be used to implement associative referencing. There are no directly usable associative reference features more complicated than nested data structure label concatenation.

Although Ada is not a complete development environment, it does provide many structures which make it convenient to abstract applications to a highly logical level. It is possible to extend the objects used in an Ada program to a level which reflects the objects of an application being developed. The syntax for the specification of applications must be implemented by a separate manual or automated system.

Ledgard, Henry, Ada: An Introduction, Springer-Verlag

Pyle, I.C., The Ada Programming Language, Prentice-Hall, Englewood Cliffs, NJ, 1981

Hibbard, P., A. Higgins, J. Rosenberg, M. Shaw and M. Sherman, Studies in Ada Style, Springer-Verlag, 1981

Wegner, Peter, Programming with Ada: An Introduction by Means of Graduated Examples, Prentice-Hall, Englewood Cliffs NJ, 1980

Skelly, P.G. for the ACM Standards Committee, "The ACM Position on Standardization of the Ada Language", Comm ACM 25.2 (Feb 1982) 118-120

Ledgard, H.F. and Andrew Singer "Scaling Down Ada(Or towards a Standard Ada)", Comm ACM 25.2 (Feb 1982) 121-125

Hoare, C.A.R., The Emperor's Old Clothes(ACM Turing Award Lecture) Comm ACM 24.2 (FEB 1981) 75-83

Ada Programming Language. Department of Defense Military Standard MIL-STD-1815. 1980 (available from Naval Publications and Forms Center, 5801 Talbot Ave., Philadelphia, PA) (also included in Supersoft's Ada Manual)

McKeeman, W.M., J.J. Horning, P.B. Wortman, A Compiler Generator, Prentice-Hall Inc., Englewood Cliffs, NJ

An Overview of MicroPlan

Raymond J. Sonoff

This article is intended to highlight some of the key features associated with MicroPlan, a financial planning package from Chang Laboratories, Cupertino, California. Note that this overview is highly limited, since only a demo diskette was supplied by Chang Laboratories in response to a request for a copy of MicroPlan. (While this article was in process, I was contacted by Terry Cheng of Chang Laboratories. She told me that had a nondisclosure agreement been signed a full package might have been supplied; however, *Lifelines/The Software Magazine* was not informed of this problem until too late. However, should you desire a more detailed review or applications examples of financial planning software packages [such as MicroPlan] in future issues of *Lifelines/The Software Magazine*, please phone or write the editorial department to register your particular areas of interest.)

Because utilities involving SAVE and PRINT functions had been disabled on the MicroPlan DEMO diskette, I could not perform any program storing or hardcopy printout. Therefore, actual exercising of MicroPlan's features or computational commands was limited in both nature and extent.

Material Received

The DEMO Package of MicroPlan (Version 3.02) came with an 8" diskette (IBM single-sided single density CP/M-80 compatible) inside a 3-ring binder having a manual of approximately 100 pages divided into nine sections, two appendices, and an index. The words "DEMO COPY not for resale" had been rubber stamped in ten locations throughout the manual as a reminder that I had but a limited edition for review.

Installation

Installation of MicroPlan was clearly described in Appendix A of the manual. Several prerequisite conditions or suggested system features were also pointed out. These included:

1. Requirement for a CP/M-80 (or CP/M-80 compatible) operating system, Version 2.0 or later.
2. At least one floppy disk drive (but two would be preferred) of either 5.25" or 8" size; about 140 Kbytes of storage capacity is required just to handle MicroPlan's system files.
3. At least 48 Kbytes of RAM (64 Kbytes is preferred). Note: With MP/M-80, MicroPlan can only be used in 48 Kbyte user banks.
4. A CRT display with at least cursor-addressing and clear screen features is needed, and
5. A printer is needed for report generation activities. (With a DEMO copy of MicroPlan, printout cannot be implemented anyway.)

Although fifteen terminals were listed in the terminal menu, I had to use "OTHER" for my Heath (Zenith) H19 CRT terminal. (This selection did prove sufficient to get MicroPlan up and running immediately. However, no highlighting of rows or columns, stated in the manual to be a standard feature, was achieved.)

Documentation

The manual was arranged as follows:

INTRODUCTION:	Sec. 1: Getting Started
WORKING WITH TABLES:	Sec. 2: Simple Usage
	Sec. 3: Report Generator
	Sec. 4: Advanced Usage
WORKING WITH PROGRAMS:	Sec. 5: Using Models
	Sec. 6: Elementary Programming
	Sec. 7: Programming Techniques
APPLICATIONS:	Sec. 8: Doing Financial Computations
	Sec. 9: Forecasting Cash-flows & Balances

Appendix A: Installation
Appendix B: Command Reference
Index

Commands: Heart of MicroPlan

When you first initiate MicroPlan, a MAIN MENU of available commands will always appear on the right hand side of the CRT screen. Because MicroPlan commands are represented as numbers ranging from 1 to more than 150, you simply select the desired command by keying in the appropriate number, and MicroPlan will execute your intended command.

Command groups. Commands are organized into groups under the following labels: FORMAT; DATA ENTRY; MATHEMATICAL; FINANCE; PRINT; STATUS; and UTILITY.

A HELP command (7) is included in the MAIN MENU. When the "ENTER COMMAND:" prompt statement appears on the CRT screen, the HELP command can be called by entering 7 < CR > . A three-line summary of the features of any subsequently specified command of MicroPlan will appear on the CRT screen once you enter a command number followed by a carriage return in response to the prompt statement "WHICH COMMAND (1-150):" Referral to the MicroPlan manual can be minimized as a consequence of the HELP command feature.

Data Entry and movement commands. A given table can occupy up to 500 rows and 99 columns. MicroPlan displays 17 rows and 5 columns on the screen at any one time. A series of ESC key sequences (followed by either a 2, 8, 6, 4, or 5) allow you to move about a given region within a table. Essentially, up or down shifts of rows in increments of ten, movement left or right of columns in increments of four, or returning to row 1 column 1 are the five basic ESC key commands. These ESCape key sequences can be entered at any time during the use of MicroPlan. Terminals having cursor keys reduce the operations to single keystrokes. Of course, for larger tables you would likely employ the GOTO (36) command to position the screen at a specific portion of the table by specifying the desired row and column (i.e., DATA POINTER) upon observing the respective prompt on the screen.

Data entry can be for individual "cells" (any specific row, column "coordinate"). If desired, a given entry can be extended (simply by a 0, 1, 2, or 3 number selection) to be either a zero value, constant value, growth rate, or an amount to be added to each successive (by row or by column) cell. This extension can be limited to a user-specified range if desired.

Math Commands. Mathematical computations can be between any two rows or columns, or they can involve any given row or column and a constant. A SUM command is also available that will compute the sum of values for a range of rows or columns. Besides ADD, SUBTRACT, MULTIPLY, DIVIDE, and these four operations coupled with a constant, the remaining math commands include NEGATE, INVERSE, INTEGER, ROUND, CUMULATE, ABSOLUTE, GET, FLOOR, and CEILING, as described in the manual. These commands work with one row or column of the table, and values are computed and stored in the current (data pointer) row or column.

Among other data entry commands that are incorporated in MicroPlan are FORMULA, PLUG, FIX, and NULLIFY. These commands allow highly useful, unique operations to be achieved easily. Stated briefly, FORMULA permits you to create your own relationship to apply to any row or column, values, or positive constants in the formula. PLUG provides a means to apply a formula to calculate results for a particular cell, and the formula may reference table values (i.e., contents of row/column "cells") or constants. NULLIFY cancels any command associated with a specified row or column, without affecting the data in the table.

Finance Commands. There are 17 FINANCE commands, including calculations involving depreciation, internal rate of return, ratios, percentages, growth and tax schedules.

PRINT commands. Six PRINT commands appear to provide for nearly any possible option, including interfacing with word processors or spooling of output to a printer, setting global report options, and setting CRT display options.

STATUS commands. Ten STATUS commands allow for RANGE selections for rows and columns, as well as for MODE selections. MODE selections include RUN PGM to start execution of current program, PROGRAM, COMPUTE, DELETE, and ORDER (to set the computing order for your table:ROW/ONLY; COL/ONLY; ROW/COL or COL/ROW).

UTILITY commands. The UTILITY commands include load, save, list, print, erase, clear and reset tables, as well as redisplay screen.

Added note: The command list is certainly not static. Without doubt, additional commands for each of the above categories will emerge as revisions and add-on modules are made available by Chang Laboratories.

User Evaluation

Manual. Documentation is excellent. User-entered keystrokes are shown in boldface type, and numerous examples are provided throughout each section of the manual. Most examples include screen images of what should be observed as you implement the set of commands for a particular example. Particularly noteworthy is the fact that many statements throughout the manual teach you about MicroPlan's operations, assumptions, distinctions among similar commands; the applications information provided reflects a desire to meet the needs of end users. Among the topics that are concisely interleaved among the examples are the following: what is done with the entries you make, what general methods are used to perform various calculations, how long MicroPlan can be expected to take before displaying results when certain commands are incorporated into a program, subtleties between similar commands (Example from Section 9: "GROW differs from the grow option in the ENTER command in that the GROW command uses growth rates that are stored in a row or column of your table and updates the forecast whenever you use the COMPUTE command."), and what command sequences you can use to achieve specific planning evaluations.

Tryout. Section 1 introduces essential features of MicroPlan, using a five-year projection as a walk-through example. The example proved easy to implement, follow, and understand. And, by the time Section 6 has been completed, you will understand how to save a program (defined as a series of MicroPlan commands), retrieve programs, do numerous what-if analyses (data field computation results can be erased with but a single "CLR DATA verify (113)" command, for example), redimension tables to a different number of rows and columns, print a description of your model which will include a listing of row titles, options, table commands and global options, and so on.

Other features. Programs of mathematical, financial, or other types of planning and analysis can be created to suit your own personal or business requirements; tables can be merged, and reports can be formatted or edited easily prior to printing (using conventional CP/M-80 compatible word processing packages to perform such merging of MicroPlan-generated tables or complete programs).

Summary

MicroPlan represents an excellent software product that can be employed by end users with little doubt that they will get their money's worth. As with any program, there will be cumbersome operations, as well as the ever-present requirement to carefully check all computations to minimize the likelihood of user-generated errors (e.g. by misapplied commands, formulas, etc.).

8080 Assembler Programming Tutorial: Subroutines, Part 2

Ward Christensen

I'll open this month with a MOVE routine which will work fast on either 8080 or Z80, because it detects which processor it is running under.

MOVE Subroutine

The secret to this subroutine's success is that it tests whether it is running under 8080 or Z80. How? Well, the 8080 bit in the PSW called "parity" is used by the Z80 for two purposes: parity in logical instructions, and overflow in arithmetic instructions. Suppose the accumulator has a 2 in it. If this were incremented to a 3 with an INR A, an 8080 would set the even parity flag (3 in binary being 0000 0011, i.e. an even number of bits are on). A Z80 would think of the bit as the overflow bit, and since incrementing 2 to 3 doesn't overflow, would set the bit off. Thus "JPE" (jump if parity even) will JMP if the processor is an 8080. Here is the subroutine.

```

;
;System-independent MOVE subroutine. Does Z-80 LDIR
;if a Z-80 processor is being used.
;
;(HL) = source (from) address
;(DE) = destination (to) address
;(BC) = # bytes to be moved
;
MOVE   MVI    A,2      ;a=2
       INR    A        ;a=3, parity even if 8080
       JPE   MV8080   ;running on 8080
       DB    0EDH,0B0H ;Z-80 ldir
       RET
;
;8080 move routine using same register
;conventions as Z-80 LDIR
;
MV8080 MOV    A,B      ;check if
       ORA    C        ;      bc = 0
       RZ     ;      yes, return
       MOV    A,M      ;get byte(HL)
       STAX  D        ;store byte(DE)
       INX   H        ;bump (from) addr
       INX   D        ;bump (to) addr
       DCX   B        ;decrement count
       JMP   MV8080

```

If you are running on an 8080 only, the "MV8080" routine will suffice. However, the routine "MOVE" with its subroutine "MV8080", will work faster if you should switch to, or otherwise run on, a Z80.

Character I/O and Standard Equates

It's time to get into CP/M-80 related Input/Output. CP/M-80 I/O is usually executed by a call to BDOS, CP/M-80's Basic Disk Operating System. Let me introduce a set of standard equates for the BDOS call functions. There is little point in memorizing the actual numeric values for the functions, since all of the assemblers support an EQU capability to save you the trouble. If you *do* want to memorize a few, functions 2 and 9 are the only ones I found to be worth memorizing.

The functions work as follows: place the function code in register C, and if the function requires a parameter (address,

character, etc.) to be passed, place it in DE if it is 16 bits long, or just in E if it's 8 bits long.

