

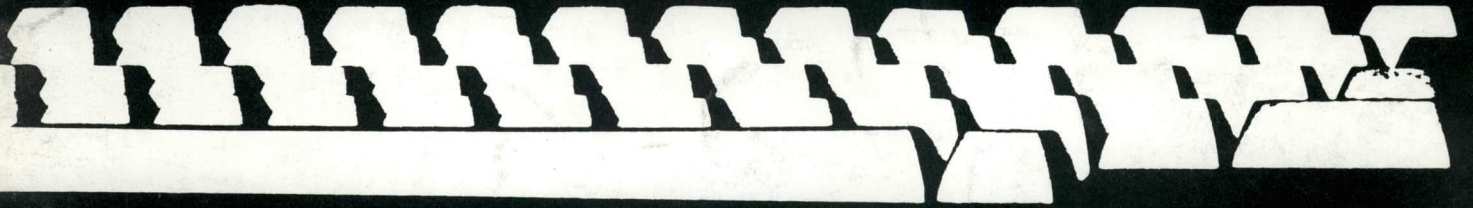
LIFELINES™

The Software Magazine™

\$3.00

June 1982

Volume III, No. 1 (ISSN 0279-2575, USPS 597-830)



Evaluating Application Development Software

Introduction To Microcommunications

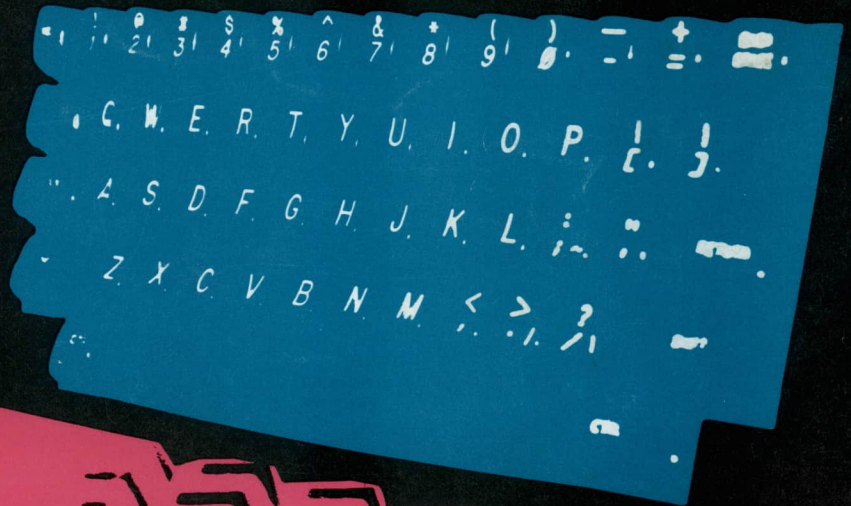
MicroSpell, MicroProof, And SpellGuard

8080 Assembler Tutorial: Subroutines

A Review of FMS-80

CPMUG Volume 81

Using PLAN80



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.

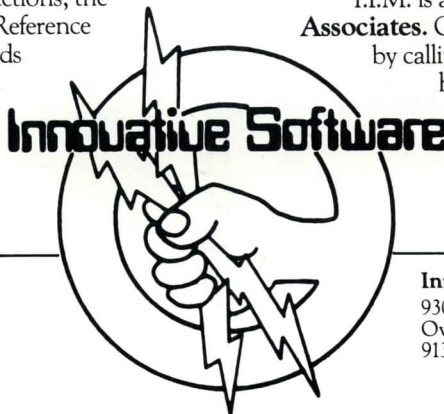
Features of T.I.M.

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.

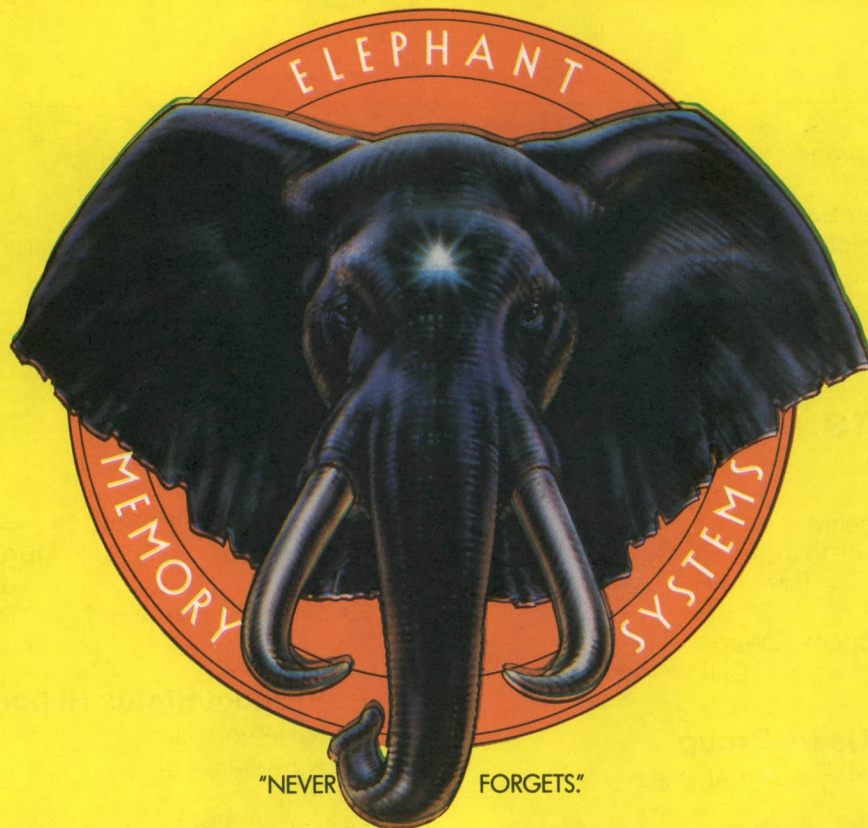


Available for CP/M,* and
IBM PC DOS.**
CP/M version—\$695. IBM PC version—\$495.

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.

LIFELINES

The Software Magazine

June 1982

Volume III, No. 1

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

Design/Production: K. Gartner
Typographer: Harold Black
Cover by K. Gartner
Cover photography by Bruce Weiss

DEPARTMENTS

Opinion

- 6** Editorial Comments
All things come to those who wait
Edward H. Currie
- 7** The Pipeline
Pick Your Modem, Folks
Carl Warren

The CP/M® Users Group

- 27** Volume 81 Catalogue and Abstracts

Software Notes

- 21** Tips & Techniques
- 26** For COBOL-80™ Users
- 28** Macros of The Month
Edited by Michael Olfe
- 37** Patches For MAGSAM™
- 40** Pseudo-Relocatable Subroutines,
Part 2
Gregory A. Knott

- 42** For BSTAM™/BSTMS™ Users
- 50** Modifying Control-C In MBASIC™
Bill Norris
- 53** Notes On dBASE II™, Version 2.3B
Michael Olfe

Product Status Reports

- 51** New Products
- 52** New Versions
- 53** Bugs
- 54** Version List

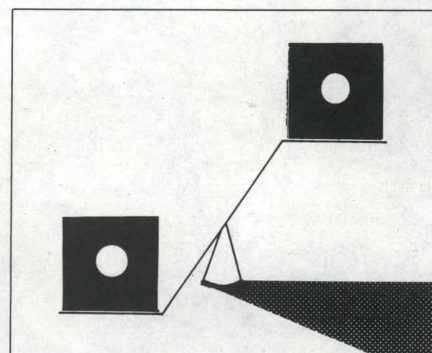
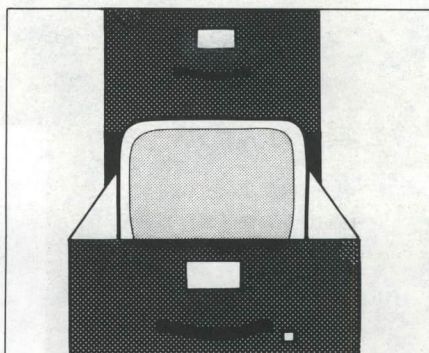
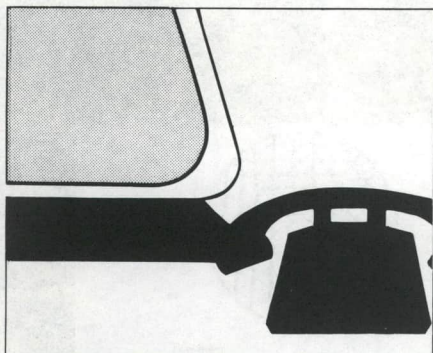
Miscellaneous

- 10** Notice
- 19** KIBITS™
- 20** Renew
- 34** A Call For Manuscripts
- 42** Attention Dealers

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.
BSTAM, BSTMS - TMs Byrom Software.
CB80, CBASIC2, PL/I-80, SID-86 CP/M-86 - 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.
FMS-80 - TM Systems Plus.
KIBITS - TM Bess Garber and Seton Kasmir.
MAGSAM - TM Micro Applications Group.
MicroProof - TM Cornucopia Software.
MicroSpell - TM Bob Lucas.
PLAN80 - TM Business Planning Systems, Inc.
PMATE, PLINK-II - TMs Phoenix Software Associates, Ltd.
SMARTERM - TM Advanced Logic Systems, Inc.
SpellGuard - TM Innovative Software, Inc.
WordStar, SpellStar - TMs MicroPro International Corp.
Z80 - TM Zilog Corporation.
Program names are generally TMs of their authors or owners.



FEATURES

12 A Review of FMS-80™

Mark Rettig

This powerful, menu-driven file management system is evaluated as part of our data base management series.

17 AUTOLOAD For SB-80™, CP/M-86™ And CP/M-80 On The Osborne I Computer

Kelly Smith

Did you know that the AUTOLOAD feature of CP/M-80 also exists in SB-80 and CP/M-86? And a special assembly language programming trick will help you implement it on the Osborne I computer.

20 SMARTERM™ Inverse Video In CP/M-80 For The Apple™

Lou P. Rivas

This article will let you bypass some problems in achieving inverse video with the SMARTERM card, the Z80™ SoftCard™ and your Apple.

23 Criteria For Evaluating Application Development Software

Steve Patchen

"Fourth generation" software is emerging in a wide variety of styles and capabilities. This author has established some standards by which to assess these new products.

30 8080 Assembler Tutorial: Subroutines

Ward Christensen

Subroutines for data movement, arithmetic, logical and input/output are covered; more subroutines will be examined later on.

35 MicroSpell™, MicroProof™, and SpellGuard™

James K. Mills

The best of current spelling checkers are compared (including SpellStar™, which was fully described in last month's issue). This article will help you decide which product best suits your needs.

38 A Detailed Description of PLAN80™, Part 2

Raymond J. Sonoff

PLAN80 can eliminate the drudgery which used to be part of financial modeling. Other financial planning packages will be discussed in future issues.

44 An Introduction To Microcommunications

Davis Foulger

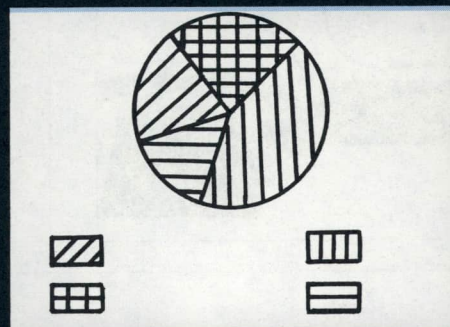
The possibilities for microcommunications are rapidly expanding – in terms of methods, software, and the wealth of information available to the microcomputer user.

Use your desk-top computer to produce sophisticated graphs and charts with a small number of simple, straightforward commands in plain English.

With this easy-to-use graphics software system, even a first-time user can portray data in visual form — pie charts, line graphs, bar graphs, symbol plots, mixed charts, you name it.

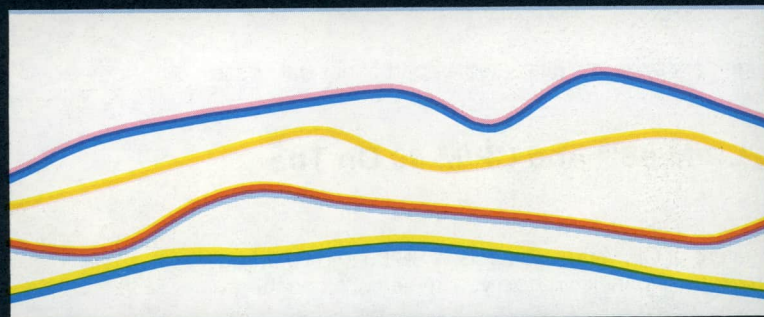
You can display productivity reports, profit trends, budgets, absenteeism, tax outlays, office expenses, and sales projections. Useful for all types of business and scientific applications.

Optional full-color support is available for use with color terminals, color printers and plotters.