I don't believe I have ever used all of these equates, but prefer to cover them all. Note the familiar format, of label, <tab> operand (EQU in this case) <tab> then the operand, then if any comments, a <tab> or two, then the ";" starting the comment:

```

;
RDCON  EQU    1      ;Console input to (A)
WRCON  EQU    2      ;Send char in (E) to console
RDRDR  EQU    3      ;Reader input to (A)
PUNPUN EQU    4      ;Punch output from (E)
LISTOUT EQU    5      ;Printer output from (E)
DIRCON  EQU    6      ;Direct console I/O (CP/M 2.2)
;      more about this later.
GETIOBY EQU    7      ;Get the I/O byte value (more later)
PUTIOBY EQU    8      ;Put the I/O byte (more later)
PRINT  EQU    9      ;Print string pointed to by (DE)
;      end the string with '$'
RDCONB EQU   10      ;read console line into buffer (DE)
CONST  EQU   11      ;Get console status
GETVERS EQU   12      ;Get CP/M version number
RESETDK EQU   13      ;Reset the disk (allow changing disks)
LOGIN  EQU   14      ;Log-in a new disk
OPEN   EQU   15      ;Open a file, FCB in (DE)
CLOSE  EQU   16      ;Close a file, FCB in (DE)
SRCHF  EQU   17      ;Search for first file, FCB in (DE)
SRCHN  EQU   18      ;Search for next file, FCB in (DE)
ERASE  EQU   19      ;Erase a file, FCB in (DE)
READ   EQU   20      ;Read a file sector, FCB in (DE)
WRITE  EQU   21      ;Write a file sector, FCB in (DE)
MAKE   EQU   22      ;Make a new file, FCB in (DE)
REN    EQU   23      ;Rename a file
GETLGIN EQU   24      ;Get vector of logged in disks
INQDISK EQU   25      ;Ask what current disk is
SETDMA  EQU   26      ;Set the DMA (Disk I/O) addr. (DE)
INQALC  EQU   27      ;Point to the CP/M disk allocation map
WRPROT  EQU   28      ;Find which disks are write protected
GETROV  EQU   29      ;Find which disks are read only
SETATTR EQU   30      ;Set the attributes of a file
GETPARM EQU   31      ;Get the disk parameter block
SGUSER  EQU   32      ;Set or get the user number
RDRAND  EQU   33      ;Read a random file record
WRRAND  EQU   34      ;Read a random file record
COMPSZ  EQU   35      ;Compute a file's size
SETRAND EQU   36      ;Position for random I/O
;
;Here are a few other equates which you will find useful:
;
BDOS   EQU    5      ;entry point to BDOS
FCB    EQU   5CH     ;Main file control block address
FCB2   EQU   6CH     ;Second file control block address
FCBEXT EQU  FCB+12   ;Extent byte of first file
FCBRNO EQU  FCB+32   ;Record number in first file
CR     EQU   0DH     ;ASCII carriage return character
LF     EQU   0AH     ;ASCII line feed return character

```

The first character I/O subroutine will simply print a character. Here it is:

```

;
;print character in (A) via call to BDOS
;
PRINTA MOV    E,A      ;move char to (A)
       MVI    C,WRCON  ;get function
       CALL  BDOS     ;print char
       RET           ;      and return

```

That's a pretty trivial subroutine, but frequently used. Another, perhaps even more frequently-used subroutine

prints a string, employing BDOS function 9. The string must be terminated with a '\$':

```

;
;print signon message
;
PRSIGN LXI    D,SIGNON ;point to message
        MVI    C,PRINT ;get function 9
        CALL   BDOS   ;print it
        RET
;
SIGNON DB    'FOO.COM AS OF 05/23/82',0DH,0AH,'$'

```

Note how the string being printed is terminated with a '\$'. I wanted a carriage return and linefeed printed, so these had to be included in the string before the '\$'.

The CALL to BDOS will end somewhere inside of BDOS with a RET, and then will execute the RET I coded. Here would be an alternative:

```

:
JMP    BDOS ;print it
:

```

Thus, when BDOS issues the RET, it goes directly back to the caller of PRSIGN. In general, any time you code a CALL followed by a RET, you can delete the RET, and change the CALL to a JMP. This has one drawback however; when you're debugging, you cannot stop execution on the RET to see what's going on upon return from the subroutine.

Character Input

The next most common character I/O after character output is character input. You can input a single character via BDOS function 1, which I call RDCON:

```

MVI    C,RDCON
CALL   BDOS

```

Whatever character is typed will be echoed for you by BDOS.

If you want a bit fancier input, like that used when CP/M-80 is at an A> prompt, you can use the BDOS read string function, number 10:

```

LXI    D,INBUF
MVI    C,RDCONBF
CALL   BDOS

```

The INBUF area is specially set up. Its first byte contains the length of the buffer. After data has been keyed in, BDOS places the actual number of characters read into the second byte. The third byte contains the first character keyed. The carriage return typed when entering the data is *not* stored in the buffer. You must compute the location of the buffer end by using the length byte. I like to store a 00 there, as it is easy to test with an ORA A instruction:

```

;
;routine to store 00 at end of input buffer
;which was obtained by BDOS function 10.
;
FIXUP LXI    H,INBUF ;point to buffer
      INX    H       ;skip to length byte
      MOV    E,M     ;move it to e
      MVI    D,0     ;make DE = length
      DAD    D       ;add to HL
      INX    H       ;bump past last char
      MVI    M,0     ;store 00 there
      RET

```

Of course, the first instruction could have been "LXI
(continued next page)

ANNOUNCING THE FOX & GELLER dBASE II PROGRAM GENERATOR! QUICKCODE™

Now, without *any* programming, you
can create these in seconds:

- * DATA ENTRY PROGRAMS
- * DATA RETRIEVAL PROGRAMS
- * DATA EDIT/VALIDATION PROGRAMS
- * MENUS
- * dBASE FILES

INTRODUCING FOUR NEW DATA TYPES:

DATE • DOLLARS • TELEPHONE
• SOC. SEC. NO.

With QUICKCODE, you can **have** your program, but you don't have to **write** it. So, you can do things like knocking out an **entire accounting system** over the weekend! And QUICKCODE includes a powerful new version of our popular QUICKSCREEN™ screen builder, so you will put together screens and reports that'll dazzle even the most skeptical (you can even use Wordstar™ to set up your screen layouts).

YOU MUST SEE IT TO BELIEVE IT.

And is QUICKCODE EASY TO USE? You never saw **anything** so easy. You don't have to know how to program. You don't even have to answer a lot of questions, because there **aren't any!**

QUICKCODE \$295

ALSO FROM FOX & GELLER

QUICKSCREEN

Microsoft BASIC version	\$149
CBASIC version	149
dBASE-II version	149

dUTIL dBASE utility	75
----------------------------	----

**Fox & Geller Associates
P.O. Box 1053**

Teaneck, NJ 07666 (201) 837-0142

dBASE-II™ Ashton-Tate
Wordstar™ Micropro Int'l

H,INBUF+1", and the following INX H skipped.

Hex Input

Let's do something practical and print a message which asks for a hex number, then gets that number as an ASCII character string via BDOS call 10, then converts it to a binary value in a register.

In this subroutine, I'll call another subroutine, "UPCASE", which is usable in itself to change any alpha value from lower to upper case.

Here's the routine. To test it, I wrote the following four lines:

```
org 100h
lxi sp,300h
call inhex
rst 7
```

I then tested it under SID (DDT would do, too). The "rst 7" returns to SID/DDT when the program finishes.

Here's the actual program itself:

```
;
;routine to input a hex number from CP/M console.
;
inhex lxi d,hexmsg ;point to msg
mvi c,print ;get print fnc
call bdos ;print prompt
lxi d,hexbuf ;point to input buf
mvi c,rdconbf ;get function
call bdos ;read msg

;
;bdos doesn't send linefeed, so send one
;
mvi c,wrcon ;get function
mvi e,0ah ;get a linefeed
call bdos ;print it

;
lxi h,hexbuf+1 ;store 00
mov e,m ; into
mvi d,0 ; byte
dad d ; after
inx h ; last
mvi m,0 ; char

;
lxi h,0 ;init binary result
lxi b,hexbuf+2 ;init pointer

;
;process the characters, converting to binary
;
hexlp ldax b ;get a char
inx b ;point to next char
ora a ;end of buffer?
rz ; yes, return
call upcase ;change to upper case
cpi '0' ;if less than '0'
jc hexerr ; then error
cpi '9'+1 ;is it 0-9?
jc hexdig ; yes, a digit
cpi 'A' ;is it valid hex?
jc hexerr ; no, error
cpi 'F'+1 ;valid alpha?
jnc hexerr ; no, error

;
;process alpha hex digit
;
sui 'A'-10 ;make it binary
jmp hexmul

;
;got decimal digit
;
hexdig sui '0' ;make it binary
;
;multiply hl by 16, add in this digit
;
hexmul dad h ;x2
dad h ;x4
dad h ;x8
dad h ;x16
add l ;add digit to l
mov l,a ;store back
jmp hexlp ;loop back
```

```
;got invalid digit. Print message, re-prompt
;
hexerr lxi d,errmsg ;point to err msg
mvi c,print ;get print fnc
call bdos ;print it
jmp inhex ;go get input again

;
hexmsg db 'Enter hex value:$'
;
errmsg db 'Invalid hex, reenter',0dh,0ah,'$'
;
hexbuf db 5 ;buffer length
ds 1 ;length byte
ds 6 ;actual buffer
wrcon equ 2
print equ 9
rdconbf equ 10
bdos equ 5

;
;upper case subroutine
;
upcase cpi 'a' ;lower limit
rc ;return if less
cpi 'z'+1 ;upper limit
rnc ;return if higher
ani 5fh ;make upper case
ret
```

This time, for variety, I did the program in lower case. The assembler (ASM or MAC) doesn't care about case. However, be aware that earlier versions of ASM didn't handle things like the "cpi 'a'" properly - the value of 'a' was changed to upper case. A safer way to code this would be to "cpi 61h", which is the value of a lower case A, or "cpi 'A'+32", another way to generate a lower case A.

Let me discuss some aspects of this subroutine which may not be obvious. I made heavy use of my little gimmick for recalling the state of the carry flag after a compare: "CAL", or "carry if accumulator is lower". Thus, the first validation I did for a digit was "cpi '0'". This would set carry if the accumulator was less than an ASCII '0'. That's the reason for the "jc hexerr" following it.

Having taken the JC if the accumulator was less than '0', the test for being a '9' or less also uses the carry flag test. However, I couldn't just "cpi '9'" because carry would be set only for values *less than* 9. I want it to be set for '9' also, thus "cpi '9'+1". The actual value of that operand ('9'+1) is 3AH, or ':', but neither of these convey the reason I did the test. '9'+1 on the other hand, clearly explains that I am dealing with a test against '9'. Similarly, if the character wasn't 0-9, I tested for A-F, again with "'F'+1" to properly set carry if the digit was valid.

If the character I got was in the range '0'-'9', I JMPed to label "hexdig", which converted the number to binary, by simply subtracting '0' from it.

To convert an ASCII 'A' into a 10 is a bit more difficult. I let the assembler do that for me. Think of how you might do it: you want to subtract the 'A', but if you did only that, you'd get a 0. Thus, you must "leave 10 behind", i.e. "'A'-10". If that is not obvious to you, don't worry. It wasn't to me, either. Matter of fact, at first I coded "'A'+10". In testing the program, it produced the wrong value. I guess I'm more comfortable diddling with SID or DDT than going through the thinking to work it out in advance. I don't necessarily recommend this attitude, particularly for complex programs.

Next month, a bit more on character I/O: accessing the BIOS for character I/O, and the direct console I/O function supported by CP/M-80 2.2 and MP/M. Then, on to CP/M-80 disk I/O.

T.I.M. Data Base Software: First Impressions

Davis Foulger

Microcomputer data base software development is at a crossroads today; the market has shown that this type of product is here to stay, and a variety of software titles are gaining preeminence. Of course, more and more packages are being released and this segment of the software market is becoming increasingly competitive.

Users are demanding more for their money. They want software that performs a larger variety of tasks, faster, and with fewer demands on the user — who is often more concerned with getting a task done than with becoming an expert in programming a computer.

These demands have become more and more difficult to satisfy in the world of eight bit microprocessors. 64K has proven to be a very limited amount of memory in which to fit a sizable program and a reasonable amount of the data. Thus software developers have found themselves continually juggling user-friendly features, task-oriented features and the execution speed.

Today, a range of new machines offer a way out. With memory capacities ranging as high as one megabyte, and speedy 16 and 32 bit microprocessors, these machines have made fast, friendly and highly talented software possible.

T.I.M. (Total Information Management) has distinguished itself in the business software market in two ways. It was the first business software package for the IBM PC to be offered through a source other than IBM's software central. Second, it was the first data base management software package available on the IBM Personal Computer, which has sold more than a few copies of T.I.M. But being available for the IBM PC by no means guarantees that a piece of software takes advantage of the IBM PC's special talents.

It doesn't take much of a look at T.I.M. to realize that it is a talented and versatile database management system. Any given data file can have up to 32,767 records, no small amount. Each record can have up to forty fields. Each field, moreover, can be up to sixty characters

long, enough to allow the input of complete lines of text.

Unfortunately, as a statistician, I have a few applications that won't fit within those constraints. I could personally use a database that handled about 250 fields, most of them much smaller than the T.I.M. fields can be. But I have special needs — in truth, unless you are into measuring large numbers of variables you are unlikely to find T.I.M. constraining.

There are, moreover, more than a few other aspects of T.I.M. to recommend it. First, it is rather easy to use. Well-written menus guide one through the various functions of T.I.M. easily. Indeed, a careful reading of the chapter entitled Preliminary (Chapter 0) gives you most of the information that cannot be readily gleaned from the menus.

The functions available include the necessary commands for creating a file; adding records to, updating and inspecting a file; displaying the files on a particular disk, getting help, erasing files, renaming files, sorting fields and records, selecting records and generating lists and reports. Beyond these functions, additional functions enable the user to write files for use in a word processor, read ASCII files into T.I.M. files, add new fields to existing files, and do a variety of other useful things to and with the T.I.M. database file.

These features make T.I.M. a fairly powerful database management system. Unfortunately, many of T.I.M.'s capabilities can lead to problems. First (in the IBM PC version), it resides on three disks, which must be shuffled constantly during some of T.I.M.'s operations. This is not, of course, so much due to T.I.M.'s deficiencies as to the limited capacity of the IBM PC's single-sided disks (160K capacity in SB-86). This will be alleviated when IBM's double-sided disk drives reach the market. But the present arrangement accentuates another of T.I.M.'s problems, its speed.

T.I.M. seems to take an inordinate amount of time to do just about any-

thing. Even shifting between records takes a noticeably long time. With some tasks, like sorting, you can do something else while waiting for T.I.M. to complete the job. Since disk changes are often required in the course of the operation, however, you frequently can get little accomplished while waiting. The long tasks are not as frustrating, however, as the extra half second it takes for T.I.M. to move from one record to the next.

The speed problem seems to arise from two sources. First, T.I.M. is written in BASIC, a convenient language in which to write software, but not necessarily the fastest language in which software can be run. The solution to this seems simple enough — compiling the BASIC with the new BASIC compiler promised for the IBM PC later this year. A still better solution would be converting and optimizing the code in assembly language. Each action would speed T.I.M. considerably, and one suspects that those actions will ultimately be taken.

The second problem, however, is the fact that T.I.M. doesn't adequately use the extended memory capabilities of the IBM PC. Again, this is partly a function of using BASIC, which doesn't take advantage of the IBM PC's extended memory capabilities; but T.I.M. could have better utilized the space available. Right now, it appears to read only one record into RAM at a time. Clearly, more records could be in memory at any given time, especially in a version for the many IBM PC's with extended memory. Surely such a measure would speed up T.I.M.'s execution on a machine that is no slowpoke.

On balance, however, T.I.M. remains a very strong database management package that shows a great deal of promise for satisfying the needs of most microcomputer users.

This article is a brief introduction to T.I.M. A future article will study T.I.M. in more depth and include benchmark tests, so that you can get more of a feel for T.I.M.'s capabilities.

The Basics of Microcommunications Software

Davis Foulger

In last month's *Introduction to Microcommunications* we explored the various ways in which microcommunications might be used. I argued that there were generally two sets of microcommunications applications: communications to other microcomputer users and access to remote timesharing services. It was also noted that microcommunications can be less expensive than other modes of communications (including telephone calls and first class mail, under specified conditions). This month we look at the nature of microcommunications software, examining the features good microcommunications software must have. Next month we will extend this discussion by discussing the nature of "ideal" microcommunications software packages.

It isn't always necessary to use a microcomputer for communicating with another computer. A terminal is more than adequate for many data communications applications. Basically, a terminal is nothing more or less than a communications device, whether used with a modem for remote computer communications or as the input/output device for a microcomputer. It comes with its own built-in communications firmware, and sometimes even has a modem built in. These integrated features make the terminal a simple communications device to master.

As a result, terminals are often used for computer-based communication. In the business and academic world, where access to mainframe computers has made more powerful equipment unnecessary for a wide range of applications, the terminal fills a need. Terminals are often shared among many users who access them as required. For home users, the terminal is a low cost introduction to consumer timesharing services. Indeed, consumer terminal equipment lacking nothing but a television for display can be obtained for as little as \$400.

Why Microcommunicate?

That \$400 buys a device that can communicate with another computer, but it can't do much more. It cannot, for instance, save the information received from the other machine. It does not let the user prepare messages off-line, and won't permit the user to edit the information received in a word processor or to use that information in a spreadsheet program. And it's dependent on the functionality of the main computer.

Here, of course, is the value of substituting a microcomputer for a terminal. When microcommunicating, the microcomputer can do anything a terminal can do, and more. When not microcommunicating, the microcomputer can perform many other tasks. Indeed, it can perform many tasks with greater facility than the remote timesharing computer it communicates with.

The software for communications is not built into a microcomputer, as it is into a terminal, and a modem is rarely built into a microcomputer. Providing these "extras" on the microcomputer is not inexpensive. Indeed, the modem and software alone may cost as much as a terminal that comes with the modem and communications software built in. It is not unusual to find only modestly appointed microcommunications software packages priced at \$150 or more. Modems can cost considerably more. The software, modem and microcomputer must then be mated, a task that can take no small amount of effort, at least initially.

The versatility of the microcommunications system more than makes up for its complexity, however. A single microcomputer can emulate a wide range of different terminals, according to need, while providing access to features that are generally unavailable in any terminal mode. These features may include the ability to reconstruct and

reformulate data for particular purposes at the terminal and the ability to write messages for later transmission even as remotely generated information is being displayed on the terminal. A transcript can be kept, allowing the user to reconstruct information that might be lost in case the communications or remote computer fails. The microcomputer can, moreover, be used for other functions while the remote equipment is not working.

Indeed, these "off-line" functions represent the primary uses of microcomputers today. Because of differences between displays on different terminals, some of these functions are easier to implement on the microcomputer than they are on the mainframe computer. Because others are memory and processor intensive tasks, they are more efficiently implemented on the microcomputer. Thus, substituting a microcomputer for a terminal not only increases the reliability with which microcomputer-sized tasks can be performed (off-line) and increases the versatility of communications with a remote mainframe, but reduces the load of small tasks on the mainframe, freeing it to work on the large scale tasks to which it is best suited.

A Microcommunications Machine

The choice of modem and software can be an involved one. A recent listing of communications software packages ("Software Vendor Directory"; available from Hayes Microcomputer Products Inc., 5835 Peachtree Corners East, Norcross, GA 30092) lists nearly fifty different communications software packages currently being marketed for different microcomputers. The modem market is very nearly as wide open (witness our review of the modem market in the June *Lifelines/The Software Magazine* — The Pipeline).

Microcomputer users are often surprised to learn that they will need special software to use their microcomputer as a communications machine. This confusion stems from the similarity in appearance of the terminal and the microcomputer. Both have a keyboard and, more often than not, a CRT display. Occasionally, the terminal even carries disk drives, RAM and a microprocessor.

The terminal is, however, a dedicated communications machine that comes with its own built-in communications software. A microcomputer, on the other hand, is a versatile applications machine that can communicate if fed the right software. Unlike the terminal, which generally only knows how to communicate, the microcomputer can do almost anything you want it to. But it doesn't do anything unless you tell it to.

Of course, the terminal function could be "hard-wired" into the microcomputer. Fortunately, it usually isn't. Although putting communications software on your computer will require an additional investment in time and/or money, the freedom you will get from being able to use the microcommunications software of your choice will more than pay for itself over the long term.

Understanding Communications Software

Software is generally the most difficult feature of the communicating microcomputer for a user to understand. Software creates the machine for the user, telling the machine what to do while guiding the user through the doing. If it does its job well, the user should hardly notice it. This is particularly true with microcommunications, where the machine doesn't seem to be doing anything more difficult than passing keystrokes between different machines.

A friend and I have spent the last month and a half sharing the results of our efforts to develop our own microcommunications software. He writes an enhancement to his program and calls me up to test the new features of the package. He microcommunicates the software to me so I can use it and he can see what it looks like from his end. I write an enhancement to my program

and call him up to help me out.

The exercise has been good for both of us, partly because we each have learned a lot about the inner workings of communications software, but mostly because it has enabled us to develop highly personal microcommunications software. Pieces of his software have crept into mine. Pieces of my software have crept into his. But we have developed very different packages.

One of the joys of implementing microcommunications software is the flexibility you can introduce. Screens can be split to separate messages received from messages sent. Data received can be automatically reformatted into a spreadsheet or rebuilt into graphics. Software can be made to save a transcript of a microconversation and to transmit a copy of a pre-existing file, all while saving new keyboard input (another message) for later transmission.

The problem, as my programming friend has said, is that everybody has a different idea of what "ideal" microcommunications software looks like. That's why there are at least fifty different commercially available communications software packages. Each does something different, according to the vision of the particular author; while most can communicate with each of the others (sometimes with effort), most have some features that only work when the same software is at the other end of the microcommunications connection.

I currently own five widely varying pieces of communications software for my IBM PC (reviews of some of these will be appearing in future issues of *Lifelines/The Software Magazine*). None are the same. Some don't talk to each other very well. No single package fully satisfies me. That is, of course, consistent with my friend's comment. It may be that I will never find my "ideal" microcommunications software.

Clearly, good microcommunications software does a great deal more than simply transfer keystrokes between computers.

Emulating the Terminal

The first thing a competent microcommunications software package must do

is emulate a "dumb" terminal. This "emulation", the ability of the microcomputer to look like a terminal to a distant mainframe, really isn't terribly complex, but it does involve a number of microcomputer programming tasks. These tasks basically entail the definition of the characteristics of an array of input/output devices and the creation of a processing logic to govern the movement of data between those devices.

The most basic of these input/output devices is the one that will connect the microcomputer to the other computer. But other devices, including those that create the machine's interface with the user and its various peripherals, must also be defined. In reality, the task is one of creating personalities for the microcomputer. One personality must be agreeable to the remote computer, and another must be agreeable to the user. Neither personality is necessarily dependent on the other, but both must be consonant with the microcomputer's capabilities.

If it weren't for this need to use the special abilities of the microcomputer to advantage, the task of creating those personalities and a processing logic to connect them might be a simple one. Indeed, IBM publishes a simple dumb terminal program in its Personal Computer BASIC manual that accomplishes the task with less than thirty BASIC commands. The result of that short program is a microcomputer that does everything a dumb terminal does, and nothing more. Keystrokes are carried between computers and displayed for viewing. Getting microcommunications software to do more than that, to take advantage of the special characteristics of the microcomputer, requires a great deal more code.

The program does, however, perform those functions fundamental to a successful microcommunications program. First, it establishes a communications port from the microcomputer, a relatively simple task on the IBM PC, which has a special BASIC language command for opening a user-defined communications port. On many other machines, the communications port would have to be opened at a specific memory address. The characteristics of that port, moreover, would have to be defined in user-written assembly language routines.

(continued next page)

The communications port is defined by several characteristics, all of which have an important impact on how the microcomputer will appear to a remote computer. The port's baud rate, for instance, establishes the speed with which the microcomputer will be able to move data.

This *baud rate*, which can typically vary from 75 bits per second to 9600 bits per second, is constrained by a variety of factors. If communications is being established between two computers in the same room, speed will be limited only by the speeds attainable on the individual machines using the available software. If the user is transmitting over telephone lines, they will generally be limited by the speeds available on the users' modems, frequently 300 baud among microcommunicators. Note that if none of the speeds available through a user's microcomputer/software/modem combination are available on the remote computer, communications will not be possible.

A second characteristic, the **data format**, (the length of a transmitted or received character), generally varies from four bits to eight bits, with the seven bit data format being by far the most commonly used in microcommunications. This data format determines, on the one hand, how fast characters (keystrokes) can be transmitted from one machine to the other, and on the other, the variety of characters that can be transmitted. A four bit format, for instance, could deliver characters to the remote computer twice as fast as an eight bit format (depending on other factors). But where the eight bit format could deliver 256 different characters, the four bit format would only deliver sixteen.

A four bit format would allow a user to send a full set of numbers (zero to nine) and some control characters, but nothing more. A five bit format (thirty-two characters) would allow the transmission of a full upper case alphabet (A to Z) and some control characters, but little more. Six bits (sixty-four characters) provides for a full upper/lower case alphabet (A to Z and a to z), but the inclusion of numbers leaves no room for punctuation or control characters. It should be obvious, of course, that each of these formats has applications for constrained data and message transmission. All are used from time to time in microcommunications applications.

The speed increments that these formats allow do not, however, generally balance well against either the restricted range of characters available in four to six bit formats, or the added handling necessary to recode the characters transmitted in these formats in the microcomputer's eight bit data format. As a result, few microcommunications software packages offer four, five or six bit formats as options, preferring to stick with seven and eight bit formats.

The seven bit format (128 characters) allows the transmission of the complete ASCII character set, including a full upper/lower case alphabet, numbers, punctuation, special characters and a wide range of control characters. The eight bit format (256 characters) provides for the transmission of system software between compatible microcomputers and the transmission of a wide range of special characters, including graphics characters.

Error Checking

An advantage of the seven bit format is the capacity for character-by-character error checking. Inevitably, transmission errors will occur - all it takes is a change in a single bit to turn a "y" into a "9". This probably isn't all that important in most microcommunications applications, but it can represent the difference between a clean running program and several hours of debugging.

Parity checking, as this character-by-character error checking is called, involves the addition of an eighth "parity" bit, either a one or a zero according to the form of the parity check and the particular character. "Odd" and "even" parity checking operate the same way but take different forms. Both involve adding up the number of ones in each character.

In even parity, the eighth bit is coded as a zero if the total of the first seven bits of the character is even; it is coded as a one if the total is odd. The result is the transmission of eight bits: seven bits of data and one parity bit, the total of which is always even. If the remote computer operating with even parity receives an "odd" character (a character whose seven data bits and single parity bit total to an odd number), it knows that a transmission error has occurred and can request the retransmission of the message.

Odd parity works the same way, but uses the parity bit to make the total of the bits transmitted odd instead of even. The choice between the two is trivial, but making sure that both machines use the same form of parity is critical. If your microcomputer is receiving odd parity while the remote computer is transmitting even parity, every character will seem to be in error (except those that actually are).

Because transmission errors are relatively unusual, parity is often ignored, but because it is easier to transmit eight bit characters than seven bit characters, the parity bit is often filled anyway, either with a "zero" bit (space parity) or a "one" bit (mark parity). Space and mark parity can provide limited error protection (if the eighth bit of every character is supposed to be a one and a zero shows up, something is wrong). Generally, however, they are used as a convenience - a way to fill up a meaningless eighth bit - and nothing more.

When the full eight bits are used for data, there is no room for a parity bit, and if error protection is to be provided, it must be provided in other ways.

Coordinating Microcommunications

A ninth bit may or may not need to be added to a transmission, depending on how transmissions are coordinated by the communicating computers. There are generally four circumstances under which the coordination of communicating computers can be important. Together they lead to a cluster of characteristics that must be taken into account when building microcommunications software.

The most basic circumstance under which coordination is important occurs when communications must flow in two directions over a single wire. Quite obviously, data cannot move in both directions at once without creating noise on the line and, probably, transmission errors. The solution is to coordinate the computers so that information only flows in one direction at any given time.

This coordination, called *half duplex*, depends on the use of special turn-taking characters. When one machine

finishes sending a message it sends a turn-taking character, signalling the other machine that it can reply. That machine, in turn, signals the end of its reply, allowing the first machine to transmit again. Half duplex microcommunication is not unlike a paper conversation in which one person writes a note on a piece of paper and passes to another person, who replies on that same piece of paper and passes it back.

Full duplex, by contrast, allows simultaneous communication in both directions. Full duplex is sort of like a paper conversation with two pieces of paper. You don't have to sit around waiting for the other person's message to come so that you can write another one. This can, of course, get confusing, especially in microcommunications, where the simultaneous writing of messages can, if not carefully controlled or formatted, lead to lines of gibberish - with the characters from one person's message thoroughly mixed in with the characters from the other person's message.

This gibberish leads us naturally into a second kind of coordination, *synchronous communications*. Although not commonly used for microcommunications, synchronous communications represents an important way of speeding up the transfer of data between computers. In synchronous communications, the communicating computers are synchronized to a common clock signal and transmit data in accord with that clock. The coordination of the machines in this way cuts down on the need to use other kinds of coordination, speeding up the exchange of data by eliminating unnecessary bits and control characters.

The value of synchronous communications is probably better understood when it is contrasted with *asynchronous communications*. In asynchronous communications, data is transmitted and received one character at a time, with an additional one or two *stop bits* signalling the end of each character. Where each character is represented by eight bits (or seven bits with parity), nine bits are transmitted for every character.

This has its advantages, especially where transmission is irregular, as would be the case in many microcommunications applications, but it does reduce the effective data transmission speed. Eight bit characters translate to

37.5 characters per second at 300 baud. Nine bit characters translate (at 300 baud) to 33.3 characters per second.

Whether coordination of character transmission is achieved with stop bits or a clock, such coordination must be maintained. The clock allows characters to be transmitted a line or a block at a time, speeding up the overall data flow. But maintaining such synchrony is expensive, and unless data is flowing at maximum speed continuously over most of the transmission period (an unusual situation in microcommunications), asynchronous communications will be less expensive for the user and easier on any remote timesharing computer the microcommunicator may dial up.