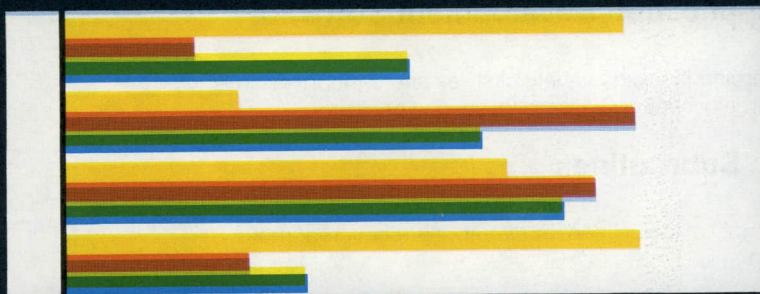


This fully interactive graphics package sacrifices nothing in the way of power and versatility to help communicate even your most complicated presentations.

For more information about GrafTalk and 200 other programs suitable for use in professional, programming, and personal environments, contact:



GrafTalk Speaks Your Mind



Lifeboat Associates, 1651 Third Ave. NY, NY 10028, (212) 860-0300; or TWX: 710-581-2524 (LBSOFT NYK) or Telex: 640693 (LBSOFT NYK). Dealer/Distributor/OEM inquiries invited.

GrafTalk requires a computer using a CP/M[®] 80 compatible operating system as well as a video terminal and one of the wide variety of printers*, plotters** or graphics CRTs. The following is a partial list of supported printers and plotters.

*Printers

Diablo 1640, 1650, 630†
NEC Spinwriter†
The following must be used in conjunction with a graphics terminal, such as Autograph II[®], etc.
Epson with Graffiti[†]
Anadex with R options

**Plotters

HP 7220/1/5
Houston DMP3/4/6/7
Tektronix 4662
Watanabe WX4630/75

Copyright © 1982 by Lifeboat Associates
GrafTalk™ Redding Group, Inc.
CP/M reg.™ Digital Research, Inc.
†Trademarked by the manufacturers noted
The illustrations are artist's representations of graphs produced by GrafTalk and not actual reproductions.
Created and produced by DocuSet(TM)



Lifeboat Associates
1651 Third Avenue
New York, New York 10028
Telephone: (212) 860-0300
TWX: 710-581-2524 (LBSOFT NYK)
Telex: 640693 (LBSOFT NYK)

Lifeboat Associates, Ltd.
PO Box 125
London WC2H 9LU, United Kingdom
Telephone: 01 836 9026
Telex: 893709 (LBSOFTG)

Lifeboat Associates, SARL
70 Avenue D'Argenteuil
92600 Asnieres, France
Telephone: (1) 733 08 04
Telex: 620154 (LBFRA)



Lifeboat Associates, GmbH
Hinterbergstrasse 9
Postfach 251
CH 8330 Cham, Switzerland
Telephone: 042 36 8686
Telex: 865265 (MICO CH)

Intersoft, GmbH
Schlossgartenweg 5
D-8045 Ismaning, W. Germany
Telephone: 089 966 444
Telex: 521 3643 (ISOFTD)

Lifeboat, Inc.
5-13-14 Shinba
Minato-ku, Tokyo, 108 Japan
Telephone: 03 456-4101
Telex: 2423296 (LBJTYOJ)

Lifeboat Associates
World's Foremost Software Source

The exchange library of The CP/M® Users Group (CPMUG™) contains 100's and 100's of programs catalogued into over 80 volumes of software available to you at nominal cost.

Everything from:

- Languages such as FELIX, SAM76, BASIC E, etc.
- Assemblers and Disassemblers
- Editors
- System utilities, including a complete disk cataloging system
- Games, including Adventure
- Special applications such as Animation and Computer music
- And almost everything in full source code.

The complete catalog of CPMUG volumes is available for \$6 prepaid to the U.S., Canada and Mexico. \$11 prepaid to all other countries.

*Domestic price for 8" disks.
Inquire for NorthStar and overseas prices.

Copyright©1982, by CPMUG.
CP/M, reg. trademark of Digital Research, Inc.
The CP/M Users Group is not affiliated with Digital Research, Inc.

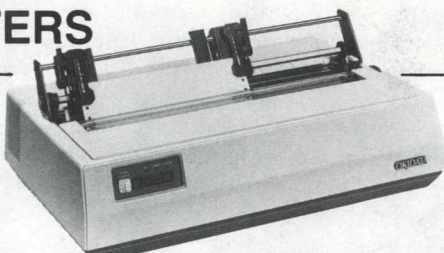
Over
10
MBytes
Of
Software
At \$8* Per
Diskette
Full

THE CP/M USERS GROUP

1651 Third Avenue, New York, N.Y. 10028

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)

Opinion

Editorial Comments

Edward H. Currie

All things come to those who wait . . .

Interestingly enough, microcomputer telecommunications is evolving rapidly for microcomputer systems and this, as we shall see, is indeed a timely development.

IBM decided some years ago that the world's supply of programmers was, in their words, "running out". This led them to study in more detail how best to meet the growing needs for computers. Several interesting conclusions were drawn.

For one thing, the average response time of a human was determined to be on the order of one hundred milliseconds. IBM's studies showed that it is very important for a computer system's response time to be compatible with its human counterpart since a slower system response time would result in an interruption in the thought processes of the user. Ideally all operations and concatenations of operations should take place in less than one hundred milliseconds.

Furthermore, data movement and manipulation were found to be often more important than data reduction. The establishment and maintenance of data bases from which data could flow and ebb was another important consideration. The ability to move large masses of data over large distances is obviously equally important.

Clearly floppy-based systems with three to ten millisecond track-to-track times are not likely to yield overall system response times of fewer than one hundred milliseconds, but micros can be the vehicle for the movement and manipulation of data.

The simplest mechanism for linking micros to other micros, as well as to larger computers, is currently via telephone lines.

As mentioned in previous editorials, a flurry of activity has occurred with the advent of computerized bulletin

boards. These systems typically support the transfer of programs/data at baud rates of 300-600 and in some cases as high as 1200 baud. A significant body of software has resulted from interests in such systems.

Another development in the microcomputer field is also having an impact upon the growth of microcomputer telecommunications, while appearing to be somewhat unrelated.

Since it is true that many microcomputer operations are not "compute bound" but rather "I/O bound", software developers concluded that this spare CPU time could be used to good advantage. From this conclusion sprang the multiuser operating system for microcomputers. Unfortunately eight-bit architecture does not lend itself well to such designs. To date the various multiuser systems for micros have for the most part met with little success, due to several factors.

Curiously enough, the significantly greater transfer rates possible with Winchester drives have been overlooked by many system designers. Typically Winchester drives are configured to function as floppy systems with thyroid problems (i.e. they function as little more than oversized floppies) and the resultant I/O transfer speeds are so slow as to seriously degrade the system in multiuser applications in all but the most trivial cases.

However, multiuser software has proven to be of significant value in the multitasking environments of business and professional end users. In such situations a typical user does not want to lose access to his system while it is performing such simple tasks as transferring a file to another system or printing.

So-called "concurrent" systems support multiple tasks like background printing and file transfer via phone lines. These "background tasks" fortunately are slow enough to permit them to occur contemporaneously with foreground tasks, such as word processing,

applications programs, assemblies, etc.

Thus the agonizing over the best schemes for locking records and files, and the protection of data bases' integrity against transgressions committed by trespassers in a multiuser environment can be avoided. There is still, however, the nagging problem of how to handle different tasks accessing the same files, so that the integrity of the data base is maintained.

Many hardware manufacturers are assisting. They offer buffering for printers which allows high speed bursts of data to be transferred to a printer buffer during slack periods, further freeing up CPU time. In those cases where CPU time is available, considerable advantage is gained by passing as much data as possible to the buffer. Then when CPU demand becomes critical for other activities, the printer has lots of text to print while waiting for the next time the CPU is available.

Five, six and eight megabyte clock rates for the standard CPUs further facilitate the support of multitasking. Add to this the buffering employed in many of the newer terminals to avoid character loss, together with faster track to track times, transferring of whole tracks into memory, etc. and you can readily see that the ground work for multitasking is being laid rapidly.

The advent of low cost 64K DRAMs (Dynamic RAMS) means that micros capable of addressing a megabyte or more of memory will be commonplace. This means, among other things, that multiple tasks can reside in memory simultaneously to further increase system throughput.

Finally, the use of DRAMs to simulate floppy disks (a virtual floppy) will do much to increase effective disk access and at prices which are well within the reach of most end users. Expect to see many memory vendors offering the software necessary for floppy simulation.

(continued on page 11)

Lifelines/The Software Magazine, June 1982

Pick your modem, folks

Although direct-connect modems have been available since the mid-seventies, the latest advances in LSI technology have made it possible for modem manufacturers to develop an entirely new breed of low-cost direct-connect modems.

Specifically, modem manufacturers are offering designs with a number of attributes that greatly simplify integrating a modem; these include:

- Increased use of LSI and uP technology to reduce overall size and cost
- Modular sub-system packaging
- Flexible interconnects; cable and direct pin
- FCC certification
- Auto functions: dialing and answer, plus stored number features

This new breed of modems is significant because they reduce overall system complexity by eliminating extra cables and circuitry required for RS-232 interconnection; they also reduce the extra power supply required, since most operate off standard voltages available within a system.

Moreover, by using modular designs such as those offered by Novation and Rockwell, for example, you can "design in" a modem to best fit your space and operational requirements.

The Novation module series consists of a low speed modem P/N 490280-X and the phone line interface P/N 490278. The 1200 baud 2- and 4-wire Bell 202 compatible version (designated by -2) of the modem module is priced at \$74(500), with a similar price for the phone line interface.

The modem module employs LSI technology, is crystal controlled, and operates at 300 and 1200 baud in either half or full duplex. In addition, the module,

which measures 2.1- \times 2.75- \times .5-in., features answer/originate, self test mode, dial tone/busy filter mode and can be easily programmed by uP or switch controls. The phone line interface occupies the same amount of space as the modem and if added provides FCC certification as well as automatic pulse and tone dialing with an external uP.

Rockwell International offers similar capability with the R24 2400 bps integral modem package. This modem series is a high performance synchronous serial 2400 bps DPSK modem. The modem employs MoS/LSI technology and is implemented in three modular building blocks: a transmitter Module T at \$118; a receiver in two modules R1, R2 at \$218; a complete kit is available for \$395.

The Rockwell modules allow you to implement a modem using only those modules necessary for the operation. Like the Novation modules, Rockwell's designs offer FCC registration, Bell 201 B/C and CCITT V.26 compatibility.

Of course, not all applications call for modularity. Therefore, Rockwell offers the R24 fully configured on a 5- \times 7.8- \times .6-in board with full auto functions, FCC certified direct connect functionality and line equalization. The price is \$450.

In modern modems size is just as important as modularity. Currently, manufacturers are striving to develop modems which require as little real estate as possible, exemplified by the modular designs previously mentioned.

For instance, Radio Shack is offering the Modem II, for \$249. This modem is a Bell 103- compatible direct connect design, and uses a built-in uP to control the automatic functions: answer, originate, pulse and tone dialing, and auto on-hook off hook.

Universal Data Systems, however, has elected to be speedier with the UDS 212

LP, priced at \$495 (single quantity). This modem is Bell 212-compatible and operates at 1200 bps; it's designed for direct connect applications, and uses line-derived power. The modem is 1200 baud only, and doesn't offer auto functions.

But if dual speeds and auto functions are a requirement, you should consider the Codex Model 5212R for \$695. This Bell 212A compatible modem employs uP control, and combines 300 and 1200 asynchronous functions as well as 1200 baud synchronous capability in one box. In addition to typical auto functions of auto answer, dial, and line control, the 5212R includes auto speed sensing and automatic switching to answer mode.

This size reduction is achieved by employing LSI and uP technology for everything from control to the analog portions of the modem.

Furthermore, box modems designed to plug into a terminal or system are being offered with some innovative packaging. The Universal Data Systems Model UDS 202 LP, 0-1200 bps half duplex two-wire model, priced at \$265(25), for example, is designed in a slimline package to fit directly under a telephone. The small size is achieved by deriving power from the telephone line (20 mamp at 5 V, 13 mamp internationally). The goal of this type of modem, also available as a board level product, is to give hardwire functionality with acoustic coupled portability. This goal has become much easier to achieve with the now-relaxed requirements for a Direct Access Arrangement (DAA) - for using the Bell switched network system.

Not everyone is opting for the same design philosophy, though. The Micro-peripheral Corporation offers a slim line 300 bps modem for personal computers, priced at \$199.50 with a \$79 autodial option. This modem fits neatly under the telephone, but rather than using the line voltages requires a small

(continued next page)

5V power source. This design, according to the company, is to ensure that adequate power is available for the various functions their modem permits.

Still more to choose from

Depending on your application, you might elect to choose from among the many offerings of Astrocom Corporation. For specialized applications, you could look at Model 140-0, which transmits at 150 bps and receives at 1200 bps in an asynchronous mode 4-wire full duplex, or the model 140-A which transmits at 1200 bps and receives at 150 bps. Astrocom supplies modems either boxed for plugging directly into a system, or on the card level.

If you're looking for a design with such full features as 1200 bps operation, auto dialing and answering, touch tone signaling and the ability to understand either direct or system commands, you might want to check out the Model 1012 Intelligent Modem from Bizcomp Corporation. This \$895 modem offers user programmable answer back, the ability to store the last number dialed, FCC certification, and automatic speed sensing. If you don't need the modem packaged the company offers it at the board level.