Another form of coordination that is commonly used in full duplex, asynchronous communications system is the *stop character* (sometimes referred to as XOFF), which is used when a machine is close to having received more characters than it can handle. Sending the XOFF character (generally ASCII character 17 - hex 11) is the microcommunications equivalent of saying "Shut up, I can't take anymore". XOFF is reversed with a "go" character, XON (generally ASCII character 19 - hex 13), which essentially says "You can start talking again".

Provision for these various forms of coordination will vary in importance depending on the kind of microcommunications you are involved in. Few microcommunicators presently need to worry about synchronous communications. Most, however, will have to deal with stop bits and stop characters, which are important to the full duplex asynchronous communications system.

Taking Advantage of the Microcomputer

There are several different elements of communications software important in making the microcomputer communicate with a remote computer. Complicating the plot somewhat, however, is the fact that different computers, each using their own data communications software, recognize different terminals.

Some support stop characters. Others don't. Some operate half duplex. Others expect the terminal to echo characters back to the host (retransmit

characters received so that the host can be sure there were no errors). Some echo characters back to the terminal. Others don't. Some use seven bits, some eight bits. Among those that use seven bits, any of four forms of parity may be encountered.

Good communications software must adapt to these differences, allowing for variations in the types of terminal the communicating microcomputer emulates. This adaptation cannot be achieved in the less than thirty instructions which can turn an IBM PC into a terminal. Indeed, IBM's asynchronous communications package contains thousands of instructions, each of which offers flexibility in communications. But there are variations in microcommunications not accounted for in the software. Fortunately, the program, written in BASIC, is easy to modify, which cannot be said of many microcommunications packages.

After the terminal interface to the remote computer is taken care of, the microcommunications software developer can start to have some fun. There are still some tasks to complete before the necessary can give way to the ideal, but the biggest hurdles have been jumped. When the terminal interface is finished, the microcomputer will talk to other computers. Now it must communicate with people.

This job starts simply enough, with routines to move a user's input from the keyboard to the communications port and, when the remote computer is not echoing characters back to the terminal, to the CRT, printer and, where desired, disk storage. Still other routines will be needed to move data from the communications port to the same CRT, printer and disk storage. Special characters need to be recognized, operated on, and, where necessary, deleted.

The task heats up as the software begins to take advantage of the special characteristics of the microcomputer. Menu-driven special functions can be built in to allow files to be saved to disk; to permit files saved on disk to be transmitted; to let users look at the directory of the files they have available; to allow review of the information received or stored on disk; to change the form of the terminal in the midst of the terminal contact.

(continued next page)

These kinds of features are necessary components of good microcommunications software. No microcommunications package can expect to succeed if it doesn't take advantage of the machine's ability to reprocess and electronically store the information flowing through it. Otherwise, the microcomputer will be wasted as a communications device.

Giving The User Interface A Personality

When the bare necessities are complete, the software developer must attend to the user interface. The necessity of creating a good user interface is easy to underestimate, but it is as important for the user to be comfortable with microcommunications as it is for the computer to be comfortable with the terminal.

Good software attracts the user by guiding her or him through a given task in the same way it guides the machine. Function keys are useful where available, as are simple, uncluttered menus that remain visible at all times, but which change as the user moves through various tasks, anticipating user decisions to the largest extent possible.

The real fun in creating the user interface comes with the realization that microcommunications do not have to be presented to the user in the same fashion that the microcomputer is presenting itself to the remote computer. The structure of data communications is full of opportunities for building creative user interfaces.

Asynchronous communications need not show its character-by-character delivery. Although communications packages persist in mirroring the terminal structure of asynchronous communications by delivering each character to the modem as it is written, the microcomputer has the flexibility to present a very different face to the user than the one it presents to the modem.

Characters can easily be made to *seem* like they are being delivered a line at a time or in blocks, as is done in synchronous communications systems. Instead of instantly sending each character off through the modem as it is written, communications packages could simply display the character on

the screen until the line was finished. When finished, the line could be sent off through the communications port. In the meantime, it could be edited and changed without having to reveal all the spelling and logical mistakes made in writing the message.

Of course, the communications would still be asynchronous and the message would still be sent a character at a time. Still, the system would look synchronous to the user, who would also gain some of the advantages that go with a synchronous communications system.

There are disadvantages to this kind of "quasi-synchronous" presentation. It is likely to keep you connected for three seconds longer than you might be otherwise (it takes about 2.5 seconds to transmit an 80 character line at 300 baud). In computer conversation, moreover, the wait for any sign of a message can seem interminable without some mechanism for keeping things going. Fortunately, just such a mechanism exists.

If one notes that the full duplex communications interface is fully simultaneous (allowing two microcommunicators to "talk" to each other through their keyboards at the same time), the solution becomes readily apparent. Full simultaneity means that each person can potentially respond to the other's messages as they are read. In other words, people don't have to sit around waiting for another's message to come down the pike. With the proper software, the user can write whenever he or she wishes.

The problem is, of course, that most of the currently available communications software does not permit this to occur. We have evolved a tradition of developing and selling communications software packages that scroll up over the entire screen in the way a transcript is written.

Mind you, there is nothing wrong with a transcript. Indeed, I'm all for keeping a computer transcript of useful computer conversations. But the transcript format encourages microcommunicators to talk over the computer in the same way they might talk on the telephone. One person talks. The other person talks. The first person replies. If both people talk at the same time, the result is unintelligible.

A creative user interface could solve this problem in a number of ways. Screens could be split to allow one person's comments to be displayed on the top while the other person's comments were displayed on the bottom. Another screen split might place comments side by side, with thirty-nine or forty columns of a eighty column screen devoted to each person. Split screens do, however, create a problem. How do you save a transcript in a usable form?

Word Wrap And Paragraphing

A still better solution would combine some of the features of word processing with the microcommunications software, so that computer could "create" a transcript out of the simultaneous dialog. Microcommunicators would write in the same way they might write on a word processor, with a word wrap feature keeping track of the line feeds, "soft" carriage returns and general format of the screen while the interactants concentrated on their messages.

Microcommunicators would then only be obligated to hit the carriage return when they were finished with an idea. The microcommunications software would order those ideas, moving each communicator to a new frame (below the bottom message on the screen) with each carriage return.

The result of this "conversation processing" would be an ongoing transcript that kept entire thoughts both in one piece and in relative proximity to related comments. It could be saved for future reference and editing, if desired, or simply read as it went along, with each microcommunicator able to read and write at the same time without fear of gibberish.

Actually, word wrap just by itself is a pretty good idea for inclusion in future microcommunications packages. I hate to think of the number of times I've lost lines of remote computer input because I neglected to hit a carriage return in the middle of a message.

These features are by no means a necessity in a microcommunications software package. They do, however, suggest possibilities for creating an interesting and friendly user interface to the microcommunications machine. Many other features could be added.

It would be desirable to connect microcommunications software more closely with other software - like word processing, spreadsheet mathematics, database, business graphics, games, statistical packages, etc. We are, however, beginning to stray into the realm of the ideal, and that is the subject for next month.

In Microconclusion

There's no reason that you *must* use good microcommunications software to communicate with someone else. In truth, you can get by with as few as thirty BASIC language instructions.

But if you're going to spend a signifi-

cant amount of time using communications software, it should at least be comfortable, and do as much of the work as possible. It ought, moreover, to point to future microcommunications software, which will involve voice input and integrate other software applications with microcommunications.

Software Notes

Getting Off Base

Michael Olfe

Since most of us don't have CPUs that execute Ada statements as object code, we sometimes find ourselves in situations where the higher-level language we're using prevents us from getting at some very fundamental lower-level features. When you find yourself creating submit files like this from within your dBASE program, you are probably experiencing this particular angst:

quit to "STAT *.*", "DBASE SAMEPROG"

Figure A

```
A> ddt
  NEXT  PC
  0100  0100
  -a100
  0100  hlhd 5
  0103  shld <safeaddress>
  0106  ret
  0107  .
  - g0
A> save 1 getad.com
A> getad
A> ddt
  NEXT  PC
  0100  0100
  -s<safeaddress> XXXX

  -HXXXX,0a400
  SSSS,DDDD
```

All gentle persons will cringe at having to do such a thing. There is, luckily, another way. Version 2.3B has provided us with "PEEK", "POKE", "CALL", and "SET CALL TO". With these we have access to any machine-language routine or CP/M-80 system call as a subroutine to dBASE.

The goal is to be able to execute a subroutine or group of subroutines without leaving dBASE. We would like to be able, for example, to call SD.COM (the Sorted Directory program on CPMUG) from within dBASE for a list of filenames, sizes, and remaining space on the disk, so as to avoid a disk full error when adding records to a database.

Here is how to do it:

- 1) First determine how much RAM you have between 0A400h and BDOS. Use the method in Figure A or your own method to do this.
- Note: 'safeaddress' is some address which will not be overwritten by DDT when you load it again; XXXX is the address of BDOS; DDDD is the difference between BDOS and A400, or how much space there is between them.
- 2) The total length of all the called subroutines can be no greater than the number determined in Step 1.
 - 3) Edit your routines (or write them) for origin 0a400h. With 'SD.ASM', for example, this can be done by just changing the equate of TPA to 0a400h. If there are multiple entry points, note them. You will have to specify to dBASE what addresses to call.
 - 4) Find where the routine returns (either to CP/M-80 or to the calling routine). This could be a JMP 0, an LXI SP, <adress>, RET, or some such sequence. Change any such returns to simple RET statements.
 - 5) Re-assemble the routine.
 - 6) Load dBASE with DDT, then load the routine on top of it and save the large (approx. 44K) file.
 - 7) Exit DDT and save the new file under a new name.
 - 8) Write a calling routine like the one below.
 - 9) As long as you avoid a "SORT", which you will probably never use if you have indexed files, your routines will always be available for calling from the modified dBASE.

This method allows you to access commonly-used programs and CP/M-80 system calls from dBASE with minor modifications to them. The dBASE "CMD" file is general - unless parameters have to be passed back and forth, the sample CMD file below can be used to call any subroutine.

(continued next page)

(Getting Off Base continued from previous page)

The only disadvantage I see in this is having to include the routines as part of dBASE instead of being able to load them. Can anyone come up with a way of loading binary files at arbitrary memory locations from within dBASE?

```
* Sample dBASE "subroutine" call with parameter passing
* M. Olfe 5/19/82

* Parameters are passed to the subroutine through
*the CP/M-80 FCB at 5DH so that no modifications
* need be made to the subroutin itself
* (which in this case was "SD.ASM").

set echo off
set talk off
erase

* display current fcb

stor (5*16)+13 to fcbl
stor 0 to count
stor ' ' to s
do while count<12
  stor s+chr(peek(fcbl+count)) to s
  stor count+1 to count
endd
@ 10,0 say "Current fcb= "+s

* blank the fcb
poke fcbl,32,32,32,32,32,32,32,32,32,32,32,0,0,0,0

* get parameters for called program

stor ' **'
  to param

* truncate,capitalize for CP/M

stor trim(param) to param
stor !(param) to param

* convert ascii to decimal -- dbase v2.3b has no 'ASC()'
* function

stor '*+,-./0123456789;<=>?@ABCDEFGHIJKLMNPOQRSTUVWXYZ[\]^_`' to table
stor 41 to base
stor ' ' to dstring
stor 0 to pointer
do while pointer<len(param)
  stor $(param,pointer+1,1) to srchchar
  stor @(srchchar,table) to position
  stor dstring+str(position+base,2)+' ,' to dstring
  stor pointer+1 to pointer
enddo

* take off the last comma

stor len(dstring) to leng
stor $(dstring,1,leng-1) to dstring

* poke the fcb

stor 5*(16)+13 to fcbl
poke fcbl,&dstring

* set up the call address

STOR 10*(16*16*16) TO CA
STOR CA+(4*(16*16)) TO CA
set call to ca
stor 'dummy' to adress

* do the call

CALL adress
```

Change of Address

Please notify us immediately if you move. Use the form below. In the section marked "Old Address", affix your *Lifelines* mailing label — or write out your old address exactly as it appears on the label. This will help the Lifelines Circulation Department to expedite your request.

New Address:

NAME

COMPANY

STREET ADDRESS

CITY

STATE

ZIP CODE

Old Address:

NAME

COMPANY

STREET ADDRESS

CITY

STATE

ZIP CODE

Digital Research's Pascal MT+ compiler optionally includes a useful utility called the Speed Programming Package, containing the following functions:

- a screen-oriented text editor
- an interactive syntax scanner
- a variable-name spelling checker
- a pretty-printing source code reformatter
- a short-cutting fast compiler
- a version-renumbering and logging facility.

This program must be patched with terminal control characteristics for each terminal; the existing file named NSB.SRC is configured for the Televideo SOROC terminal. In the middle of that file is a clearly-labelled User Modification Area; Figure A is a listing for a patch for part of that area, which works on the HP125 terminal.

Figure A

```
PROCEDURE XYGOTO(X,Y:INTEGER);
BEGIN
  SB_OUT_CH(CHR(ESC));
  SB_OUT_CH('&');
  SB_OUT_CH('a');
  SB_OUT_CH(CHR((X DIV 10) + 48));
  SB_OUT_CH(CHR((X MOD 10) + 48));
  SB_OUT_CH('c');
  SB_OUT_CH(CHR((Y DIV 10) + 48));
  SB_OUT_CH(CHR((Y MOD 10) + 48));
  SB_OUT_CH('Y');
  SB_LAST_X := X;
  SB_LAST_Y := Y;
  (* THESE ARE USED ONLY BY USER *)
  (* SOFTWARE ROUTINES THAT PERFORM *)
  (* CLR TO EOS AND CLR TO EOL *)
END;

PROCEDURE SB_CLR_SCRN;
BEGIN
  SB_OUT_CH(CHR(ESC));
  SB_OUT_CH('H');
  SB_OUT_CH(CHR(ESC));
  SB_OUT_CH('J');
END;

PROCEDURE SB_CLR_EOS;
BEGIN
  SB_OUT_CH(CHR(ESC));
  SB_OUT_CH('J');
  SB_OUT_CH(CHR(0)); (* GIVE IT TIME TO WORK *)
  SB_OUT_CH(CHR(0)); (* GIVE IT TIME TO WORK *)
  SB_OUT_CH(CHR(0)); (* GIVE IT TIME TO WORK *)
  SB_OUT_CH(CHR(0)); (* GIVE IT TIME TO WORK *)
END;

PROCEDURE SB_CLR_LINE;
BEGIN
  SB_OUT_CH(CHR(ESC));
  SB_OUT_CH('K');
END;
```

(* User modification area ENDS WITH SB_CLR_LINE *)

After NSB.SRC is patched to read like Figure A, it must be compiled (by the command A>mtplus nsb) to create an NSB.ERL which will become part of the input for a lengthy link procedure to produce the operative SPP files. Since the code in the patch is somewhat longer than the original, the buffer pointers in all the .CMD files must be modified before linking, as in the samples below. SPPMAIN.CMD should look like:

```
SPP=NSB
A:PASLIB/S/V1:1780/X:1480/D:8280
```

and the last number in all other .CMD files should read 1780 instead of 1600.

You who have versions of the H-P BIOS earlier than 1.1 should get an update. But here's a temporary fix. In the absence of a WELCOME.COM file on A:, the SUBMIT procedure will die upon warm-boot; using DDT (available on the H-P programmer's utility disk), a dummy WELCOME.COM may be created as in Figure B.

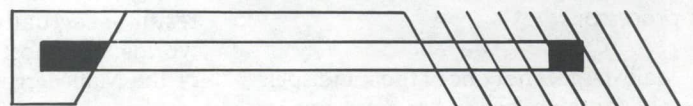
Figure B

```
A> DDT <cr> (after signon, you will get a hyphen prompt)
  -a100 (start assembly input at address 100)
100 RET (sole contents of dummy file is a RETURN command)
101 . (no more code needed, leave assembly loop)
  -g0 (to exit DDT, leave one-byte file in memory)
      (NOTE: that's a zero, not the letter O!)
A> SAVE 1 WELCOME.COM
```

Figure C is a directory of disk b:, ready for A>submit linksp. The a: disk contains submit.com, paslib.erl, linksp.sub and linkmt.com (at least). The b: disk should contain nothing more than files shown in Figure C, as the Hewlett-Packard CP/M-80 allows only 64 directory entries, and the linkage procedure will create several temporaries which take up directory space.

Figure C

```
A>dir b:
B: NSB      ERL : SB008  CMD : E2      CMD : SB006  CMD
B: SB001   CMD : E2      ERL : SBRUN  ERL : CPYINS  ERL
B: DIROVL  ERL : UTILMOD ERL : SYNCHECK ERL : MISCLIB  ERL
B: VARCHCK ERL : EDITWRIT ERL : SBINIT   ERL : SBLOG   ERL
B: UTIL3   ERL : CHN2    ERL : CMDS    ERL : SR      ERL
B: PRETTY  ERL : PPMOD1  ERL : PPMOD2  ERL : PPMOD3  ERL
B: SCANNER ERL : IDSEARCH ERL : MINIASM  ERL : TRAVERSE ERL
B: PPINIT  ERL : E3      CMD : SB003  CMD : SB00A  CMD
B: SB004   CMD : SPPMAIN  CMD : SB007  CMD :
```



MailMerge, Do I Need It?

Robert P. VanNatta

Have you ever plunked down some hard earned clams for a computer program - only to find that it gathers dust in your closet? This writer would suggest that if you are tempted to answer this question with a 'no', you are either a liar or don't own a computer.

This assertion is made, not for the purpose of suggesting that software merchants are rip-off artists (which they usually aren't), but rather to emphasize the difficulty in evaluating the usefulness of a particular program by looking at the program name.

MailMerge is the program name assigned to one of the extra cost options to the WordStar word processing program. The purpose of this article is, hopefully, to provide the reader with enough information to permit an intelligent purchasing decision with respect to this program.

MailMerge is an overlay to WordStar and will not work without WordStar (more specifically, the same version of WordStar). Obviously unless you either own WordStar, or are contemplating its purchase, MailMerge is out of the question.

Those of you who have gone through the agony of buying a computer, only to learn belatedly that the screen, the keyboard, the memory boards, the disk drives, and the printer were all optional at substantial extra cost, may well wonder if the the word 'optional' when applied to MailMerge has a similar meaning.

The answer to this question is a qualified no. Those of us who have been using WordStar for several years, as I have, recall that MailMerge was first introduced in the summer of 1980 with version 2 of WordStar. (It was then called 'Mergprin'.) Prior to that time we all got along without it and, indeed, WordStar will, without MailMerge, do almost anything expected of a word processor.

MailMerge is not one of those indispensable options, but rather a genuine ex-

tension of an already useful program. As nearly as this writer can discern, authors of word processing programs have never fully agreed on whether page formatting (linespacing, margins, page advances, etc.) was best accomplished as part of the editing function or as part of the printing function. The printing function writers argue that formatting the text while it is on its way to the printer is the simplest and most flexible way to accomplish the job and darkly suggest that the edit style of formatting results in a befuddlingly complex editor. The edit formatters argue the other side and claim that complexity is justified by the fact that the operator doesn't need an imagination to figure out what the printed document is going to look like, since the screen is a reasonable facsimile of the ultimate printer output. The dichotomy is not unlike the logical distinction between the LIST and RUN commands of an interpreter BASIC.

WordStar is advertised as a word processing program featuring "What you see is what you get!", meaning that, generally, your formatting is performed on screen and the appearance on the screen will be reasonably reproduced when the printer is activated. This 'garbage in-garbage out' approach is reflective of the real world, in that the average hacker usually manages to master the LIST command of BASIC before getting anything meaningful out of the RUN command.

This selling point distinguishes WordStar from some of its competitors, where more formatting may be performed subsequent to the editing routines. (The format commands are usually passed to the formatter by extensive 'dot' commands or by long printer menus.)

Interestingly, MailMerge is a print time formatter which is capable of overriding the edit time formatting. The result is actually the best of both worlds. The most fundamental feature of the MailMerge option is the ability of the user to insert 'code words'

(technically string variables) into the text and have the assigned value of these variables printed in lieu of the variable.

Consider the following BASIC program for example:

```
X$="Output" : PRINT X$
```

This or a reasonable facsimile thereof (depending on the dialect of BASIC) will result in the word "Output" being printed somewhere, and is illustrative of the substitution feature of MailMerge. In the MailMerge implementation this feature is used by declaring the string variables as: &X& instead of using the dollar sign traditional in BASIC. The result is the same. Any reasonable number of variables so declared may be interspersed through the text; when the document is printed the values of these variables will be substituted for the variable name. Thus whenever &X& is inserted the value of 'X' will be printed and not the character 'X'.

Now suppose that your task is to print out several letters which are identical, except for some name or address changes. This could, of course, be done by editing as many different files as you have letters and by making appropriate changes to each one, or by making one giant file consisting of repeated copies of the letter separated by a page advance dot command. The block copy and block read commands make this sort of procedure relatively quick and easy, but it is obvious that such a procedure would quickly get out of hand if one had fifty or one hundred (or maybe several thousand) letters to prepare.

But, suppose that you could as an alternative create just a single document interspersed with appropriate 'code words' where changes are to be made, and this single document could then be printed repeatedly, while you merely indicate during the print routine, what should be substituted for each 'code word'. The advantage is obvious. You have the effect of a global search and replace routine without actually hav-

ing to perform it. The code words are not physically replaced, but whenever the document is printed the replacement value is printed in place of the code word.

"Sounds neat!", you say, but "How is the stupid computer going to know what the replacement value ought to be?" MailMerge supports three methods of getting the information in the right place at the right time. One way is to set the value of the replacement statements at the beginning of the file (the .sv command). If you use this method you will have to re-edit the document each time you want it to print differently, but instead of having to run through with a global search and replace, and then having to mess around reforming the text, you can just edit the list at the beginning of the file and MailMerge will take care of the rest.

The second alternative is to have the computer ask you the values while the printer routine is engaged (.av command). In this mode, you will be prompted to type in from the keyboard the values you want assigned during the print routine. This is very convenient, because you can print out a unique document without ever going into the edit mode. Suppose that you have a document that occasionally needs to be printed with only a few words changed. You can go directly to the print mode; the printer will pause for a moment and you will be prompted on the screen to type in the values of the code words. Once that is done, the printer will proceed directly to complete the print.

The third and most exciting way you can get the values assigned is by creating a granddaddy list of all the values (call it a data file, if you wish) that you

ever intend to assign to those code words, save it on a disk and then have MailMerge read the list (.rv command) and merge it with your document.

Suppose that you have a form letter to produce. Suppose further that you have six different variables in your form letter. Now suppose that the computer would read six variables off your list (data.file), substitute them for the code words, print the letter, then go back and read another six variables off the list and print the letter again until the list was exhausted. This is how MailMerge works and why it is so nifty for those personalized form letters. You need only the name and addresses in an ASCII type file (one of those that is readable if you use a 'type' command in CP/M-80), and you are in business. Technically speaking OPEN (.df) and READ (.rv) are inserted in the text file near the beginning; when these are processed a flag is set during the print routine causing the text file to be repeatedly processed without resetting the file pointer in the data file, until the end of file condition is detected. The logic is illustrated by the short BASIC routine in Figure 1.

What Do I Use For a Data File?

The answer is, as I said above, any ASCII data file. Your favorite name and address program is, of course, a good candidate. Otherwise you can simply type up a list using WordStar. Appropriate lists can usually be generated out of your accounting system, but the services of a semi-skilled hacker who can write a short utility to read through your customer list, sift out the

rubbish and write a dedicated data file, will often be helpful.

MailMerge is completely lacking in any ability to sift, sort, select or randomly access a data file. Likewise there is no method of generating substrings out of the string variables. Thus, if you intend to print only a portion of your master list you will have to use some utility to generate a worklist containing only those names before invoking MailMerge.

The mailing list need not even be on the same drive as the master letter, but, of course, it may be. MailMerge, itself, is an 8k overlay which must be on either the logged in drive or drive A. You may not edit or use any other WordStar functions while MailMerge is in use.

There are also a number of other minor features of MailMerge which won't be mentioned here; I doubt if any of them alone would justify the purchase of the program.

Matching Hardware Important

As so far depicted, the logic of the MailMerge routine is that of an infinite print loop limited only by the size of the data file being read. The management of a mailing list of several thousand names is well within the capability of today's microcomputers and this program permits you to write a personalized computer letter to each of them. However, in all probability the people who designed the printer you bought may not have engineered it on the assumption that you were going to use it 8 or 10 hours a day non-stop! MailMerge has the potential for setting

Figure 1

```
(IN CB-80)                                     (MAILMERGE equivalent)
TRUE%=1
IF END # 1 THEN QUIT
OPEN "B:NAMELIST.DAT" AS 1                       .DF B:NAMELIST
WHILE TRUE%=1
  READ # 1;NAME$,ADDRESS$                         .RV NAME,ADDRESS
  PRINT NAME$                                     &NAME&
  PRINT ADDRESS$                                 &ADDRESS&
  PRINT "rest of letter"                          rest of letter
WEND
QUIT: STOP
```

(continued next page)

up a print loop which will literally beat your printer to death. Every printer, while in operation, tends to build up heat in critical 'hot spots'. This is the nature of any mechanical (or electronic) contraption. Cooling fans, heat sinks, ventilation, and non-use are all acknowledged methods of controlling the heat problem. For the same reason that your car may or may not survive if you try to drive it at full throttle all day long, your printer may not survive either. In evaluating printers for this application the magic words are 'duty cycle rating'. Candid information on which printers have what duty cycle rating is hard to come by, but sometimes the warranty cards will contain some cryptic message such as "If you use this printer the warranty is void!" Such language hints at a less than 100% duty cycle rating. (*Editor's Note:* Carl Warren's *Pipeline* column will be devoted this month and next month to the array of printers on the market.)

Enough said, but remember that an error message to the effect that your printer is about to become a plastic blob (due to core melt down) is not included with the program.

Bugs

MailMerge would appear to be reasonably free of bugs. Some time ago it was

reported that there was a problem with the .av command when multiple copies were being made. It seems that with some equipment under some circumstances the last word or so of the text gets dropped. I also recently had an unconfirmed report of a reluctance to print four up labels with version 3 on Radio Shack equipment. On my own part, however, regular use for almost two years has not revealed any misbehavior worth reporting. The biggest single annoyance is the lack of a convenient escape routine from the print time input (.av) routines. The documented method is to complete the sequence and then quickly hit the print pause key, before the printer can get started. Although this procedure works if you are quick enough, I miss the boat about half the time and wind up wasting a sheet of paper.

Documentation

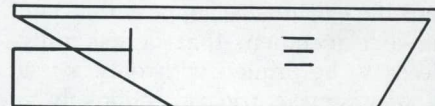
The documentation for MailMerge is extensive. It consists of over 50 pages of written matter, with numerous examples and several working examples in media form on the furnished disk. Documentation should be rated as good.

Conclusions

If you have the urge to take a form letter and a data file (usually a name and

address list) and have the list and the letter form interact in such a way so as to generate as many letters as there are names and addresses, MailMerge is a good solution. Furthermore, if your application involves working with some standard forms which, although they may only be printed one at a time are prepared by global search and replace routines, you may be able to get some enhanced productivity out of the MailMerge option. On the other hand, if your use of WordStar consists mainly of creating unique documents mostly from scratch, or from very heavily edited prior work, MailMerge is obviously of little value.

MailMerge is not menu-driven but is driven by a series of dot commands which must be inserted in the text file. If the user has the skill sufficient to write a short BASIC program to open, read and print a data file, use of MailMerge will be a breeze. On the other hand, those who can't tell a data file from a text file had better be fast learners if they want to make MailMerge work. The ability to blend a text file with rudimentary data file processing insures great flexibility in the hands of a talented user and a recursive mess in the hands of a rank novice.



Opinion

Why Standards On Operating Systems Interfaces?

Jack Cowan

The computer industry in general, and the software portion of that industry specifically, has expended intense effort during the 1970's developing systems which are portable across hardware environments. The initial efforts were in the area of language standardization. It was found that the same language implemented across multiple hardware environments was not consistent. This diversity was in part created by software designers taking maximum advantage of a specific hardware environment. However, to a great extent, the differences within a

particular language implementation were generated by the fact that software designers were designing not for a specific piece of hardware but for a virtual machine environment which was defined by the operating system and the hardware in concert with each other.

Unfortunately, this problem has become even more significant with the growth of the third party software industry and the proliferation of microprocessors. For most of the 1970's, the growth of microprocessor software was near homogenous, because of the

consistent hardware environment for which the software was targeted. However, the growth of microprocessors being witnessed today has created a great amount of diversity within the software community. Specifically, vendor-written operating systems to provide increased functionality and ease of use on a particular microprocessor have created a virtual machine environment which must be interfaced by software. It is clear that a particular software vendor has a very large task to provide a consistent software product across multiple operating systems

and hardware environments. The vendors are also caught in this quagmire, as they need software to leverage their products.

Is There A Solution?