Another notable modem design is the VA3450 series of triple modems offered by Racal-Vadic. These modems, which sell for around \$350(25), include Vadic VA3400, Bell 212A and 103 compatibility in a single package. Moreover, the modem operates in either a synchronous or asynchronous mode, depending on operation. The Vadic and 212A portions employ a quadrature AM four level PSK modulation scheme, while the 103 portion uses binary phase coherent FSK. The VA3450 series also sports a receiver sensitivity of -50dBm when receiving with equalizer in, and in the 103 mode operates at a nominal -45 dBm.

Offering a similar modem is Prentice, with the Model P-V.22. This unit (priced at \$1034) meets CCITT v.22 standards and has 2-wire full-duplex capability; data rates of 1200 and 600 bps are possible in the synchronous mode, while 1200, 600 and 0-300 bps rates are possible in the asynchronous mode.

If your application calls for speed above 1200 bps, the Codex Model 5208R Data Modem might fill the bill. This \$2450 modem offers strap selectable switching between Bell 208A (leased line) and 208B dial up line modes. In addition, the modem sports self-testing and condition reporting via front panel LEDs.

Another modem for high speed operation is the Kinex Microprocessor Data Modem K9600. This \$3950 unit meets CCITT v.29 standards, operates over 4-wire unconditioned domestic and international leased voice grade lines, and provides user selectable data rates from 4800 to 7200 bps, to accommodate severely degraded lines.

The K9600 is unusual in that all functions (including filtering) are implemented using firmware. This feature, explains company president Carl Nordling, means that all updates to the modem can be accomplished in the field at very low cost.

Still notable, but not falling into the direct-connect class, is Paradyne's Model T-96 priced at \$2515(100). This 9600 bps modem is intended for use in full-duplex, point-to-point applications, and features a training time of 253 Msec, fallback operating rates of 7200 and 4800 bps; and it is CCITT v.29 compatible. Like other modems in its class, the T-96 sports built-in diagnostics and permits either local or remote testing.

The 300 bps connection

Although data rates of 1200 bps and above are rapidly gaining great importance for high speed transmission in the commercial world, 300 bps modems are still well-entrenched for personal computers.

One innovative design that meets the personal uC user's requirements of direct connect and low price is the Model TC 4007 from Tek-Com. This \$495 modem operates at 0 to 300 bps, has a dynamic range of -10dBm to -56dBm, is FCC certified, and sports a built-in dialer with automatic re-dial capability. In addition, the TC 4007 allows for field programming of an auto dial function either via the built-in keypad or by the uC system software.

A unique design built around the Motorola 6860 modem chip is the BC 103, a 300 bps modem from BC Electronics. This \$225-modem offers auto answer, and dialing. The BC 103 is strictly a hobby modem and is sold through Heath Electronic centers, as is the optimized HDOS operating system designed to work with it.

Considered the undisputed leader in 300 bps modems for personal systems is Hayes Microcomputer Products Inc. The latest model yet introduced, called the Hayes Stack Smartmodem, is designed to operate with any system with an RS-232 serial port, and can be controlled by any language using ASCII character strings. This FCC-approved modem sports built-in diagnostics, and a system monitor program that controls the functions of the modem. In addition, the \$279 modem supports touch-tone or pulse dialing and can work through a PBX board.

Look at the standards

Virtually all modems currently available conform to either those standards established by Bell or the standards established by the Consultative Committee for International Telephone and Telegraph (CCITT). The latter's standards are rapidly becoming the most widely accepted, because of increased world-wide data communication, with even Bell moving towards the CCITT definitions.

These standards are those definitions describing exactly how signaling will be executed over a given line; bandwidths, scrambling, and training sequences are laid out so that compatibility exists between modems. Of course, other methods do exist but currently design criteria for modems operating over the switched network, or private 3002 series lines, require compliance with the accepted standards.

This compliance, although it helps avoid chaos in the communication world, isn't cheap. Meeting Bell 212 standards for 1200 baud operation, for example, ups the cost of the modem, since extra circuitry is required to fulfill the standards requirements.

Information pushes designs

Another influence that is pushing rapid innovations in modem design is the

increasing use of dialup databases, like those offered by Compuserve (Columbus OH), and The Source (McLean, VA).

According to a report from the Yankee Group (Cambridge, MA) by 1989 it is expected that 65 percent of American women will have full-time jobs, leaving 60 percent of all metropolitan households unattended during school hours. This is likely to fuel demand for home security systems and remote-controlled appliances, stimulating a need for very specialized communications devices. In addition, because of the time being taken away from traditional homemaking, it's expected that services like QUBE (Columbus, Ohio), Viewtron (Coral Gables, FL), and Hi-Ovis (Osaka, Japan) will become commonplace, as they offer the general public two-way communication for handling everything from banking to shopping.

Moreover, systems like The Microperipheral Corporation's 'public modem', a variation of the Bell 202 type device, will grow in usage. This 'public modem' is capable of continuous reception of data at 1200 baud while simultaneously sending data at rates of up to 150 baud. The modem is connected to the terminal at 9600 baud, for example, and through speed-changing circuitry the transmission speed over the line is lowered to 150 baud. Currently, the 'public modem' is priced at \$199.50 with a \$49 autodialing option.

The Microperipheral Corporation's chief engineer, Don Stoner, explains that cost can be lowered by developing modems with slow transmit speeds but high receive rates. He expects that similar techniques will be employed even as LSI costs go down. What *will* change, however, will be the speeds. So don't be surprised to find under-\$200, full-duplex 1200/4800 bps modems as early as 1983.

In related moves towards greater flexibility in providing communications functions, a number of terminal manufacturers are getting on the communications bandwagon, by providing user-oriented communications terminals.

One company seeking to provide remote data communications capability at a low price is Zenith Data Systems (ZDS), with their model ZT-1 personal information terminal.

The ZT-1, introduced this past March at the West Coast Computer Faire (held in San Francisco, CA.) was developed, according to product line manager Michael Brenner, to meet the needs of new information services like the Source (McLean, VA).

The ZT-1 is a two-piece unit made up of a keyboard terminal, and the Zenith ZVM-121 video monitor. The keyboard unit measures 2.9- \times 15.4- \times 7.1-in. weighs 4.4-lbs, and houses a Bell 103, 300-baud modem, a 63-key keyboard with 26 alphabetic, 10 numeric, 4 cursor/special function keys, a serial RS-232C port selectable from 110 to 2400 baud, and a Centronics type 8-bit printer parallel port. The system I/O is controlled using an Intel 8051, and battery backed-up CMOS memory is utilized for storing directory information, used by the automatic dialing function.

The video display unit weighs in at 14-lbs. and measures 11.75- \times 16.25- \times 12-in.; it has a green phosphor screen, supports a display format of 80 characters by 25 lines with a character matrix of 5 \times 9 in a 8 \times 10 character field. In addition, the CRT has a bandwidth greater than 15MHz, and a typical video rise time of 50 nsec.

Connecting the ZT-1 to the telephone line is handled by a standard RJ-11C, 12C, or 13C telephone jack. The dialing method is pulse, which some observers feel may be an inhibiting factor should the terminal be employed in an office with a computer controlled PABX.

On power up, the ZT-1 displays the choices available. A series of menus or indices makes it possible for you to choose one key for most functions; the functions include: setting terminal parameters; baud rate (110-300); sign on message sequence for a remote service; parity and data word length. The ZT-1 also has the ability to enter up to 26 telephone directory numbers. Once entered, the numbers can be dialed by simply tapping the appropriate letter.

In addition to serving as a communication terminal, the ZT-1 can be attached to a printer and used as a low-cost electronic typewriter with essentially one page of storage. Or if you like, received information can be driven to the printer, with automatic XON/XOFF

protocols to avoid loss of data due to a filled buffer.

Interestingly, the ZT-1 is offered at \$695 (single quantity), and ZDS is willing to provide custom firmware to fit your specific needs. In addition, you can probably expect some more additions to turn the system into a relatively low-cost uC - and possibly a LAN node, by year's end.

Offering similar capability, but with a different twist, is Tymshare, with the Scanset personal information terminal. This terminal, ranging in price from \$495 for the Model 410 without an integrated modem to \$649 for the Model 415 with modem, was designed and built by the French manufacturer MATRA.

The Scanset units have six multifunction keys that you can program, or it will accept downloadable information from a host. Up to 12 user-defined tasks can be assigned to the keys, providing easy access to host systems or frequently-used data bases.

The autodialer feature of the Model 415 can dial up to 36 phone numbers stored in the terminal's inviolate memory, and automatically handle the necessary password and sign-on functions.

Like the Zenith terminal, the display handles an 80 character by 25 line display with the twenty-fifth line serving as a function key descriptor. Unlike the Zenith system, however, the Scanset employs a 9-in. screen, and has a built-in speaker that echoes the line during dialing and connect.

The Scanset is also smaller, measuring 9.5- \times 10.25- \times 14.5-in. and weighing 12-lbs. Furthermore, the terminal is a single unit with a small square button 63-key keyboard, not really suitable for typing, a factor which Tymshare officials consider unnecessary for database query.

The Scanset employs a 6802 uP, and battery backed up CMOS, which like the Zenith unit is recharged every time the terminal is plugged in. Should higher speed than 300 be important, Tymshare also offers an optional Model 912, a Bell 212A compatible modem for \$900.

Although modems are built to exacting
(continued next page)

standards established by Bell and CCITT, you might want to consider implementing an alternate approach, especially if you're establishing a Local Area Network (LAN).

The method is to employ a new modem design from The Microperipheral Corporation. The yet unnamed modem operates at 4800 bps, has no filters and sells for about \$75 in OEM quantities. The technique, explains Don Stoner, is to encode data on the half-cycle. One bit of data at 4800 bps, for example, is equal to one-half cycle of 2400 Hz; by knowing this and using zero crossing detectors in the receiver the time domain between the zero crossings conveys the digital information. Through this technique, the signal amplitude becomes unimportant and obviates the necessity of building in line conditioning circuitry and providing training sequences.

Even though the modem doesn't conform to accepted standards, Stoner believes that personal computer users will find it more than acceptable, since they can set up high speed networks for very little cost. Moreover, he believes that timesharing houses will find the modem acceptable, since an infinite number of them can be connected together with the controlling function being the Clear to Send (CTS) line. Stoner explains that it's very much like a hardware network system with masters and slaves.

Because this modem approach is so much like a network server concept, software is required to arbitrate line collisions, and to set priorities.

A few buying rules

If you're in the market to buy a modem, you might do well to follow some simple rules offered by Jim Jordan, president of Moxon Electronics Anaheim, CA.

Jordan suggests that you contact those vendors with a reputation for reliable delivery, who can meet the quantities that you need. Next, visit the plant and see how the modem is made. Look for ATE equipment, and a smooth manufacturing flow.

Should the vendor(s) meet your expectations, take a look at the packaging.

Specifically, look for Bell 103/212 functionality: for example, insist on built-in diagnostics, look for auto functions and stored number features. Most importantly, pass on anything that isn't FCC registered.

Jordan points out that just about anyone can build a modem and that pricing is about the same across the board. The real difference, therefore, is in who is the most responsive to your needs and can make a major commitment.

For more information . . .

For more information on the modem products discussed in this article, contact the following manufacturers directly:

Astrocom Corporation
120 West Plato Blvd
St. Paul MN 55107
(612) 227-8651

BC Electronics
1001 W. Kristal Way
Phoenix, AZ 85027
(602) 869-9650

Bizcomp Corporation
P.O. Box 7498
Menlo Park, CA 94025
(415) 966-1545

Codex Corporation
20 Cabot Blvd
Mansfield, MN 02048
(612) 364-2000

Hayes Microcomputer Products Inc
5835 Peachtree Corners East
Norcross, GA 30092
(404) 449-8791

Kinex Corporation
6950 Bryan Dairy Rd
Largo, FL 33543
(813) 541-6404

The Microperipheral Corporation
2643 151st Place N.E.
Redmond, WA 98052
(206) 881-7544

Novation
18664 Oxnard St
Tarzana, CA 91356
(213) 996-5060

Paradyne Corporation
8550 Ulmerton Rd
Largo, FL 33541
(813) 530-2000

Prentice Corporation
266 Caspian Dr
Sunnyvale, CA 94086
(408) 734-9810

Racal-Vadic Inc
222 Caspian Dr
Sunnyvale, CA 94086
(408) 744-0810

Rockwell International
Electronic Devices Div
3310 Miraloma Ave
P.O. Box 3669
Anaheim, CA 92803
(714) 632-3729

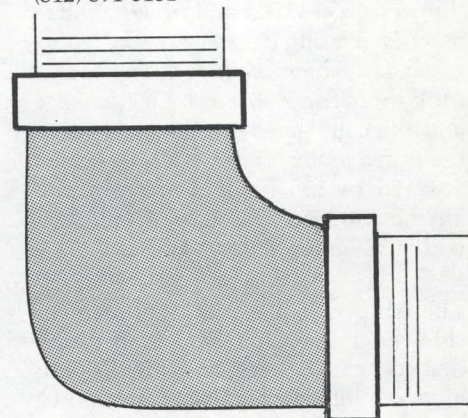
Tek-Com Inc
2142 Paragon Dr
San Jose, CA 95131
(408) 263-7400

Universal Data Systems
5000 Bradford Dr
Huntsville, AL 35805
(205) 837-8100

Tandy Corporation/Radio Shack
1800 One Tandy Center
Fort Worth, TX 76102
(817) 390-3300

Tymshare Corporation
20705 Valley Green Drive
Cupertino, CA 95014
(408) 446-6000

Zenith Data Systems
1000 Milwaukee Ave
Glenview, IL 60025
(312) 391-8181



Notice

The May issue was placed into the mail on April 25th. 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 June 1982, into the mail around May 29th. We will print each month the date of the previous issue's mailing and would appreciate your help in tracking the deliveries.