It has become obvious that a standardization effort is needed in the areas of operating systems interfaces. It is not clear that this effort could have been accomplished until now, due to the general adhoc method of operating systems design utilized until just the last few years. Operating systems designers still do not agree on all methodologies; but there are large areas in which agreement can be reached. Because these areas of agreement do exist, it is clear that operating system interface standards can be defined. A recent effort at the Lawrence Berkeley Laboratory has demonstrated that this concept of virtualizing the operating system interface is possible. The Berkeley effort found, "that under certain conditions a uniform system interface can be provided across machine boundaries without disturbing vendor software. The method is by creating a virtual operating system."

An operating system is simply a collection of capabilities or functions. Assuming that major functional areas could be selected and the interface to these major functional areas be standardized, a higher level program could then be written to interface to these standard functional interfaces. For example, a language translator or run time environment can be designed to be consistent with the functions for:

- Process Management
- Support
- Data Transfer Support
- Environment Support
- Data Management Support

and be assured of executing upon any operating system that has implemented the standard interface for these functions. The benefits of this scenario are obvious. A compiler could be developed which would be totally portable across not only hardware environments but the virtual environment also. Third party software vendors could develop a product once, not every time they desire to approach a new hardware or operating system environment. End users could design

their products for one environment and move to another without undergoing catastrophic results. In this age of 8-bit, 16-bit and 32-bit microprocessors, the ability to design a solution as nearly independent of hardware and operating systems software would be a fantastic hedge against "picking" the wrong hardware. More importantly, a software package written for execution on a 8-bit environment could be easily moved to a 16-bit processor when system growth dictated.

So What Is Being Done?

Under the Auspices of the IEEE software standards effort, a new standards group defined as a Microcomputer Operating Systems Interface (MOSI) has been formed. The goal of this new standards group is to define a set of standard interface specifications for microprocessor operating systems. The scope of the group is limited to the microprocessor environment; however, it is fairly obvious that most operating system functions and technology span the entire computer community rather than being strictly limited to a computer type. The progress made by the MOSI group has been significant considering its very short time in existence. The general concept of the standardization definition has taken the form of a series of interface layers. Drawing upon the effort within the local network community, it was noted early on that if an operating system is truly a virtual extension of a piece of hardware, then we can think of an operating system as a series of layer functions beginning immediately above that hardware and extending outward. By beginning the standardization effort at the layer closest to the hardware and extending the standardization effort to those layers furthest away from the hardware, all functions of an operating

system can be encompassed. In fact, any layer can be standardized independently of the other layers as long as a global model has been pre-defined.

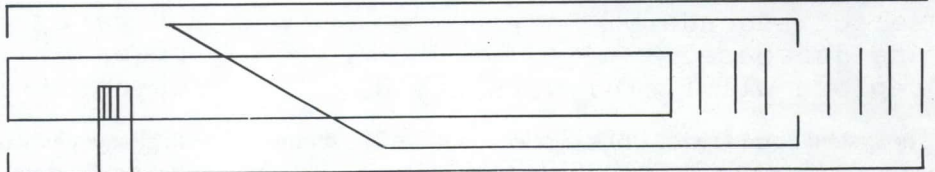
An early attempt at a standard interface which supports language processors by the MOSI group, demonstrated that a homogenous interface can be designed which has no hardware dependence. Specifically this initial standardization effort has constricted itself to seven capability areas:

- Memory Management
- Exception Processing
- Program-operator interface
- Process Management
- System Clock Management
- Data Transfer (input/output)
- Data Management

MOSI is currently structuring itself to approach the more robust solution of the entire operating system and anticipates expansion of the initial draft standard by early 1982. It is obvious that each capability lends itself to multiple layers of standardization. This layering approach will allow the most obvious levels of the interface to be defined and standardized while the more obscure interfaces are researched. In this manner it is hoped that the primary interfaces may, in fact, be standardized in 1982. This layering is actually a misnomer. In reality, the capability areas of an operating system may or may not be layered. The objective of approaching the interfaces in this modular fashion is to allow differentiation across the operating system functions as well as allow an operating system implementor to support the level of the operating system. In this manner the operating system designer can select the level of the operating system interface that best suits his product and gain the benefits associated with that standardization.

OOPS!

In the May issue, on page 31, column 3, there is an error in Ward Christensen's 8080 tutorial (pointed out by Robert Minnihan of Minneapolis). In the middle of the column is a listing ending in the line: "RET". In the second line above "RET" and the fourth line above "RET", "MOVE" should be changed to read "MV2".



Opinion Letters

May 6, 1982

Dear Sirs,

I read with interest Bob Kowitz's article "Printer or Console from BASIC-80". I wanted to point out that a much cleaner way exists for users of CP/M, *provided that your CP/M version has implemented the optional IOBYTE*. IOBYTE is a single byte in low memory that can be used to set the logical to physical device mapping (see Digital Research's "CP/M 2.0 ALTERATION GUIDE", pages 15-16).

If you wish to redirect LPRINT output to the console, all you need to do is reset the LST: fields of the IOBYTE to specify the console. Then, since LPRINT writes to the logical LST: device, that output will go to the console instead of the printer. Assuming that your CP/M is a standard ORG 0 version, IOBYTE is located at memory location 3.

I have attached a short listing of a program that will demonstrate this technique. You should be sure to restore IOBYTE at the end of your program so that subsequent programs will work correctly. But should you forget, the CP/M STAT command also can be used to reset IOBYTE. One other point - this method will work for the BASIC-80 interpreter and for the BASCOM compiler.

If you have access to "REMark" (the magazine of the Health Users Group), you may be interested in the April 1982 issue where Pat Swayne discussed a similar method. In that same article, he presented several extensions to BASIC-E.

```
10 ' Save Default IOBYTE for Printer
20 PRINTER = PEEK(3)
30 ' Define New IOBYTE for Console
40 CONSOLE = PRINTER AND 63 OR 64
50 '
60 ' At this point, you could ask the user
70 ' "Which device (Printer or Console) ?"
80 ' and POKE the desired value.
90 ' But, for this example, let's just
100 ' do it to see how it works.
110 '
120 LPRINT "THIS LINE GOES TO THE PRINTER"
130 ' Switch to the Console
```

```
140 POKE 3, CONSOLE
150 LPRINT "THIS LINE GOES TO THE CONSOLE"
160 ' Switch to the Printer
170 POKE 3, PRINTER
180 LPRINT "BACK TO THE PRINTER AGAIN"
190 END
```

Sincerely,
William R. Brandoni
Willoughby, Ohio

May 12, 1982

Dear Editor-in-Chief,

I found Bob Kowitz's article "PRINTER OR CONSOLE FROM BASIC 80" (*Lifelines* Apr. 82) very interesting.

I tried to use this on VECTOR GRAPHIC System B with BASIC vers.5.2. Unfortunately, it failed to work. Some alterations should be made on the original program in order to obtain the right values for C1, C2, P1, F1 and F2. These right values are:

127, 215, 163, 215, 19503 and 19504

and they permit to switch the output to console or to printer, by properly inserting in the programs the following lines:

```
XXXX POKE 19503, 163 ' PRINTER ON-SCREEN OFF
YYYY POKE 19503, 127 ' PRINTER OFF-SCREEN ON
```

Hereafter are the alterations to be made in order to obtain the right values:

1. Line No.585
FOR I=16600 TO 19999
2. Line No.1010
GOSUB 555 'GET PRINTER AND SCREEN BASIC LOCATIONS(F1,F2)

I remain,
Sincerely yours,
Robert Valent
Consulting Engineer
Saint-Cloud, France

Software Notes

Bugs And Anomalies In dBASE II, Version 2.3

Michael Olfe

The following sequence bombs the system: (Field1 and field2 are fields in database1, Field3 and field4 are fields in database2)

```
Set linkage on
use database1
select secondary
use database2
replace all field1 with field3
```

The system bombs (with a BDOS select on drive "J", on my system) before finishing the first "REPLACE".

The "SET CARRY ON" directive carries over the fields as they were BEFORE a replace. Thus, this command does not work in the very typical situation where you are appending blanks, getting values and doing replaces inside an input loop. For example,

```
Set carry on
append blank
replace fieldone with field two
```

will always carry over blanks to the next record, regardless of what replaced the blank.

New

Products

The software described below is available from the authors, computer stores, software distributors and software publishers. These descriptions are based on information supplied by the software authors, and are not the result of *Lifelines/The Software Magazine* testing; no claims are made by *Lifelines/The Software Magazine* as to the veracity of author's descriptions, and we can accept no responsibility for them.

BASIC/Z System/z, inc.

For this product, CP/M-80 version 2.x and a console device with addressable cursor are required.

This new BASIC supports I/O at source code level. Features include absolute cursor addressing for CRT and printer, reverse video, blinking fields, erase to end-line/screen, screen-oriented editing of console input at run time, non-destructive cursor movement, character and line deletion, insert change modes, dynamic character count and clear screen.

Floating point numerics have a range of $1E-61$ to $1E+61$ with a precision from six to eighteen digits. A program designates the precision required, and BASIC/Z allocates the number of bytes for each variable. Floating point math is performed in decimal (BCD); BCD integers are also provided, with six to eighteen digit precision. Also included are one and two byte control binary types, with functions for insertion or extraction of strings from control arrays. String space is allocated statically, and arrays may be dimensioned dynamically, and erased to reclaim memory space.

Sequential files with read/write capability are supported, so that extension or truncation can occur without the need to copy the whole file. Two forms of random files are also provided, with sequential and/or random access. Nu-

meric data may be accessed in ASCII or in internal binary/BCD format. An entire array, or a selected portion, may be read or written with one statement. Blocking and de-blocking of logical records from masks constructed at run time is possible.

BASIC/Z generates executable object code, and has a serial function to limit program execution to a single system.

A search feature is included to scan an array to match an expression using any rational operator; sorting is also supported. Nested constructs, recursive user-defined functions with multiple arguments and unlimited numbers of statements are also part of the package. BDOS errors are trapped, and PUSH/POP is supported. Dynamic function calls allow up to four expressions to be passed as arguments.

An editor is included, with such features as global search and change, fifteen local edit commands, and syntax testing while typing. A debugging facility allows line trace, error line retention, and the ability to single step a compiled program with continuous display of selected variables.

BRIDOS Systems Design International

This multi-user operating system is a processor-independent system for application programs; all processors can have access to the same files at the same time. The system can be used with either synchronous or asynchronous hardware for communications.

File management in BRIDOS is centralized, so a smaller quantity of information must be communicated; hence low-speed communications can be utilized (like some modem telephone lines).

Each work area is independent, and served by its own processor. In this way the designers hope to greatly speed data processing and make queuing unnecessary.

Applications and file management software being separate, a program fault cannot crash information registers or interfere with other programs. Index files and data files are simultaneously updated, lessening the chance of crashed files.

The Coach Open Systems, Inc.

The Coach is written in Business BASIC and runs under CP/M-80 and MP/M-80 operating systems. Its purpose is to allow a user to create a comprehensive individualized learning program with English language commands. Frames of subject matter text, instructions, and practice situations can be presented to the student. Multiple choice quizzes can be developed using The Coach. On each learning segment, the learner can be routed on up to 36 choice paths.

An option allows the "teacher" to set response wait time for the individual student's learning speed. Operator file names allow the learner to start where he or she left off in an instruction sequence, and frames can be personalized for individual users. Multiple operators can learn at their own individual rates.

Menus of subject matter can be built for easy user access. A special program allows the printing and auditing of newly-created learning programs. Ten lessons are supplied with the product.

CP/M Disk Utilities Ficom, Inc.

These twelve utilities are supplied for Z80 systems running under CP/M-80, version 2.2. A special version is available for the Osborne 1 computer. The author indicates that these are not re-named public domain programs.

VERSION.COM appends a version number to any file, to keep track of program development. In addition, this utility sets file indicators or attributes and allows a protected file to be deleted without first resetting it to R/W. This utility can also display the size of files; the actual file size, rather than the space allocated on disk, is shown.

FDIR.COM is an extended disk directory utility, presenting a four column alphabetical listing of files and their sizes. A command option will display *all* files, not just DIR files. This utility also indicates whether a disk is single or double density, and shows the amount of space remaining.

DIRU.COM is similar to the
(continued next page)

FDIR.COM, but is employed in systems that subdivide files into different user numbers. It can be used in hard disk environments; read only and system files are flagged, as with **FDIR**. The space remaining on a disk is computed based on all user numbers.

VCOPY.COM utilizes sequential and random access operation to provide disk-to-disk file copying. Copied files are verified byte-by-byte. The utility will run on any size system, with all available RAM used for read and write file buffers. Disks may be changed at any time.

FCOPY.COM is similar, but is faster because it does not verify and all RAM is used for the write buffer.

UCOPY.COM is the same as above, but for multiple user number systems.

CMPAR.COM compares files of the same name on different drives. The files are compared byte-by-byte; when a difference is detected, a 16 byte segment of the file is displayed, with a pointer at the first different byte.

MLIST.COM supplies paginated listings of ASCII files on the system list device. Each page starts with the file name, version, size and page number printed at the top. The bottom margin is set so that 57 lines are printed on each page. This may be modified, however, by changing the constant at the proper location. Line wraparound is automatic for lines longer than 79 characters, but this constant may be changed. Tabs are expanded normally and control characters are represented with the "↑" flag.

MLIST-NF.COM is identical, but is used with printers that lack a form feed function.

READ.COM is like **MLIST**, but the file is displayed on the CRT instead of on a list device.

TITLE.COM provides a banner type large block letter display for a file title page.

FMENU.COM produces a numbered display of all COM files in the current directory. When the number of the selected program is entered, it will be automatically loaded and executed.

FORTH-79 MicroMotion

This implementation of the 79-Standard FORTH language runs on Z80 machines under CP/M-80. It is a structured language designed for applications software requiring speedy execution: data acquisition, process control, animation, and video games, for example. The interpreter, compiler, assembler, editor and operating system are all co-resident.

Two versions of the product are available, one for CP/M-80 1.4 (supporting CDOS) and one for CP/M-80 2.x. Certain features are designed to aid the writing of transportable software. One feature pinpoints words in a user program which call a non-standard word, such as an assembly language-coded one.

A user configurable screen editor and a Z80 assembler with Zilog-like syntax are intended to make this product more efficient. The assembler allows local labels to be defined within an assembly language code word definition and includes error testing on operand constructions. Editor configuration definitions are supplied for various terminals and video display boards. A full set of double-number extensions and a string handling word sets are included with **FORTH-79**. Executable FORTH words permit access to BDOS and BIOS function calls, to vector the character input and output streams to devices other than the console, and to vector disk I/O for screen storage to any device (via a user-supplied driver). Most CP/M-80 special control characters are available during FORTH interpreter input, but these may be switched off. Keyboard input can be upper and lower case, or forced to all upper case.