(Editorial Comments, continued from page 6)

Ultimately, the drastic reduction in hardware costs means that each end user will typically have their own printer and mass storage devices and therefore a vested interest in multitasking.

Thus it appears that multiuser systems may well be prevalent, but multitasking will undoubtedly take precedence

File servers, which are devices for supporting access by a number of microcomputers to a Winchester disk, are also evolving rapidly for the implementation of local networks of micros.

Transfer of data via phone lines does introduce a number of additional needs for standardization. For example, handshaking is imperative in order to assure that file integrity is maintained. Compression schemes have also evolved rapidly in order to avoid needless transmission of large numbers of blank lines, spaces, tabs etc. These schemes must be standardized, so that files can be readily and reliably moved from one location to another. Also very important are schemes used for files with embedded format instructions, like those produced in word processing applications to be readily interpretable by other word processors from different vendors.

Thus it appears that multitasking will come into widespread use rapidly, with considerable emphasis placed upon telecommunications.

Note that all of the handheld micros are supported with telecommunications schemes. A particularly exciting development is the announcement by Sinclair, Sony, et. al. of flat, miniature TV screens of various designs. Liquid crystal as well as CRT technology will soon provide a handheld computer the size of a book which will, with bubble memory or other suitable mass storage, mean a system with mind-boggling capabilities. Imagine a handheld, multitasking microcomputer with virtual disk storage, optional miniature floppies, integral 24x80 CRT, full keyboard, built-in modem and printer in a package, all fitting easily into your attache case.

One of the areas now receiving considerable attention is computer graphics. Newer micros will undoubtedly be designed to support various graphics schemes, some of which will have incredible resolution. Printer graphics of surprising resolution are rapidly coming into widespread use as well. Light pens, touch screens, voice recognition and speaking terminals are also becoming more prevalent. Clearly, all of these features are greatly enhanced by a multitasking environment.

In the years ahead software authors have their work cut out for them to take full advantage of all of these exciting hardware developments.

Perhaps the best part of all is that their efforts, together with the emerging hardware technology and falling hardware costs will place these micros within everyone's reach. And all we have to do is wait . . . and not too long at that . . .

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 TM Ashton-Tate
Wordstar TM Micropro Int'l

Features

A Review of FMS-80

Mark Rettig

Name of package:

FMS-80 Release 2.21D

Author:

David Rodman, DJR Associates
Distributed by Systems Plus, Inc.

Addresses:

Systems Plus
1120 San Antonio Road
Palo Alto, CA 94303
Phone - (415) 969-7047

DJR Associates
2 Highland Lane
North Tarrytown, NY 10591

You may have seen the recent full page advertisements for "FMS-80: The Two Door Data Base Plus", and wondered whether it really lived up to all its claims. I first saw FMS (File Management System) nearly a year ago, and was impressed enough to choose it over all the other file management systems on the market, thinking it was exactly what I needed to develop a rather extensive data management application. My experience with FMS-80 has been much like my experience with my first high school girl friend: I fell in love for a month, became disillusioned when I learned she wasn't perfect, and finally decided to be "just friends". That could be an outline of this review.

FMS-80 is a large and very powerful package. You could use it to develop a major application and still not find occasion to utilize all the features or explore all the possibilities. This review will describe the major components of FMS and briefly evaluate their effectiveness and ease of use.

Installation was no problem. Parameter files are included for many popular terminals, and a utility is provided to set up FMS for systems whose parameter files are not predefined. Once you get everything up and running, you can easily customize many features to suit your taste. For example, reports can be routed to different devices, paper size defined, file name conventions can be changed, and default drives for different file types can be specified. Nice.

The tutorials in the manual are great for learning the system initially. Overall, the tutorials are the best part about the documentation. It is easy to learn the basics, but the manual suffers somewhat from a lack of examples in the reference section (the report generator excluded). Some help in that area is found in the diskette of sample programs, which contains some very helpful code (some of it is even useful!). Between that and the manual you have most of what you need.

FMS-81: Menu-Driven Utilities

If an application neither involves complex relationships between records nor demands that many files be opened at one time, it could probably be developed using only the set of menu-driven utilities marketed as FMS-81. The command language is sold separately as FMS-82, so non-programmers need not spend money on something they will never use. Whatever you are doing, the first part of FMS you see, and the part you will use in first setting up your application, is the set of menu-driven utilities provided in FMS-81.

Despite what the ads might say, FMS is a file management system and not a data base management system in the strict sense of the term. This means that the user must be concerned with the physical structure of his files, and take pains to insure that file indices are current. For the most part, record updating is a "batch" process of applying a transaction file against a data file. This set of menus and utilities is designed to make all that as painless as possible, and saves the non-programmer from having to learn all about file management to develop his application. The menus are divided basically into three functions: file definition, file maintenance, and report generation. These are described below.

Definitions Editor

File definitions, screen and menu definitions, and report definitions are all created using a "split-screen editor". The bottom of the screen provides a place for entering field names, data types, headings, etc., and the top is where you see the results of what you typed in. FMS's editor is not like a screen or line editor where you can type in a free format. It is a very specialized editor which helps you enter valid definitions through its formatted entry area, by reminding you of what is valid, and by refusing to accept things like invalid field types or digits in an alphabetic field. This helps make life easier. The only features the editor lacks are a few short-cuts, such as the ability to copy or move entries. If you have a long definition to key in, be prepared to sit down and type it all, even if many entries are nearly identical.

The definitions themselves are very straightforward. For file definitions, specify the field name, data type (alphanumeric, decimal, or variable length), field length, and, in the case of decimal fields, "picture" format. Screens for data collection and display are set up just by telling FMS what fields and literals to display and where on the screen to put them.

One type of definition merits special mention: the menu definition. By using the same editor described above, you can develop menus to tie all the functions of a custom application

into a package. If you've ever coded such an animal in BASIC you can appreciate the following procedure. Just tell FMS where to print the options on the screen, and what programs, screens, SUB files, or other menus to execute for each option, and FMS takes care of the rest. Without coding a line, you can set up a custom menu-driven application full of screen input, sorts, selects, and reports. Of course there are limitations, but for many common office and information handling chores, this is just the ticket. I found this feature to be helpful in setting up a development environment for myself. A "test-ing system" can be set up with menu options to enter your favorite program editor, to compile your EFM programs, and to run each one with test data. Other options can print reports of test results.

File Maintenance Utilities

Now that you have all those files defined, it is time to do something with them. The File Maintenance Menu provides facilities for entering data into your files, sorting files, and updating the index for random access. Each of those functions can be performed separately, or the "update" option can be used to do them all in one swoop. You enter "add", "change", or "delete" transactions, validate and sort them, apply them to a data file, and update the index to the file all in one operation.

Rejected transactions are written to a file for later examination. Update can be combined with a screen definition for "instant" custom data-entry, complete with validation of data types and transactions. This whole process is pretty straightforward, and once you get used to it you can get a lot of work done in a short period of time. Besides the update utility, a few CP/M facilities are available, such as ERase, RENAME, and DIRectory, or you can execute a CP/M SUBMIT file. If you are imaginative you can get very creative with just CP/M SUBMIT files and FMS commands.

One other way of browsing and updating files is through the "Direct Query/Update" utility. It is faster than "update", but watch out when you use it, as it does not leave a printed transaction trail or update the index to the file.

Report Generator and Selections

FMS includes a very complete report generator. I can't think of anything you might need to do in a report that isn't provided for here. Reports are defined using the familiar definitions editor, and one command applies the report definition to a file to produce printed copy. You can produce summary totals at field breaks, page breaks, and end of report; headers, footers, page numbers and dates can be placed appropriately. New fields for the report can be calculated from fields in the file being printed.

By creating a "selection definition", you can get a report of only the records in the file meeting specific criteria. Those criteria can be fairly complex. Records can be chosen based on whether certain fields equal a value or fall within a certain range. Criteria can be combined using logical operators. For

example, a selection might be made in a sales application for all records whose department equalled "sports" or whose salesman was "Jim" and whose quantity was less than 100. Of course, use of selections is not limited to the report generator. Output from selections can be routed to a disk file as well as sent directly to the report generator.

SHELL-80

An interesting and powerful feature of FMS-80 is called "the SHELL". What it amounts to is a command line monitor which replaces that of CP/M, and provides an area of storage that remains "live" across program calls. This means that command streams can be built dynamically, programs can easily be linked with menus and other programs, and programs can receive input from text files rather than the keyboard. By setting a "base program" before calling other programs, return to the caller is guaranteed no matter what happens in the called routine. The SHELL is the heart of FMS, and is an exciting idea. It makes possible such niceties as the "help" facility, which allows applications screens to be tied in with text screens. While the utilities are in use, SHELL-80 is invisible and many users will never need to bother learning about it. An applications programmer has an interface with it through EFM, FMS-80's command language. If you really want to get into it, all you assembler types can order the SHELL manual and opcode your hearts out.

There are twenty-seven functions available through the SHELL, such as: SUBMIT, which submits any CP/M command for execution, set base program, return to base program, and get console input from a file. One drawback. I had a hard time figuring out how to use the SHELL, and had to get help from my dealer. The problem isn't so much the product as the documentation, which dedicates only nine pages to describing the SHELL and its commands. The manual does not tell you how to use the SHELL. The SHELL manual, available separately for \$15, is some help, but is mostly for assembler programmers and is still only forty pages long. If you have time to experiment, great. If not, find somebody who can help you out, or call Systems Plus. They do their best to aid bewildered programmers, and their best is pretty good. You may have to pester an operator to finally get your man. They seem to be pretty busy out there. But once you get help, you'll get help.

FMS-82: The EFM Command Language

So far, so good. We have everything defined, data keyed in, reports coming out, and the package is tied together with nice menus. But now, suppose your needs are a little more complicated. You want to display information from two or three files at a time on a single screen, or have input from one screen update several files. That means it is time to use EFM, the command language of FMS-80. EFM, in case you were wondering, stands for Extended File Maintenance - perhaps the idea is that you can go beyond the file maintenance capabilities provided in the utilities described above.

EFM has some good features, the strongest of which are the file access and screen development commands. You can
(continued next page)

perform random and sequential access and update of up to nineteen files in a program. Through the SHELL, programs can call other programs (even BASIC programs) or execute CP/M commands. Given time and the patience of Job you could write some very complex applications in EFM. Screens are fairly straightforward to write, and with the use of graphic characters and reverse video, you can make them look very classy. It is really a lot of fun to write a screen and then see FMS make it look fancy by allowing wild-card searches and validating all the input. Execution is reasonably fast, and once the indices are loaded into core, random access of files is also fast.

Another good feature of EFM is its compiler. Although it may seem limited compared to some of the bigger compilers around, it serves a great purpose by reducing source code to a module about one third its original size. This process seems to involve the removing of blank space and the replacement of keywords with tokens. The compiler catches most syntax errors, and tries hard to let the programmer know what he or she did wrong. It occasionally gets confused, but don't we all? The compiled modules are not directly executable COM files, but require, along with them, a run time module in core to be executed. All this is very good. But there are some weaknesses which offset the good points of the language and are somewhat limiting: the lack of control structures, the lack of named variables, and the lack of string handling functions.

The language contains just barely enough control structures to let you get through a program. It has IF - THEN - ELSE, a SWITCH statement (case structure), and provision for CALLing internal subroutines. That is all. There are no loop structures and nested IFs are not allowed. Of course, this can be overcome by careful use of labels, CALLs, and GOTO statements, but such coding takes time, is awkward, and discourages good programming habits.

The use of variables in EFM is very limited. Data types are character, decimal, and variable length characters. Named variables are limited to the letters A - Z, which must be decimal numbers. All other variables are referred to by a pair of numbers representing the file number and field number. The third field in the first file opened for input is referred to as "1,3". Its header in the file description might be "number of widgets", but unless you comment your code thoroughly or memorize all your file descriptions you will have no idea what "1,3" means when you see it on page three of your program. To add two fields together you would say something like "1,5 = 1,5 + 3,2". You could write that in the morning, read it at lunch time, and have no idea what you meant by such an incantation. By setting up a dummy file description you can have temporary storage for up to 255 fields which you can use as program variables. But these are still referenced only by number. So to write an EFM program you need at hand printed copies of all your file descriptions and your temporary fields. And unless you comment every line, the program will be completely unreadable five minutes later.