The screen set resides on the disk and in the directory as a normal file. A stand-alone screen manipulator utility is provided with the product; it permits the creation of new screen files, or the transfer of groups of screens to different locations within a file or to different screen fields on any BIOS-supported drive.

An optional enhancement disk contains a floating point arithmetic extension. It uses the 9511 floating point format and has optimized Z80 code for the low-level functions. Trigonometric and transcendental functions are included,

along with exponentials, logarithms, square root, floating point number format and conversion to and from integer formats.

The Incredible Text Printer Datamed Research, Inc.

This product is available for the UCSD Pascal operating system, but does not require that the user have knowledge of UCSD Pascal. 56K to 60K of system memory is required for use of The Incredible Text Printer; two or more floppy disk drives and a 24 line by 80 character CRT are also needed. An Apple II with 80-column board or external terminal can run this program.

This word processing program works with a text editor to produce letters, reports, manuscripts, and offset masters. A setup section provides menus to allow speedy selection of a printing format. Formats may be saved for recall via a single keystroke. The user may press a help key for further aid. The design philosophy behind this formatter is based on the idea that format changes can be made without editing the text itself. Centering, margins, indents and other format features are isolated in the format setup. (However, underlining, boldface and other features are supported by instructions imbedded within the text.)

ITP can be used with line printers, dot matrix printers, or typewriter printers, but is most thoroughly exploited by daisy wheel or thimble printers. Line and paragraph indents are followed automatically, and tabular columns align properly with proportional fonts.

Under user control are headings, sub-headings, footings, centered and left and right justified fields, alternating headings on even numbered pages, swapping right and lefthand fields. Files may be inserted from within other files, and indices and tables of contents may be created automatically. Page numbers, chapter, sections and subsection numbers may be added to headings, footing, indices, or tables of contents. Where applicable, print styles may be selected for them.

The user may elect to declare tables and figures which should be floated to the bottom of a page or the top of the next page, depending on space available. Name and address files may be used to

generate personalized form letters. Information can be keyed during printing.

macroP
Pluto Research Group

This macroprocessor is for computers running IBM PC-DOS, CP/M-80, and CP/M-86. It permits the addition of commands to document formatters and language processors. Simple assemblers can be converted into macro assemblers for the assembly language programmer. For high level programming, macroP adds indefinitely long variable names, compile-time expression evaluation, conditional compilation, and time and date stamping of source programs.

Conditional nested inclusion of text files and automatic numbering of section headings are supported. With the program are supplied files of macros suited to BASIC, Pascal and PL/I-80 programming. 56K of main memory is required.

Math ★
Force 2, Ltd.

This program allows users of WordStar to perform arithmetic calculations while editing a document. It serves much the same purpose as an adding machine, supporting addition, subtraction, multiplication, and division. However, numbers of up to nineteen digits may be entered; oversized numbers are automatically reduced and rounded where possible. Equations may be entered in columns or on one line. Commas, decimal, dollar signs and parentheses are permitted. The distribution disk includes examples and an INSTALL program for users with CP/M 2.0.

Math ★ requires a Z80 microprocessor and WordStar 3.0 (uncustomized).

Precision BASIC
Epdel Systems

This BASIC is specially designed for computational accuracy; the precision of computation can be adjusted up to one hundred decimal digits. For faster calculations, precision can be limited to as few as six digits. Interval arithmetic error monitoring is employed so that all results may be tested for accuracy before printing.

Written in Z80 assembly language, Precision BASIC requires 30K RAM and CP/M-80.

Priorities
Big Island Computer Systems, Inc.

This time allocation product runs on 64K CP/M-80 systems and is available on 8-inch single density diskettes. It is designed to serve up to ten professionals, and is intended to target the user's attention on tasks scheduled for each day. The user receives a daily report organized for a mix of appointments and prioritized tasks. The user may mark tasks for re-scheduling later; task priority levels and the amount of warning given for a task can be adjusted.

Dates, appointment times, priority, detail (twenty characters), description (sixty characters), cycle (day, week, fortnight, month or year) and estimated hours (or tenths of an hour) for completion can be entered. Appointments can be overlapped, and tasks given priority from one to 99.

SOFTCOM
The Software Store

This communications program runs under CP/M-80 and requires a serial port not used by the console device. It can provide a terminal emulator to connect the console and modem. Conversations with other computers can be copied to disk files or to the printer. Files can be moved between machines with incompatible disks. Files can be transferred and 19,200 baud. With the use of a modem, speed is limited to 1200 baud. Eight-bit characters can be used, so that ASCII and binary files can be transferred.

Text files can be sent to full or half duplex systems not running SOFTCOM, with or without waiting for a response. Line feeds following carriage returns may be filtered out.

SOFTCOM can be used unattended, being compatible with SUBMIT and XSUB.

UltraCalc
Lattice, Inc.

This electronic spread sheet package is designed for CP/M-80 and UNIX envi-

ronments. The "sheet" contains 255 rows and 64 columns, and ULTRACALC includes such operations as insert, delete, move, replicate, edit, titles, multiple windows, disk save and restore, printer output, copy file, erase file, display disk directory, set the default drive. ULTRACALC's algebraic expression analyzer supports eleven operators and twenty-two functions.

Files contain only ASCII characters and can be processed with other programs. Printer output can be sent to a disk file in a format acceptable to WordStar.

A menu-driven customization mode permits the user to select a terminal from those standard terminals supported, and to provide for terminals not in ULTRACALC's repertoire. Screen layout, printer characteristics and default option settings can also be changed using the customization mode.

New

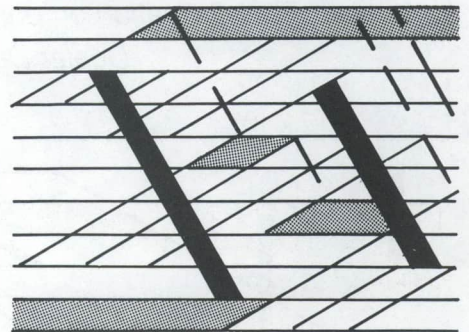
Versions

ASCOM
Version 2.02

This update corrects an error in the ACCEPT and IGNORE commands; hexadecimal values greater than 80H are now permitted. The error applied only to the 8080 version.

PAS-3 Medical
PAS-3 Dental
Version 1.79 and Version 1.66

These new versions implement the same changes. When patients are deleted from the files PAS-3 now automatically loads and executes the routine to rebuild the Hash Tables.



Software Notes

Macros of the Month

Edited by Michael Olfe

Since we are not drowning in macro submissions this month (i.e., there aren't any), I'll take some cues from Ward Christensen's PMATE review in the May *Lifelines/The Software Magazine*. The next version is to have a repeat key, auto-indent, and a simpler configuration scheme, as well as the features which have already been added to the MSDOS, CP/M-86, and PCDOS versions (see last month's column). Ward also mentioned it would be nice to be able to see what is "tagged", a la Wordstar, via reverse video. Can anyone send in a macro or IOPATCH for IOPATCH.ASM which would accomplish this?

There were two errors in last month's "Macros" column. My apologies to Mr. Andrew Hughes of Toronto, who was falsely accused there of being responsible for *all* the macros in the column. In fact, he wrote only the first, a macro to do repetitive processing of all the files on a disc. There were also some typos in the auto-loading macro. If you thought the following macro tidbit odd-looking, give yourself 5 points for having learned to think like a software product:

```
@1="0{.0$^[^}  
@1="1{.1$^[^}  
@1="2{.2$^[^}  
@1="3{.3$^[^}
```

(etc., etc.)

But if you had to type it in and execute it to realize that ↑ is really and should be a \$ (ESC), you get no points.

Laughlines

backup *n. & v. & adv.* 1 *n.* Any file, device, or person which results from backing up; the total deviance from the original is directly proportional to the number and scale of the catastrophes resulting from each copying or matching error. 2 *v.intrans.* To compound errors while merely trying to perpetuate them. 3 *v.trans.* To risk (a file, program) by attempting to copy it. 4 *v.trans.* (Of a programmer, engineer) to specify someone unacquainted with the system, job, and user. See also STANDBY. 5 *adv.* Annoyingly, as: "That salesman really got my backup."

— From *The Devil's DP Dictionary* by Stan Kelly-Bootle. Copyright (c) 1981 by McGraw-Hill, Inc. Used with permission of McGraw-Hill Book Company.

Attention Dealers!

There are a lot of reasons why you should be carrying Lifelines/The Software Magazine in your store. To provide the fullest possible service to your customers, you must make this unique publication available. It will keep them up to date on the changing world of software: on updates, new products, and techniques that will help them use the packages you sell. Lifelines can back up the guidance you give your customers, with solid facts on the capabilities of different products and their suitability to a variety of situations. Now we can also offer you an index of all back issues of Lifelines, opening up a full library of information for you and your customers.

For information on our dealer package, call (212) 722-1700, or write to Lifelines Dealer Dept., 1651 Third Ave., New York, N.Y. 10028.

Meet Lifelines™ /The Software Magazine™ and join the serious microcomputer software users who turn to us every month for timely, important software news.

and



Engineers,




executives



professionals

business people read our reports on newly-discovered bugs,

patches to overcome bugs or enhance software value,  new products, and the changes in new versions. And these aren't the only ways we stretch our readers' software dollars: in-depth, comparative reviews really tell them which products are most suited to their needs.



Programmers and



applications designers read Lifelines/

The Software Magazine to stay in touch. Like the



computer

and software dealers, the



consultants, and



professors who lean

on Lifelines/The Software Magazine, they have to know the latest on operating systems, programming tips and new trends in software – so their clients and customers are well-served.



Hobbyists keep

up with The CP/M®Users Group and the largest library of public domain software in the world. If these people sound like you, join the software community. Subscribe to Lifelines/The Software Magazine.

To order, please print below:

Name _____

Company _____

Address _____

City _____ State _____ Zip _____

U.S., Can., Mexico

1 yr. \$24 (12 issues)

2 yrs. \$40

Foreign Rates

1 yr. \$50 (12 issues)

2 yrs. \$84

VISA MasterCard

Signature _____

Card Number _____

Orders must be pre-paid; checks must be in \$US, drawn on a U.S. bank.



Lifelines, The Software Magazine are trademarks of Lifelines Publishing Corp. CP/M is a registered trademark of Digital Research, Inc. The CP/M Users Group is not affiliated with Digital Research, Inc. Copyright©1982 by Lifelines Publishing Corporation. This advertisement was created by DocuSet(SM).

(continued next page)

VERSION LIST

June 4, 1982

The listed software is available from the authors, computer stores distributors, and publishers. Except in the cases noted, all software requires CP/M-80, SB-80, or compatible operating systems.

S Standard Version
P Processor. Products marked Z80 work on Z80 only; those designated 8080 work on Z80 also.

MR Memory Required. This, where applicable, refers to the size of the CP/M-80 required. Starred (*) entries refer to the Transient Program Area required; it varies with the RAM available in the computer.

New Products and new versions are listed in boldface.

Product	S	P	MR
ACCESS-80	1.0	8080	54K
Accounts Payable/Cybernetics			Needs RM/COBOL. Runs w/CP/M-80, OASIS, UNIX
Accounts Payable/MC	1.0	8080	56K
Accounts Payable/Structured Sys	1.3B	8080	45K*
Accounts Payable/Osborne/McGraw-Hill	2.1	8080	48K
Accounts Payable/Peachtree	07-13-80		48K
Accounting Plus		8080	64K
Accounts Receivable/Cybernetics			Needs RM/COBOL. Runs w/CP/M-80, OASIS, UNIX
Accounts Receivable/MC	1.0	8080	56K
Accounts Receivable/Osborne/McGraw-Hill	2.1	8080	48K
Accounts Receivable/Peachtree	07-13-80	8080	48K
Accounts Receivable/Structured Sys	1.4C	8080	49K*
Address Management System	1.0	8080	
ALGOL 60	4.8C	8080	24K
ANALYST	2.0	8080	52K
Angel		8080	42K*
APL/V80	3.2	Z80	27K*
Apartment Management (Cornwall)	1.0	Z80	48K*
ASCOM	2.02	8080	
ASCOM/86	2.01	8086	
ASM/XITAN	3.11	Z80	
Automated Patient History	1.2	8080	48K
BASIC Compiler	5.3	8080	48K
BASIC-80 Interpreter	5.21	8080	36K*
BASIC Utility Disk	2.0	8080	20K*
BaZic II	03/03	Z80	32K
Benchmark Word Processor	2.2		40K*
Benchmark Mail List	1.1		40K*
BOSS Financial Accounting System	1.08	8080	48K
BOSS Demo	1.08	8080	44K*
BSTAM Communication System	4.5	8080	16K*
BDS C Compiler	1.46c	8080	46K*
Whitesmiths' C Compiler	2.1	8080	56K*
BSTMS	1.2	8080	24K
BUG/uBUG Debuggers	3.20	Z80	24K*
CBASIC2 Compiler	2.08	8080	24K*
CBS Applications Builder	1.33	8080	48K
CIS COBOL Compiler	4.4,1	8080	43K*
CIS COBOL Compact	3.46	8080	26K*
FORMS 1 CIS COBOL Form Generator	1.06	8080	
FORMS 2 CIS COBOL Form Generator	1.1,6a	8080	43K*
Interface for Mits Q70 Printer			CP/M-80 1.41 or 2.XX
COBOL-80 Compiler	4.6	8080	48K
COBOL-80 PLUS M/SORT	4.01	8080	48K
CONDOR II	2.06	8080	42K*
CREAM (Real Estate Acct'ng)	2.3	8080	64K
Crosstalk	1.4	Z80	
DATASTAR Information Manager	1.101	8080	34K*
Datebook-II	2.04	8080	52K
dBASE-II	2.3B	8080	42K*
dBASE-II Demo	2.3B	8080	42K*
Dental Management System 8000	8.7A	8080	48K
Dental Management System 9000 & Demo	2.03	8080	48K
DESPOOL Print Spooler	2.1A	8080	19K
DISILOG Z80 Disassembler	4.0	Z80	
DISTEL Z80/8080 Disassembler	4.0	8080	
Documate/Plus	1.4	8080	36K*
Documate/Plus/Demo	1.5	8080	36K*
EDIT Text Editor	2.06	Z80	24K*
EDIT-80 Text Editor	2.02	8080	20K*
EM 80/86	1.00	8086	
Emulator-86	1.0	8086	
FABS-I	2.7	8080	32K
FABS II	4.15	8080	48K
FILETRAN	1.20		32K
FILETRAN	1.4		32K
FILETRAN	1.5		32K
FinalWord	1.0	8080	56K
Financial Modeling System	2.0		48K
Formula w/General Accounting System	1.01		
FORTH (Timin)	3.5	8080	28K
FORTTRAN-80 Compiler	3.44	8080	32K*
FPL 56K Vers.	2.6	8080	56K
FPL 48K Vers.	2.6	8080	48K
General Ledger 9000 & Demo	1.06		
General Ledger/Cybernetics			Needs CBASIC
General Ledger/MC	1.0	8080	56K
			Needs RM/COBOL. Runs w/CP/M-80, OASIS, UNIX
			Needs CP/M-80 2.2 or MP/M-80