The third major weakness of EFM is its lack of string handling functions. To get at the middle of a field you have to assign it into shorter temporary fields, taking advantage of right and left truncation of character and decimal assignments. That works okay for numbers, but is awkward. But alphabetic strings cannot be assigned into decimal fields, which means

there is no easy way to get at just the right end or middle of a string. If you must do this, you have to write a dummy file description describing the piece you want as a separate field, and reading the record using the special description! It's enough to bring tears to a whole truckload of BASIC programmers, and will cause a PL/I-80 programmer to faint dead away.

The result of this deficiency is that a system of any complexity will be very difficult to maintain and change. Adding a field on the end of a file may not be much problem, but to add one in the middle means changing almost every statement and comment in every program that uses that file!

Programming in EFM should be carried out with careful attention to "good habits", like thorough commenting and segmentation of programs into modules that perform only one function. Try to make sure a block of code only affects variables having to do with its function, and put a piece of comment up front to tell what variables it is changing. Your comments will not take up internal storage, thanks to the compiler, and you will thank yourself for trying to use structured methods in your programming. You will probably find blocks of code that can be used in several programs and deserve to be saved as separate "include" files. This also will make changes easier to do later on. See the reference section at the end of this article for some help in this area.

By now the reader should understand how one could have a love-hate relationship with FMS-80. One ray of hope comes from the fact that Dave Rodman and his co-workers at DJR associates seem to be very dedicated to supporting and improving their product. Release 3.0 is scheduled to come out this year, and they claim it will heal many of the ills mentioned in this review. Time will tell, but I think the next few years could see FMS become one of those "great" packages if Mr. Rodman keeps at it. Until then, I think I can be friends with release 2.2. As I learned with my high school sweetie, nobody's perfect.

Review Summary

Good Points:

FMS-80's strong point is its set of menu-driven utilities for quick and easy development of file management applications (available as FMS-81). For applications not involving complex relationships between records or accessing many files at once, FMS is great. As far as I know it is the only data management package around with such an extensive and well laid out "front end". Since it writes its files in ASCII, it could conceivably be used as a development tool to create files for input to other systems.

Bad Points:

Although FMS has a great deal of power, and large applications could be written with it, any complex application will involve programming in EFM, FMS-80's command language (available as FMS-82). EFM is a primitive language, and development time is increased by the lack of good variable naming, inadequate control structures, and absence of string handling functions. For data entry and display screens and access of a few files, it is adequate. If you want to do more than that, stay away from FMS-82 until it is improved.

TABLE I Facts And Figures
Package and Version: FMS-80, release 2.21D
Price (Suggested Retail): FMS-81 (File handling and reporting features)- \$495 FMS-82 (Extended File Maintenance command language) - \$495
Systems available for: CP/M, MP/M, CDOS, and TURBODOS
Required supporting software: None for FMS-81, the menu-driven utilities. For EFM programming in FMS-82, you need a good text editor.
Memory Requirements: 48K minimum
Diskette capacity required: Varies widely with the application, but for most two drives are best.
Utility programs provided: A set of editors and screens for defining I/O screens, menus, files, keys, sorts and selections, and for printing descriptions of all types of definitions. Also EFM (Extended File Maintenance), a command language for developing custom applications. A report generator and a direct query/update utility are also provided.
Record size and type limits: Each record may have 255 fields of 255 bytes. File size is limited by disk space. One variable length alphameric field is allowed on the end of a record, but random access is not allowed on files containing variable length fields. Up to 19 files may be opened by an EFM program. There are three basic data types: decimal, character, and variable length character. Decimal fields may be given a picture format. In EFM, named variables are limited to 26 numeric variables named A - Z. A special file provides for 255 other temporary fields. Numeric variables may range from -2,147,483,648 to 2,147,483,647. Numeric literals from -999,999,999 to 999,999,999. Character fields are limited to 255 bytes.
Portability: Good portability between systems with like operating systems.
User skill level required: A novice could develop an application which does not involve complex relationships between records or accessing many files at once. An experienced programmer could develop almost anything.
Systems upgrade policy: \$35 for updates to licensed owners.

TABLE III Data Management Capabilities
A. Underlying Data Model 1. Data Types character, decimal, variable length 2. Relationships no inter-record or inter-file relationships supported.
B. Functions provided 1.a. Data dictionary maintenance File definitions are the only "data dictionary". They are created and changed through the file definitions editor or through EFM. Programs access files by referring to field number in the file definition.
b. Data reorganization and conversion Files can be referenced by many file descriptions. Reorganization or additions to files usually requires changing all related screen definitions and programs. Anything but very minor reorganization would require a conversion program.
2.a. Data entry and editing Good facilities for full screen entry, through both the utilities and the command language. Type checking for data entry is good. Data retrieval by partial keys (wild-card searches).
b. Report Generation Extensive report generation with field, page and report break totals, page numbering, and header and footer lines. Supports calculation of new report fields from fields in the file being printed.
3.a. Data selection Good. Records can be selected by several criteria and either written to a file, a report, or simply counted for a report of the number of records meeting the criteria.
b. Data joining and relating multiple data sets A utility is provided to append one file onto the end of another. No provision is made for merging files or concatenating data elements in FMS-81. An EFM program could easily be written for such functions.
c. Calculations on data Full algebraic operations in EFM. The report generator provides limited facility for calculation while printing.
4. Data independent application interface Good. Data is stored in straight ASCII, so, reading them into another system should not be hard.

(See next page for Tables II and IV)

TABLE II
Qualitative Factors

	Rating *
Documentation	
organization for learning	6
organization for reference	4
readability	6
includes all needed information	4
Ease of use	
initial start up	6
conversion of external data	5
application implementation	
FMS-81	7
FMS-82	3
operator use	6
Error Recovery	
from input error	7
restart from interruptions	7
from data media damage	3
Support	
for initial start up	5
for system improvement	4

* Ratings in this table will be in a 1-7 scale where:
 1 = clearly unacceptable for normal use
 4 = good enough to serve for most purposes
 7 = excellent, powerful, or very easy depending on the category

Table IV
Summary of FMS-80 Utilities and Commands

Command Name	Description
APPEND	Add records from file2 to end of file1.
APPLY1	Validate transactions, apply to master file.
BATCH	Execute predefined batch command file. (FMS or CP/M commands)
CPRINT	Print a control definition (key definition).
DATE	Set system date.
DEFSORT	Define keys.
DO	Execute EFM program.
EDITFD	Invoke file definitions editor.

Table IV
Summary of FMS-80 Utilities and Commands

Command Name	Description
EDITMD	Invoke menu definitions editor.
EDITRD	Invoke report definitions editor.
EDITSD	Invoke screen definitions editor.
FMS	Execute Shell-80, and enter into the FMS application development environment.
GLOSSARY	Print a file definition.
HELP	Display a text file, indexing it by keyword.
HITCOUNT	Count records in a file meeting selection criteria.
INDEX	Build an index for random access of a file.
MENU	Execute a custom menu.
MDPRINT	Print a menu definition.
PREPARE	Compile an EFM program. Options can be specified to list the entire program and send the listing or errors to the printer or a file.
PRINT	Print a file (or selected records from a file) in a "quick and dirty" format. Not pretty, but you can see your fields without setting up a report definition.
QUERY	An interactive facility for quick retrieval and update of records. No paper trail of changes, and no updating of indexes.
REPORT	Execute a report definition.
RDPRINT	Print a report definition.
SELECT	Create a selection definition which defines criteria for choosing records from a file. Selections can be used by PRINT, REPORT, SUBFILE, SORT, or APPLY1.
SORT	Sort a file by specified keys. Sort an index.
SPRINT	Print a selection definition.
SUBFILE	Apply a selection definition, writing selected records to a new file.
TRANSACT	Interactively create transactions for use by APPLY1: add, delete, change, and inquire.
UTILITY	Get a directory, rename files, and delete files

Features

AUTOLOAD For SB-80, CP/M-86 And CP/M-80 On The Osborne I Computer

Kelly Smith

Although the AUTOLOAD feature of CP/M-80 has been described in various computer/software publications, the second generation equivalent 8080/Z80 CPU operating system, SB-80 from Lifeboat Associates, has this capability – as does Digital Research's CP/M-86 operating system (for the 8086 CPU); this facility loads and executes a user-specified program. Here are instructions on implementing the AUTOLOAD facility on either operating system, using Ward Christensen's DU (Disk Utility, version 7.5 from CPMUG Volume 68 or version 7.7 from CP/M-Net). I've also included a trick for the Osborne I computer CP/M-80 implementation, to AUTOLOAD/START programs in three different ways. Follow along closely, as I explain.

AUTOLOAD For SB-80

SB-80 requires the loading of the system Command Line Interpreter file SYSTEM.CLI (the .CLI is that program portion of the operating system that handles keyboard input from the user) at "cold boot" time. Lifeboat suggests that the .CLI be the first file on the diskette, to decrease the load time required (about 3 seconds, on flexible disk). So, we know that it must start on Group 2.

As a precaution, make sure that you are working with 'back-up' copies of your diskettes. If you are not careful, it's possible to 'patch' your disks into 'disk heaven', never to be heard from again! Here we go; those <cr>'s shown below are your keyboard return key:

```
A>du<cr> <-- run Ward's Disk Utility

DISK UTILITY ver 7.7
Universal Version

Type ? for help <-- enter ?<cr> if you have never used DU!
Type X to exit

:g0;d<cr> <-- goto Group 0, and Dump it...(for the non-believers)
G=0:00, T=2, S=1, PS=1
00 00535953 54454D20 20434C49 00004020 *.SYSTEM CLI..@ *
10 02030405 00000000 00000000 00000000 *.....*
```

– as promised, Group 2 allocation!

SYSTEM.CLI, that we patch for AUTOLOAD

```
:g2;d<cr> <-- goto Group 2, and Dump it
G=02:00, T=2, S=17, PS=20
00 C378B7C3 93B77F02 58440000 00000000 *Cx7C.7.....*
10 00000000 00000000 00000000 00000000 *.....*
20 00000000 00000000 00000000 00000000 *.....*
30 00000000 00000000 00000000 00000000 *.....*
40 00000000 00000000 00000000 00000000 *.....*
50 00000000 00000000 00000000 00000000 *.....*
60 00000000 00000000 00000000 00000000 *.....*
70 00000000 00000000 00000000 00000000 *.....*
```

Ah Hah! Has a very familiar look to it . . . the same old jump vectors, keyboard string length byte, and then nulls as CP/M-80 has. Let's set-up to AUTOLOAD SB-80's extended Directory display program (XD.COM), and see what happens:

```
:ca07,<2>XD<cr> <-- Change to ASCII, address 07, using hex entry
<00><00><00> of <2> for string length, and filename XD
:W<cr> <-- write it to diskette...
```

```
:D<cr> <-- Dump it just to be sure (DU never fails though!)
00 C378B7C3 93B77F02 58440000 00000000 *Cx7C.7..XD.....*
10 00000000 00000000 00000000 00000000 *.....*
20 00000000 00000000 00000000 00000000 *.....*
30 00000000 00000000 00000000 00000000 *.....*
40 00000000 00000000 00000000 00000000 *.....*
50 00000000 00000000 00000000 00000000 *.....*
60 00000000 00000000 00000000 00000000 *.....*
70 00000000 00000000 00000000 00000000 *.....*
```

```
:X<cr> <-- eXit DU, and return to SB-80
```

```
A>; now 'cold boot' SB-80, and it will...
```

Directory on drive B

SYSTEM	CLI	32	CLI	HEX	85	DOS	HEX	106	DUMP	COM	12
LIST	COM	12	PIP	COM	47	SB80	COM	68	STAT	COM	128
STAT	COM+1	4	XD	COM	13	XDF	COM	10	XDS	COM	7
COPY	COM	12	XDIR	COM	16	ASM	COM	64	ED	COM	52
SID	COM	88	SUBMIT	COM	10	XSUB	COM	6	DU	COM	48
COPYFAST	COM	16	FINDBAD	COM	13	CRCK	COM	10	WASH	COM	27
SYSGEN	COM	8									

25 dir entries ---- 120k bytes used, 121k bytes remaining

```
A>; it "autoloaded" XD just fine!
```

AUTOLOAD For CP/M-86

CP/M-86 also has a 'patchable' file for AUTOLOAD; but in this case, it's the entire operating system CCP (Console Command Processor), BDOS (Basic Disk Operating System) and the BIOS (Basic Input/Output System) which are loaded by a special "cold boot" loader. OK, so let's go hunting again, in this case for the CPM.SYS file, but first:

```
A>pip a:=b:cpm.sys[v]<cr> <-- get CPM.SYS file from CP/M-86 disk
```

Note: DU.COM only runs on 8080/Z80 operating systems (until I whip out XLT86 on it), so put the CPM.SYS file on a diskette in your SB-80 or CP/M-80 system.

```
A>du<cr> <-- use Ward's DU to 'patch' the CPM.SYS file
DISK UTILITY ver 7.7
Universal Version
```

```
Type ? for help
Type X to exit
```

```
:= SYS<cr> <-- find ASCII 'space', 'space', 'SYS'
```