VERSION LIST

Product	S	P	MR	
General Ledger/Osborne/McGraw-Hill	2.4	8080	48K	Needs CBASIC2
General Ledger/Peachtree	07-13-80	8080	48K	Needs BASIC-80 4.51
General Ledger/Structured Sys	1.4C	8080	45K*	w/It Works Package
GLECTOR Accounting System	2.02	8080	48K*	Use w/CBASIC2, SELECTOR III
GLECTOR IV Accounting System	1.0	8080		Needs SELECTOR IV
GrafTalk	1.0		48K*	Requires 180Kb/drive. Available for certain Terminals
HDBS	1.05A	+	52K	Printers, & Plotters. N/A CDOS.
HOE	2.1	8080	48K	
IBM/CPM	1.1	8080		CP/M 1.4 only
Insurance Agency System 9000 & Demo	1.10	8080		Needs CBASIC
Integrated Acctg Sys/Gen'l Ledger		8080	48K	Needed for 3 pkgs. below
Integrated Acctg Sys/Accts Pyble		8080	48K	
Integrated Acctg Sys/Accts Rcvble		8080	48K	
Integrated Acctg Sys/Payroll		8080	48K	
Interchange		Z80	32K	
Inventory/MicroConsultants	5.3	8080	56K	Needs CP/M-80 2.2
Inventory/Peachtree	07-13-80	8080	48K	Needs BASIC-80 4.51
Inventory/Structured Sys	1.0C	8080	48K*	w/It Works Package
JANUS	1.4.3	8080		Also runs w/8086
Job Cost Control System/MC	1.0	8080	56K	Requires CP/M-80 2.2
JRT Pascal System	1.4	8080	50K*	
Legal Time Acctg Series 9000 & Demo	1.07	8080		Needs CBASIC
LETTERRIGHT Text Editor	1.1B	8080	48K*	
LINKER		Z80		
LP-DISK	1.0	8080	48K	Also for TRS-80 I/III
MAC	2.0A	8080	20K*	
MACRO-80 Macro Assembler Package	3.43	8080		
MAG/base1 (LMS)	2.0.1	8080	56K	Needs CBASIC, 2.06 or later & 180K/drive
MAG/base2 (IMS)	2.0.1	8080	56K	Needs CBASIC, 2.06 or later & 180K/drive
MAG/base3 (ADS)	2.0.1	8080	56K	Needs CBASIC, 2.06 or later & 180K/drive
Magic Typewriter	3	Z80	48K	
Magic Wand	1.11	8080	28K*	
MAG/sam3	4.2	8080	32K	
MAG/sam4	1.1	8080	32K	Needs CBASIC
MAGSORT-C	1.0		12K*	For CBASIC
MAGSORT-M	1.0		12K*	For MBASIC
MAGSORT-R	1.0		12K*	
MAILING ADDRESS Mail List System	07-13-80	8080	48K	For Compilers — BASCOM, FORTRAN-80, PL/I-80
MailMerge	3.0	8080	44K*	
Master Tax	1.0-80	8080	48K	
Matchmaker		8080	32K	
Math ★	3.043	8080		Math add-on to WordStar
MDBS	1.05A	+	44K*	
MDBS-DRS	1.02	+	48K*	
MDBS-QRS	1.0	+	52K	
MDBS-RTL	1.0	+	52K	
Medical Management System 8000	8.7a	8080		Needs CBASIC
Medical Management System 9000	1.10	8080		Needs CBASIC
Medical Management System 9000 Demo	2.03	8080		Needs CBASIC
Microcosm		Z80		CP/M-80 2.X or MP/M-80
MicroLink-80	6.0	8080	35K*	Requires serial port
MicroSEED	B.10G	8080	48K	
Microspell	4.3	8080	48K	
Microspell Demo	1.0	8080		For Dealers Only
Microstat	2.08a	8080	48K	Needs baZic 03/03 or BASIC-80, 5.03 or later
Microstat for Apple	2.0	Z80	48K*	
Mince	2.6	8080	56K	
Mince Demo	2.6	8080	48K	
Mini-Warehouse Mngmt. Sys.	5.5	8080	48K	Needs CBASIC
Money Maestro		8080	48K	CP/M-80 1.4 or 2.2
MP/M-I	1.1			
MP/M-II	2.0	8080	48K	32K RAM needed
Mr. EDit	2.0	8080	24K	Needs 24K TPA, 12 x 64 column terminal
MSORT	1.01	8080	48K	
MuLISP-80/MuSTAR Compiler	2.12	8080	24K	
MuSIMP / MuMATH Package	2.12	8080	48K	muMATH-80
NAD Mail List System	3.0D	8080	48K	
Nevada COBOL	2.1	8080	32K	
Order Entry w/Inventory/Cybernetics		Z80		Needs RM/COBOL
Panel	3.02		44K	
PAS-3 Medical	1.79	8080	56K	Needs 132-col. printer & CBASIC
PAS-3 Dental	1.66	8080	56K	Needs 132-col. printer & CBASIC
PASM Assembler	1.02	Z80		
Pascal/M	4.02	8080	56K	CP/M 2.x only
PASCAL/MT Compiler	3.2	8080	32K	
PASCAL/MT + w/SPP	5.5	8080	52K	Needs 165K/drive
PASCAL/Z Compiler	4.0	Z80	56K	
Payroll w/Cost Acct./Osborne/McGraw-Hill	2.2	8080	48K	Needs CBASIC2
Payroll/Peachtree	07-13-81	8080	48K	Needs BASIC-80 4.51
Payroll/Structured Sys	1.0E	8080	60K	w/It Works run time pkg.
PEARL SD	3.01	8080	56K	w/CBASIC2, ULTRASORT II
PLAN80 Financial Package (Z80/8080)	2.3	8080	56K	Specify Z80/8080
PLAN80 Demo	1.2			
PL/I-80	1.3	8080	48K	
PLINK I Linking Loader	3.28	Z80	24K*	
PLINK-II Linking Loader	1.14	Z80	24K*	

(continued next page)

VERSION LIST

Product	S	P	MR	
PMATE	3.02	8080	24K*	
PMATE-PC	1.05	8086		For the IBM PC
POSTMASTER Mail List System	3.5	8080	48K	
Professional Time Acctg	3.11a	8080	48K	Needs CBASIC2
Programmer's Apprentice	1.2	8080	56K	Needs BASCOM 5.3, 2-251 Kbdrive
Property Management Program (AMC)	4.2	Z80	48K	Needs CBASIC 2.07+, CP/M-80 2.0+
Property Management System	07-13-80	8080		Needs BASIC-80 4.51
PSORT	1.4	8080		N/A-Durango
QSORT Sort Program	2.0	8080	48K	
Quic-N-Easi	1.4	Z80	48K	Also runs on TRS-80 Mod III
Real Estate Acquisition Programs	2.1	8080	56K	Needs CBASIC
Remote	3.01	Z80		
Residential Prop. Mngemt. Sys.	1.0	Z80	48K	
RAID	5.0.2	8080	28K	
RAID w/FPP	5.0.2	8080		
RECLAIM Disk Verification Program	2.1	8080		
Sales Pro	5.0	8080		
SBASIC	5.4a	8080		
Scribble	1.3	8080		
SELECTOR-III-C2 Data Manager	3.24	8080	48K	Needs CBASIC
SELECTOR-IV	2.17	8080	52K	Needs CBASIC
SELECTOR-V	5.0	8080	48K	
Shortax	1.2	Z80	48K	TRSDOS,MDOS too, needs BASIC-80 5.0
SID Symbolic Debugger	1.4	8080		N/A-Superbr'n
Spellguard	2.0	8080	48K	Needs Word Processing Program
STATPAK	2.1	8080		Needs BASIC-80 4.2 or above
STIFF UPPER LISP	2.8	8080		
STRING BIT FORTRAN Routines	1.02	8080		
STRING/80 bit FORTRAN Routines	1.22	8080		
STRING/80 bit Source	1.22	8080		
SUPERSORT I Sort Package	1.5	8080		Max. record=4096 bytes
SELECT		8080	40K	
T/MAKER II	2.6	8080	48K	Avail. for CDOS
T/MAKER II DEMO	2.4	8080	48K	
TEX Text Formatter	2.1	8080	36K	
TEXTWRITER-III	3.6	8080	32K	
TIM-III	3.12	8080	32K	Needs 2 240 Kb drives
TIM-III	3.11	8086		For the IBM PC
TINY C Interpreter	800102C	8080		
TINY C-II Compiler	800201	8080		
Torricelli Author	1.04c	8080	48K	24x80 CRT, 2-100Kb/drive
TRS-80 Customization Disk	1.3C	8080		
ULTRASORT II	4.1C	8080	48K	
UT-86	1.00	8086		Specify operating system: IBM PC/CPM-86/MS-DOS
Lifeboat Unlock	1.3	8080		Use w/BASIC-80 5.2
VISAM	2.3p	8080	40K*	
Wiremaster	3.12	Z80	44K	Needs 180K/drive
WordIndex	3.0	8080	48K	Needs WordStar
WordMaster	1.07A	8080	40K	
WordStar	3.0	8080	48K*	
WordStar IBM PC	3.02M	8086		
WordStar Customization Notes	3.0	8080		
XASM-05 Cross Assembler	1.05	8080	24K	
XASM-09 Cross Assembler	1.07	8080	24K	
XASM-51 Cross Assembler	1.09	8080	24K	
XASM-F8 Cross Assembler	1.04	8080	24K	
XASM-400 Cross Assembler	1.03	8080	24K	
XASM-18 Cross Assembler	1.41	8080	24K	
XASM-48 Cross Assembler	1.62	8080	24K	
XASM-65 Cross Assembler	1.97	8080	24K	
XASM-68 Cross Assembler	2.00	8080	24K	
XYBASIC Extended Interpreter	2.11	8080		
XYBASIC Extended Disk Interpreter	2.11	8080		With EDIT features
XYBASIC Extended Compiler	2.0	8080		Requires the XYBASIC w/EDIT features to create SOURCE
XYBASIC Extended Romable	2.1	8080		
XYBASIC Integer Interpreter	1.7	8080		
XYBASIC Integer Compiler	2.0	8080		
XYBASIC Integer Romable	1.7	8080		
ZAP-80	1.4	8080	24K	Needs 50K/drive
Z80 Development Package	3.5	Z80		N/A-Magnolia, Superbr'n, mod.CP/M-80
ZDM/ZDMZ Debugger	1.2/2.0	Z80		For N'Star, Apple, IBM 8''
ZDT Z80 Debugger	1.41	Z80		N/A-Superbr'n, mod.CP/M-80
ZSID Z80 Debugger	1.4A	Z80		N/A-Superbr'n, mod.CP/M-80

+ These products are available in Z80 or 8080, in the following host language: BASCOM, COBOL-80, FORTRAN-80, PASCAL/M, PASCAL/Z, CIS-COBOL, CBASIC, PL/I-80, BASIC-80 4.51, and BASIC-80 5.xx.

BOY, IS THIS COSTING YOU.

It's really quite basic: time is money.

And BASIC takes a lot more time and costs a lot more money than it should every time you write a new business software package.

Especially when you could speed things up with dBASE II.

dBASE II is a complete applications development package.

Users tell us they've cut the amount of code they write by up to 80% with dBASE II.

Because dBASE II is the high performance relational database management system for micros.

Database and file handling operations are done automatically, so you don't get involved with sets, lists, pointers, or even opening and closing of files.

Instead, you write your code in concepts.

And solve your customers' problems faster and for a lot less than with BASIC (or FORTRAN, COBOL or PL/I).

dBASE II uses English-like commands.

dBASE II uses a structured language to put you in full control of your data handling operations.

It has screen handling facilities for setting up input and output forms.

It has a built-in query facility, including multi-key and sub-field searches, so you can DISPLAY some or all of the data for any conditions you want to apply.

You can UPDATE, MODIFY and REPLACE entire databases or individual characters.

CREATE new databases in minutes, or JOIN databases that already exist.

APPEND new data almost instantly, whether the file has 10 records or tens of thousands.

SORT the data on as many keys as you want. Or INDEX it instead, then FIND whatever you're looking for in seconds, even using floppies.

Organize months worth of data in minutes with the built-in REPORT. Or control every row and column on your CRT and your printer, to format input and output exactly the way you want it.

You can do automatic calculations on fields,



records and entire databases with a few keystrokes, with accuracy to 10 places.

Change your data or your entire database structure without re-entering all your data.

And after you're finished, you can protect all that elegant code with our run-time compiler.

Expand your clientbase with dBASE II.

With dBASE II, you'll write programs a lot faster and a lot more efficiently. You'll be able to write more programs for more clients. Even take on the smaller jobs that were out of the economic question before. Those nice little foot-in-the-database assignments that grow into bigger and better bottom lines.

Your competitors know of this offer.

The price of dBASE II is \$700 but you can try it free for 30 days.

Call for our Dealer Plan and OEM run-time package prices, then take us up on our money-back guarantee. Send us your check and we'll send you a copy of dBASE II that you can exercise on your CP/M® system any way you want for 30 days.

Then send dBASE II back and we'll return all of your money, no questions asked.

During that 30 days, you can find out exactly how much dBASE II can save you, and how much more it lets you do.

But it's only fair to warn you: business programmers don't go back to BASIC's.

Ashton-Tate, 9929 Jefferson, Los Angeles, CA 90230. (213) 204-5570.



Ashton-Tate

©Ashton-Tate 1981

®CP/M is a registered trademark of Digital Research.

Also available from Lifeboat Associates.

LIFELINES™ / The Software Magazine™
1651 Third Avenue, New York, New York 10028



Second Class Postage Paid
At New York, N.Y.