Note: Since we put CPM.SYS on this disk using whatever "free space" was available, we can't predict (as we did with SB-80) what the group allocation numbers will be. So, we will let DU find it for us, using DU's '=' command to search for a text string in the directory.

```
= AT 6C <-- last address in search string 'SYS', is at '6C Hex'
G=001:0C, T=2, S=29, PS=28
```

```
:d<cr> <-- dump the group allocation info for CPM.SYS
00 E5434241 53383620 204C4F47 00000009 *eCBAS86 LOG....*
10 CF010000 00000000 00000000 00000000 *0.....*
20 00434150 54555245 20434F4D 00000006 *.CAPTURE COM....*
30 E2010000 00000000 00000000 00000000 *b.....*
40 0043504D 2D535953 20415554 00000015 *.CPM-SYS AUT....*
50 E301E401 00000000 00000000 00000000 *c.d.....*
60 0043504D 20202020 20535953 00000075 *.CPM SYS...u*
70 E501E601 E901E401 EB01EC01 ED01EE01 *e.f.i.j.k.l.m.n.*
```

it's in Group 1E5! (this is a 5 Megabyte Hard Disk)
(continued next page)


```

:gle5;d<cr> <-- go to Group 1E5, then dump it to console
G=01E5:00, T=116, S=9, PS=8
00 01A00340 00A00300 00000000 00000000 *. .@. ....*
10 00000000 00000000 00000000 00000000 *.....*
20 00000000 00000000 00000000 00000000 *.....*
30 00000000 00000000 00000000 00000000 *.....*
40 00000000 00000000 00000000 00000000 *.....*
50 00000000 00000000 00000000 00000000 *.....*
60 00000000 00000000 00000000 00000000 *.....*
70 00000000 00000000 00000000 00000000 *.....*

```

Hmmm...loader information because CPM.SYS is a "REL" file, let's try the next logical sector in the CPM.SYS file:

```

:++;d<cr> <-- advance +1 to the next logical sector, and dump it...
G=01E5:01, T=116, S=10, PS=9
00 E92A03E9 2103E902 037F0020 20202020 *i*.i!.i.... *
10 20202020 20202020 20202043 4F505952 * COPYR*
20 49474854 20284329 20313938 302C2044 *IGHT (C) 1980, D*
30 49474954 414C2052 45534541 52434820 *IGITAL RESEARCH *
40 20000000 00000000 00000000 00000000 *.....*
50 00000000 00000000 00000000 00000000 *.....*
60 00000000 00000000 00000000 00000000 *.....*
70 00000000 00000000 00000000 00000000 *.....*

```

O.K., we found what we were looking for, three jump vectors this time (and yes, the CCP string length byte again). Let's AUTOLOAD XDIR, translated (8086'afied) from the 'Sorted Directory' utility SD.ASM (CPMUG volume 65) just last week! Note, that a number of utilities being translated to run in the CP/M-86 environment (FINDBAD, FILE-EXT, MUCHTEXT, etc.) are available from the CP/M-Net™ System via modem, at (805) 527-9321 (as well as from other RCPM's).

```

:caa,<4>XDIR<0><cr> <-- change ASCII, starting at address 'A', to:
<00> 4 (hex entry), XDIR (ASCII entry, 0 (hex entry)

```

Note that you *must* place a '0' byte after your filename text string, as a terminator to indicate 'end of command string' to either CP/M-86's CCP or SB-80's .CLI. I didn't bother in the SB-80 example, because it's already zeroed out for us.

```

:d<cr> <-- dump the sector to confirm that it's correct
00 E92A03E9 2103E902 037F0458 44495200 *i*.i!.i....XDIR.*
10 20202020 20202020 20202043 4F505952 * COPYR*
20 49474854 20284329 20313938 302C2044 *IGHT (C) 1980, D*
30 49474954 414C2052 45534541 52434820 *IGITAL RESEARCH *
40 20000000 00000000 00000000 00000000 *.....*
50 00000000 00000000 00000000 00000000 *.....*
60 00000000 00000000 00000000 00000000 *.....*
70 00000000 00000000 00000000 00000000 *.....*

```

```

:w<cr> <-- write it back to disk...

```

If you like, you can insert a command string clear to the end of the sector (but not beyond!), for some really complicated purposes like assembly invocations with various 'switch' parameters. This is true for either CP/M-86 or SB-80; just don't 'patch' into the next sector, as there is executable code there! Let's see, where were we?

```

:x<cr> <-- exit DU, and return to CP/M-80...
A>era b:cpm.sys<cr> <-- erase the "old" CPM.SYS
A>pip b:cpm.sys<cr> <-- back to whence we came...

```

Note again, that the 'CPM.SYS' file should also be the first file on your CP/M-86 just as 'SYSTEM.CLI' was for SB/80, and for the same reason!

AUTOLOAD For CP/M-80 On The Osborne I

Here is a handy little assembly language programming trick, to force your Osborne I computer to execute the AUTOLOAD/START function (in this case, built into the CP/M-80 operating system and the Osborne I BIOS), in three different ways.

Edit, assemble, and load the simple program described below with ED, ASM, and LOAD provided on your CP/M-80 utility diskette (make sure that you name it AUTOST.ASM, when you edit it!). Then press RESET on the front of your Osborne I computer, to AUTOLOAD/START in one of three ways described in the following sections.

Osborne I - AUTOSTART '0'

If the label 'auto' is equated (see the assembly language program that follows) to a value of '0', your Osborne I will sign-on by first clearing the screen, and then it will display the message:

```

Osborne Computer System
60K CP/M 2.2A

```

and then display the CP/M operating system 'A>', waiting for your command input from the keyboard.

Osborne I - AUTOSTART '1'

If 'auto' has a value of '1', it will 'loop' continuously; that's because it has nothing better to do! But, if you include a 'special application' (written in assembly language) just before the 'jmp base', every RESET or 'warm boot' with keyboard CTRL-C (CTRL and C pressed simultaneously) will cause your application program to run again (and again, and...).

Osborne I - AUTOSTART '2'

If 'auto' has a value of '2', no auto start will occur, and the system will immediately execute the CP/M-80 operating system, displaying only the 'A>' prompt...nice, if you are tired of waiting out the lengthy sign-on with the 'standard auto start' provided by Osborne Computer Corporation.

Also, I want to point out that 'patching' any BIOS or portions of an operating system "on-the-fly" is undesirable from the standpoint of future system integrity. Any changes or enhancements (always meaning 'bug fixes') performed by the supplier of your operating system or BIOS may cause displacements of the code area that your 'patch' program previously modified correctly, with *unpredictable* results (always meaning 'a feature' if it can't be fixed); this happens because your little 'trick' is now modifying something else in the new code, and has the wrong address locations for that portion of the code that you really wanted to 'patch'!

Here is the assembly language 'patch' (now if I only practiced what I preach!):

```

;
; --- Auto Start Patch Program for Osborne I Computer ---
;
; org 100h
;
; base equ 0 ; base of CP/M system memory
;
; auto equ 244h ; auto start vector offset
; WARNING: this value may have to change
; on any new releases of the BIOS or CP/M
;
; autost equ 0 ; auto start control value, where:
; 0 = Osborne sign-on message
; 1 = load AUTOST.COM on 'cold/warm boot'
; 2 = load AUTOST.COM on 'cold boot' only
;
; note: when using value 1, AUTOST.COM will
; 'loop' continuously unless it is
; linked to load an additional file
; BEFORE the 'jmp base'
;
;
; lhd base+1 ; get 'warm boot' vector address to jump table
; lxi d,auto ; get auto start vector offset
; dad d ; make vector address pointer
; mvi m,autost ; force to auto start control value
; ; <-- insert additional code here, if using
; ; a value of 1 for auto start
; jmp base ; do 'warm boot'
;
;
; end

```

Applications For AUTOLOAD

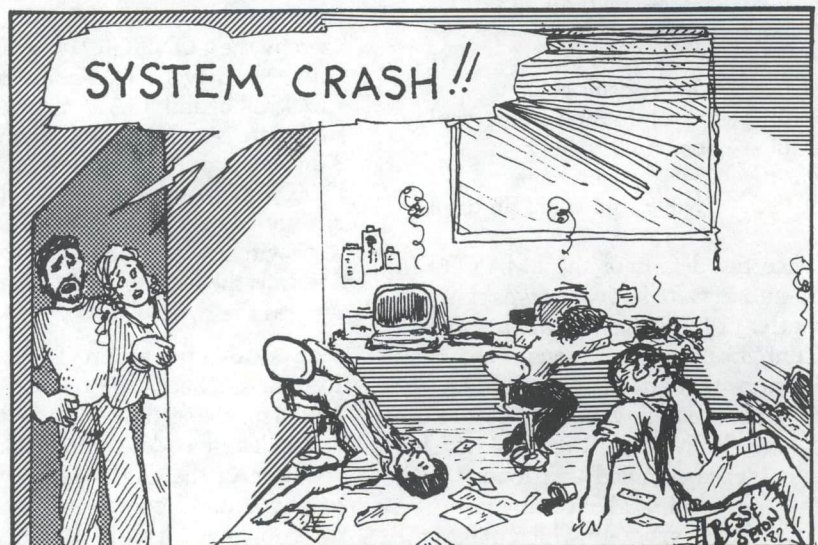
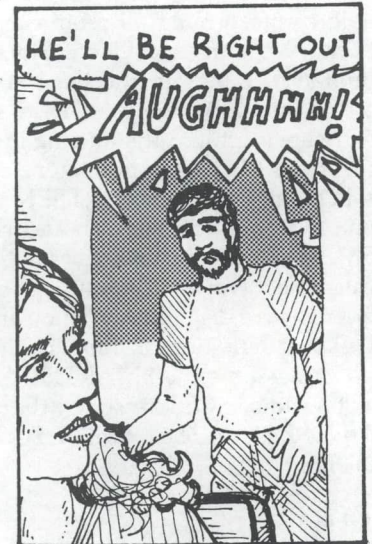
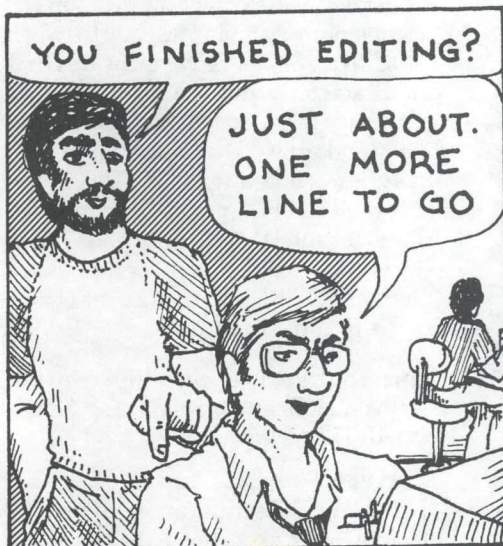
Tired of the kids playing that all-time favorite game 'ERA *.*' on your games diskette? Well, just AUTOLOAD your favorite flavor of BASIC and the game they want to play - the little monsters will never see the 'A>' system prompt.

Do you need some special system hardware initialization to occur (perhaps for printer port set-up) at 'cold-boot' without adding it to your BIOS? AUTOLOAD whatever special set-up you require!

Want to have a customized sign-on for each disk in your library, to clearly identify its usage? AUTOLOAD is a message output routine that tells you what the disk is for!

The possibilities are endless...if you think of something clever, write *Lifelines/The Software Magazine*, and share your ideas with us!

KIBITS



SMARTERM Inverse Video In CP/M-80 For The Apple

Lou P. Rivas

Without discussing the relative merits of one 80-column card for the Apple or another, let's address the user with an Apple, a SMARTERM 80-column card and the Microsoft Z80 SoftCard. After all, an 80-column card really enhances Apple Pascal; and once you have the card, adding the SoftCard is a very inexpensive way to discover whether there is anything to what your S-100 friends have been preaching - as they take turns playing their favorite games on your Apple!

Under CP/M, the SMARTERM is capable of performing all required functions except inverse video. Since the limitation is not in the hardware, it seemed necessary only to "write some software" to fill in the missing inverse function. This turned out to be easier said than done: the SMARTERM manual, which is excellent in describing SMARTERM's features, does not indicate *how* it works. The results of many hours with a disassembly listing are the I/O port definitions in Table 1. The addresses assume the SMARTERM is in slot 3 and being addressed by the 6502. To the Z80, the 6502 location \$C0B0 is OE0B0H.

Table 1
SMARTERM I/O Ports

\$C0B1	- Set 80 column GRAPHICS mode
\$C0B5	- Screen read latch
\$C0B6	- 6545 register request
\$C0B7	- Switch to 80 column display
\$C0BA	- Screen write latch
\$C0BE	- 6545 register write
\$C0BF	- Switch to 40 column display

Once the details of the SMARTERM operation were known, it was simply a matter of following the Microsoft I/O Configuration instructions in chapter 2 for non-standard peripherals. The I/O driver listed below was typed into the source file SMARTERM.ASM with ED and then assembled and loaded with the ASM and LOAD commands. The CONFIGIO program which makes the patch permanent was RUN from MBASIC and it presented a menu of

five choices. First enter a "3" to "Load User I/O Driver Software" and enter "SMARTERM" when the prompt "Source File Name?" appears. Respond "Y" to the "WARNING: A patch has already been made" prompt - there is an undocumented patch already in the space reserved for patches to the CONSOLE device. The main menu is now shown again. Select item "4" to "Read/Write I/O Configuration Block" and answer "W" to the "Read or Write" prompt. The main menu is displayed one last time - enter "Q" to return to MBASIC and "SYSTEM" to return to CP/M.

If you have already installed a custom I/O driver with CONFIGIO for some other device, it may be necessary to first read the I/O Configuration Block with menu item 4. The read or write Configuration Block reads or writes the entire block - that is, all the custom I/O drivers at one time. If one of those drivers uses self-modifying code, as this one does, the copy in memory may not be in the same form as the "boot" copy on the disk.

The various parts of the program listing are explained as follows. The EQUates at the beginning of the listing define the location of the SMARTERM or CP/M fields to be referenced. The constants before label CONSOLE are used by the CONFIGIO program to install the driver. The code between CONSOLE and ISALS checks to be sure there is a SMARTERM card in slot 3 and just continues to the "standard" CONSOLE routine if no SMARTERM is found. The standard CONSOLE routine switches from Z80 to 6502 control and calls the SMARTERM firmware to process the byte.

The code between ISALS and INVFLAG checks the byte to be processed for the set normal and set inverse codes. These codes can be changed by the user, in the CONFIGIO program, to any unused control character that does not have a lead-in. If the byte to be processed is one of these two, the code branches to either RSETIT or SETIT to

change the instruction at INVFLAG. Note that the commented Z80 JR instructions, which ASM will not assemble, are "hand-coded" by the use of the DB. If the output byte is not a set inverse code, processing will continue at INVFLAG. If the byte is not to be in inverse, INVFLAG will be a JMP to the standard CONSOLE routine in CP/M. If the byte is to be in inverse, INVFLAG is a LXI instruction to load the address of the standard CONSOLE routine into register H.

The code between INVFLAG and GOCPM ensures that the card is in the normal output state. The code between GOCPM and NOTCNTL ensures that the character is not a control. If either of these is false, the character is sent to the standard CONSOLE routine. Finally, the code after NOTCNTL and before RSETIT "pokes" the inverse character into the video memory then sends a "forward cursor" command to the SMARTERM to advance the cursor to the next display location.

This routine was developed using CP/M 2.20B and version 1.1 of the SMARTERM firmware.

Renew

If your subscription began last July, we're expecting to hear from you very soon. If you don't take time to renew now you'll miss some vital information - software is developing more rapidly than ever, and you will face some important decisions about operating systems, the new fourth generation software, telecommunications programs and other new products on the market.

So fill out that renewal form and questionnaire you've received in the mail. 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.


```

;-----
;
; SMARTERM.ASM - Patch to TTY: for ALS normal/inverse display ;
;
; WRITTEN BY LOU P. RIVAS, DECEMBER 1981
;-----
LXI D,NICODES ; AND WHAT TO STORE
CMP H
JR Z,SETIT ; BIF SET NORMAL
DB 28H,SETIT-1-$
CMP L
JR Z,RSETIT ; BIF SET INVERSE
DB 28H,RSETIT-1-$

ORIGIN EQU 0F300H INVFLAG JMP 0000 ; TO CP/M OR "LXI H"
ORG 00100H OLDTTY EQU $-2

LDA ALSTATE ; LOAD SMARTERM STATUS
ORA A ; CHECK FOR STATE 0
JR Z,GOCPM+1 ; TO CP/M IF IN MULTI-BYTE SEQ.
DB 28H,GOCPM+1-1-$ ; TO CP/M

R6545 EQU 0E0B6H ; 6545 REGISTER REQUEST ADDRESS
FRMCARD EQU 0E30BH ; FIRMWARE CARD SIGNATURE
ALSCARD EQU 0E30CH ; SMARTERM CARD SIGNATURE
SLOT3 EQU 0F3BBH ; SLOT 3 CARD TYPE BUFFER
SETNORM EQU 0F3A6H ; HARDWARE SCREEN FUNCTION
SETINVR EQU 0F3A7H ; TABLE INTENSITY ENTRIES
NICODES EQU 021C3H ; VALUES FOR INVERSE/NORMAL
ALSTATE EQU 0F77BH ; SMARTERM STATUS BYTE

DB 1 ; ONE PATCH
DW ORIGIN
DW LENGTH

DB 2 ; PATCH TYPE 2
DB 4 ; PATCH VECTOR 4
DW OLDTTY+OFF$ ; OLD VECTOR GOES HERE
DW CONSOLE+OFF$ ; THIS IS NEW VECTOR

CONSOLE LDA FRMCARD ; FIRMWARE CARD SIGNATURE
MOV D,A
LDA ALSCARD ; SMARTERM CARD SIGNATURE
ORA D ; 01 OR 81 IFF ALS CARD
CPI 81H
; JR NZ,INVFLAG ; BIF NOT ALS CARD
DB 20H,INVFLAG-1-$

ISALS MOV A,C ; COPY OUTPUT BYTE
LHLD SETNORM ; GET CODES FROM HARDWARE TABLE

LXI D,NICODES ; AND WHAT TO STORE
CMP H
JR Z,SETIT ; BIF SET NORMAL
DB 28H,SETIT-1-$
CMP L
JR Z,RSETIT ; BIF SET INVERSE
DB 28H,RSETIT-1-$

INVFLAG JMP 0000 ; TO CP/M OR "LXI H"
OLDTTY EQU $-2

LDA ALSTATE ; LOAD SMARTERM STATUS
ORA A ; CHECK FOR STATE 0
JR Z,GOCPM+1 ; TO CP/M IF IN MULTI-BYTE SEQ.
DB 28H,GOCPM+1-1-$ ; TO CP/M

MVI A,80H
ORA C ; OUTPUT WITH HOB ON
CPI '+80H
JR C,GOCPM ; TO CP/M IF CONTROL CHAR.
DB 38H,GOCPM+100H-1-$

NOTCNTL STA R6545+4 ; DATA TO OUTPUT LATCH
STA R6545+8 ; AND 6545 DUMMY REGISTER
PUSH H ; SAVE CP/M DRIVER ADDRESS
LXI H,R6545 ; 6545 REQUEST REGISTER
MVI M,1FH ; REQUEST R31
STATUS MOV A,M ; GET STATUS
ANI 80H ; AND ISOLATE UPDATE STROBE
JR Z,STATUS ; BIF NOT UPDATED YET
DB 28H,STATUS+100H-1-$
POP H ; RESTORE CP/M DRIVER ADDRESS
MVI C,1CH ; FORWARD CURSOR CONTROL CHAR.
PCHL ; GO TO CP/M DRIVER

RSETIT MOV D,E ; INVFLAG <- "LXI H"
SETIT MOV A,D ; INVFLAG <- "JMP"
STA INVFLAG+OFF$
RET

LENGTH EQU $-CONSOLE

```

Software Notes

Tips & Techniques

Michael J. Karas sent in this tip on high speed cursor address conversion.

Many currently popular CRT terminals on the market use a new type of cursor addressing that is ANSI compatible. Typical terminals include DEC VT-100, Callan Data Systems CD-100, and Televideo 950. The direct cursor addressing scheme utilized is quite a departure from the old standard method used by the ADM-3A, BEEHIVE 100, or TVI-912. The "normal" old way to position the cursor was to send a sequence to the terminal like:

```
ESC,Y,row,col
```

where the row and column are characters that correspond to specific columns and rows of the CRT screen. Typically the sequence "ESC,Y,space,space" caused cursor position to the home (0,0) screen position. Row and column positions for other areas of the screen corresponded to ASCII characters in the ASCII collating sequence starting at the space code (020H).

The new ANSI compatible format varies slightly from terminal to terminal but as an example consider the CD-100 unit. Here the cursor is positioned with the sequence:

```
ESC,[,rr,;,cc,H
```

In this case "rr" and "cc" represent the row number and column number in ASCII numeral characters. For example to speak of row 13 then "rr" consists of the ASCII characters "1" followed by "3" (or in hexadecimal 031H followed by 033H). The short program given below is an example 8080 assembly language routine that converts a binary number pair in the (HL) registers of col/row into the required ASCII sequence to position the cursor on a VT-100 or CD-100 type terminal. An inline coding scheme is used to transmit the fixed character portions of the sequence. A zero byte indicates "end of sequence". This was done to permit the same subroutine to be used to transmit other code sequences to the terminal for clear screen, delete line, etc.

```

;
;
;
;

```

```
CURSOR POSITION TO (ROW=L) AND (COL=H) LIKE MICROPRO'S WORDMASTER
```

(continued next page)


```

CURADDR:
    PUSH     H           ;SAVE ROW COL CODE
    CALL     SEQOUT      ;SEND INITIAL ESC,[
    DB      1BH,'[' ,0
    POP      H           ;GET ROW CODE
    MOV      B,L
    PUSH     H
    CALL     BINASC      ;SEND AS ASCII 2 CHAR NUMERALS
    CALL     SEQOUT      ;SEND ROW/COL SEPARATOR
    DB      ',' ,0
    POP      H           ;GET COL CODE
    MOV      B,H
    CALL     BINASC      ;SEND AS ASCII 2 CHAR NUMERALS
    CALL     SEQOUT      ;SEND TRAILER CHARACTER
    DB      'H' ,0
    RET

```

```

;
;
;SEND INLINE BYTE SEQUENCE TO THE CONSOLE TERMINAL
;

```

```

SEQOUT:
    XTHL                    ;GET SEQUENCE POINTER
SEQOT1:
    MOV      A,M
    INX     H
    ORA     A           ;IF ZERO THEN SEQUENCE IS
    JZ      SEQXIT      ;...DONE
    PUSH    H
    CALL    CHAROUT     ;SEND CHARACTER TO CRT
    POP     H
    JMP     SEQOT1

```

```

SEQXIT:
    XTHL                    ;SET STACK RETURN AFTER
    RET                    ;...SEQUENCE

```

```

;
;
;CONVERT BINARY NUMBER IN (B) TO TWO ASCII NUMERALS
;SENT TO CONSOLE TERMINAL
;

```

```

BINASC:
    XRA     A           ;INITIALIZE 00 BCD NUMBER
    INR    B
BINAS1:
    INR    A           ;LOOP TO CONVERT BINARY TO
                    ;TWO DIGIT BCD
    DAA
    DCR    B
    JNZ    BINAS1
    PUSH   PSW         ;SEND THE MOST SIGNIFICANT
    RAR
    RAR
    RAR
    RAR
    ANI    0FH
    ADI    '0'
    CALL   CHAROUT
    POP    PSW         ;SEND THE LEAST SIGNIFICANT
    ANI    0FH         ;BCD DIGIT AS ASCII CHAR
    ADI    '0'
    CALL   CHAROUT
    RET

```

```

;
;CHAROUT:
; INSERT CODE HERE TO TRANSMIT CODE TO THE
; TERMINAL.
    RET

```

```

;
END

```


Criteria For Evaluating Application Development Software

Steve Patchen

Last month I introduced some application development concepts and a menu system to help organize development of application software in the dBASE II system.

These ideas are part of a new generation of software emerging in the market. Such products are generally referred to as fourth generation software. Many products in this generation are not complete application development systems, but instead are elements of a diverse set of tools which help with some part of the development process. (Editor's Note: *Lifelines/The Software Magazine* intends to review more software in this fourth generation.) In order to put these products into perspective and provide some grounds for comparison of diverse packages, I will describe some criteria which will relate the review subjects to the application development process.

Prior to this fourth generation, computer software evolved through three stages of development. The first stage dealt with the computer on the bits and bytes level of each particular machine. The second generation was formed by the use of symbolic references to machine operations and addresses in assembler languages. The third generation of software emerged as an attempt to provide some machine independence for software with languages like FORTRAN and PL/I and with operating system environments.

This third generation of software is still very volatile, however. Many computer applications have dynamic and changing structures and others are so complex that it is difficult to design them completely without prototyping models of the system first. Using third generation languages makes both of these situations expensive. Third generation languages leave a large gap between the logical structures used in the final programs and the structures manipulated by the languages. Systems created with these languages thus have a wide variety of underlying structures.

It is only in the last few years that computer professionals have begun to examine this area of diverse structures and have begun to put some order to it.³

Fourth generation languages and development systems more strenuously address the logical structures which allow software development to be tailored closely to a specific application's problems. Because the real world incorporates a wide range of possible computer applications, a system structured to serve some applications well might not fit other applications at all. So we expect fourth generation software to be restricted to smaller domains of use than third generation languages are, implying that there will be even more fourth generation languages than there are third generation languages.

The advantages we gain by restricting this domain of use for development systems are faster and cheaper development, implementation by users expert in the area of interest but with little or no experience as programmers, a prototyping capacity which can reduce study and analysis costs, and decentralization of data processing to put computation power where it is needed.⁵

Some of the types of structures which provide this higher level of treatment in development of software are aggregate data structures along with associated aggregate operators, implicit control structures which allow specification of *what* is to be done without concern for *how* it is done, and associative referencing of information and structures.⁴ An aggregate data structure is any data structure which can be treated as a single unit but which might actually be composed of several dissimilar smaller structures. Aggregate operators are operators which can perform the same operations on these structures regardless of the actual composition of the structures. Implicit control structures are facilities providing the *how* so that functions can be performed when they are specified in a form describing only

what is to be done. Associative referencing uses some form of an expression which depicts what is referred to without regard for how it is to be found. These high level structures thus relieve the user or developer from dealing with those details not logically related to the problem he or she is dealing with.

An example of an aggregate structure could be an invoice data structure. An invoice is composed of several logical groups of data: there is information about the customer to whom the invoice is issued; there is information about the shipping of the products invoiced; there are descriptions of the products and prices, and there is billing information about taxes, total due etc. An example of an aggregate operator might be "DISPLAY INVOICE FOR INVOICE NUMBER='820412'". This operation would find and display an invoice identified by the specified number. This operation could be used interactively or it might be imbedded in a program. As part of a program the actual invoice number would have to be retrieved from some other source for each use of the statement. The DISPLAY operator is also an implicit control structure because it takes actions based upon assumptions or stored knowledge about the entities specified. Associative referencing is used by this statement because it refers to the invoice by specifying a value one of its data sub-structures must contain.

This expression seems like a natural way to request an operation of the computer because it talks in terms which are related to the problem being worked on and because it uses natural language-like phrasing. However, if we examine this example more closely, we must realize that the computer could not do anything without knowing what an INVOICE or an INVOICE NUMBER was. It must thus be either pre-programmed to recognize these terms or it must have knowledge of these terms, and implicit control structures to make decisions based upon the knowledge it has.

(continued next page)

These two possibilities reveal two different approaches to fourth generation software. The first is called very high level language (VHLL) and the second is called knowledge-based generation (KBG).⁴ VHLL's are usually evolved from third generation languages like FORTRAN while KBG's derive their experience from artificial intelligence and languages like LISP. There is another technique for program generation called the build program technique.⁶ It uses third generation languages and program templates to build programs from information converted by a specification language into a form which can be substituted into the templates. The pieces are then put together and form programs. This type of application generation is the most restrictive because it depends upon the templates being appropriate to the functions they are required for, meaning they are usually developed to fit a narrow range of needs. An absolute necessity for this type of system is the ability to extend the development facilities by adding more templates and incorporating code created the old way.

Extensibility and the ability to link to program structures written in another language are actually very important requirements of any fourth generation development system. Even more versatile systems can box the user into a corner during development, because of some unplanned-for feature, or one outside the scope of the development system yet within the scope of the application.

In addition to changing the level between the machine and the application at which the designer must work, there is an increasing concern for the interaction between the immediate user and the computer. This applies to both the designer of systems *and* the end user. So this interface must address the psychological effects of the system on users and designers. And it must cope with the efficiency of tasks the user or designer employs to accomplish his goals. Every computer system still has to deal with the four interfaces: the machine, the machine independent environment, the application and the user. First and second generation software had to handle all the details of all four of these interfaces for each application. Third generation systems relieved the user to some degree of the machine environment problems, but the user still must

deal with the other environments on a low level.

Table R5 covers several important questions for each of these four views of computer software. Evaluation of software which participates in the development process should be reviewed by examining it in light of these questions. The table groups the questions under the four interfaces which tie the computing environment together. Some of these questions involve the relationship of two or more of these interfaces and imply a test on the modularity and cohesion of the different subsystems of the computing environment. I will discuss them below.

Application Suitability

The only thing which should be visible during application specification is the application, and the entities which are logically a part of it. The first question in this section asks how easy it is to fit the application to the computing system in a logical manner - without the distraction of unrelated details. Question two asks how completely the development system can specify the logical requirements of the application problem.

Both of these questions imply the need for good modular separation between the specification and implementation parts of the development system; they also imply that the specification model must be able to depict the desired behavior of the implemented system in terms of functional responses to various stimuli.

Question three can be satisfied by an operational specification system which allows a "walk-through" of the specification, or permits interactive implementation and testing of incomplete parts of the system. The specification must be insensitive to incompleteness. Question four also requests dynamic structures which allow reworking and extension of the specification.¹

Implementation Concerns

The need for optimization is most apparent at the implementation interface. The machine environment limitations have to be handled at this level so that unrelated ramifications are not

reflected at the application level. It is this interface which must distribute the resources of the computer to the requirements of the application. How the limitations and constraints of the machine are reflected in the application must be clearly visible. The second question about this part of the environment deals with the tools used not only for the actual implementation, but also for optimizing the machine's use and dealing with extensions of the implementation environment.

It is at this level that we are most concerned about how the reviewed product relates to the operating system and what form data structures and operations have.

Needs Of The User And Designer

When a system is designed, the user's needs have typically been given low priority, despite pretenses to the contrary. This has made the data processing center a frequent battleground; order is imposed upon data processing by brute force. As in most wars, the pressures feeding the turmoil are economic ones. However, we are beginning to understand the data processing environment a little better and it is possible to make convincing economic arguments for considering the psychological part of the environment. The three questions in this section deal with only some of the obvious points revealed so far. These questions should be asked for both the development part of the system and the use of the final products.⁷

The Computer Machine

Computer system capacities are frequently exceeded many times over the life of the system. This means that the machine environment must be extended during the use and development of applications and that it is desirable to do so without a great deal of disruption to the operating environment. The questions in this section are concerned with the visibility of machine limitations and the ability of the system to adjust to machine expansions. Portability of the system to another machine is also an important concern, so that the user is not constrained by the current machine's expansion limits.

Application Structure

In addition to these four logical divisions of the data processing environment, there are identifiable functional segments of the application system. In general, a business application involves data entry, data management and reports or other output. There is more than one possible level of complexity for each of these parts. That is, data entry may involve simple appending of new records to a file structure or it may involve entry into a set of inter-related transactions. Data management may be simple file management or it may involve a complete database management system. Reports can vary from simple listings of files to complex derivations from large databases. Within the reporting and data management systems there can also be simple or complex algorithms or functions to perform transformations upon data.

There already exists a variety of both manual and automated techniques to assist in the development of various parts of the application. Individual programs are assisted by decision tables, flowcharts and function libraries. Report design is enhanced by output-input matrices, data dictionaries and data flow diagrams. Database management system data design can utilize data dictionaries and automated data design systems. Transaction processing is implemented with the aid of HIPO diagrams, data dictionaries and data flow diagrams. There are also manual design techniques for completely integrated business systems; they usually boast such names as structured analysis² and structured design.⁹ Software packages which attempt to do everything are usually called 'applications generators', 'parameterized application packages', or 'application development systems'.

Many of these development systems provide good facilities for part of the application, but are weak in dealing with other elements.⁸ The usefulness of any tool is dependent upon its ability to handle the particular problem at hand. Therefore, if the systems available are biased, a system slanted towards the application must be chosen. If a combination of tools is required because an adequate complete system is not available, the compatibility of the various tools on hand must be considered. Many tools only attempt to extend the

use of third generation languages by simplifying some frequently-performed task or by the creation of some frequently-used structure. This approach to creating a fourth generation development environment is a "bottom up" evolution of tools. It must be examined from a "top down" perspective to keep the development process requirements in view. Many of these extensions improve programmer performance, but do not provide any enhancements to the application specification part of the system. Some development tools may not differentiate between any of the four interfaces with which they deal. It is important to note these variations, because they can affect the flexibility and usefulness of the tool in question.

Table R6 attempts to provide a more complete picture of the development system from the viewpoint of the application structures. It allows the reviewer to plot the completeness and complexity for each part of the functional divisions of the business application. A facility is considered to exist if it is possible to specify *what* is desired without writing code in a procedural language. If a procedural language is provided as an integral part of the development system, and its language features reflect the structures used by the system, the reviewer might consider the system capable of complex systems; however, it is not easily capable of them.

Summary

Important questions must be asked by anyone who wishes to use data processing for business needs. What are the requirements of the application I wish to implement and how do I satisfy these requirements? A review of any software which participates in the development process must attempt to provide the reader with ways to answer these questions. Many purchasers of development software will have little or no experience with business system development. They need to be informed of the tools they will need and the preparations they will have to make before attempting any such task. Product manuals seldom make any attempt to orient the system user to the development process. They do not inform the user what other resources and techniques will be needed in addition to those provided by the system. The

reviewer can help the software buyer by putting a product into perspective and relating it to other products on the market. The reviewer must identify the range of applications for which the product is suited, as well as those for which it is not suited.

1) Balzer, R.B. and Goldman, N. "Principles of good software specification and their implications for specification languages", AFIPS Conference Proceedings Vol. 50 1981, NCC, AFIPS Press, Arlington, Virginia 1981, pp. 393-400

2) DeMarco, Tom, Structured Analysis and System Specification, Yourdon Inc. New York, 1978

3) Dolotte, T.A., Bernstein, M.I., Dickson Jr., R.S., France, N.A., Roseblatt, B.A., Smith, D.M., Data Processing In 1980-1985; A study of Potential Limitations to Progress, John Wiley & Sons, New York 1976

4) Hammer, M. and Ruth, G., "Automating the Software Development Process", Research Directions in Software Technology, The MIT Press Cambridge, Mass. 1979, pp. 767-792

5) Martin, James, Application Development Without Programmers, Prentice-Hall, Englewood Cliffs, N.J. 1982

6) Rice, John G., Build Program Technique: A Practical Approach for the Development of Automatic Software Generation Systems, John Wiley & Sons, New York 1981

7) Shneiderman, Ben, Software Psychology: Human Factors in Computer and Information Systems, Winthrop Pub., Cambridge 1980

8) Whitney, V.K. and Morse, J.G., "Choosing application development tools and techniques", AFIPS Conference Proceedings Vol. 50 1981, NCC, AFIPS Press, Arlington, Virginia 1981

9) Yourdon, E. and Constantine, L.L., Structured Design, Yourdon, Inc., New York 1975

See next page for Tables.

TABLE R5
Application Generation Systems

- I. APPLICATION SUITABILITY:**
1. Does the method required for specifying applications reflect an understandable and logical model of the domain of applications for which it is intended?
 2. Can the application be completely specified?
 3. Is the implemented system testable against the specification for the system or vice versa?
 4. Does the development system make it easy to extend and rework the specification and implementation?
-
- II. IMPLEMENTATION SUITABILITY**
1. Are the restraints and limitations of the implementation environment made clear to the designer?
 2. Is the set of tools for doing system implementation complete or are other independent tools required?
 3. Is the implementation environment extensible to include new components or components from other systems?
-
- III. USER/DESIGNER SUITABILITY**
1. Are the user interfaces developed by the system and those used to develop the application understandable in terms of the tasks to be performed or do unrelated details obscure the operation?
 2. Does the user get a feeling of having complete control of the system or do obtuse messages and unexplained operations leave him in confusion or frustration?
 3. Does the system seem to have been designed with psychological criteria for short term memory, closure of tasks, response time and user control in mind?
-
- IV. MACHINE SUITABILITY**
1. Are limitations imposed by the machine environment understandable in terms of application limitations and are application requirements translatable to machine requirements?
 2. Are any provisions made in the development system to allow optimization in different machine environments?
 3. Is it possible to extend the machine environment without major changes to applications already implemented?

TABLE R6
Application Development Facilities

Functional Parts	Completeness and Complexity of Facilities			
	Little or None	Some	Complete & Complex	Easily Complex
Individual Program Development				
Input Transactions				
Data Management				
Reports and Queries				
Integrated Systems				

Software Notes For COBOL-80 Users

The following are the current CRT DRIVER modules for COBOL-80.

CDADDS.MAC ADDS REGENT TERMINALS
 CDADM3.MAC LEAR-SIEGLER ADM-3A TERMINALS
 " LIFEBOAT CP/M (prior to 2.25) for TRS80 II COMPUTERS
 CDADM31.MAC LEAR-SIEGLER ADM-31 TERMINALS
 " LIFEBOAT CP/M for DATAPOINT COMPUTERS
 " LIFEBOAT CP/M 2.25+ for TRS-80 II COMPUTERS*

CDANSI.MAC ANSI STANDARD TERMINALS
 CDBEE.MAC BEEHIVE and CROMEMCO TERMINALS
 CDHZ15.MAC HAZELTINE 1500 TERMINALS and ARCHIVE COMPUTERS
 CDISB.MAC INTERTEC SUPERBRAIN TERMINALS and COMPUTERS
 CDPERK.MAC PERKIN-ELMER TERMINALS
 CDSROC.MAC SOROC IQ TERMINALS
 CDWH19.MAC HEATH/ZENITH TERMINALS and COMPUTERS
 CDZEPH.MAC ZENTEC ZEPHYR TERMINALS

* Note: See also an article in the April '82 issue of 80 Micro-computing by James Korenthal for an alternate TRS-80 model II screen driver. It will only work with CP/M versions prior to 2.25, but includes information on how to turn the cursor on and off. It also includes some ideas on putting a bell up on the silent screen.

Volume 81, Catalogue and Abstracts

CP/M Users Group

Catalog

DESCRIPTION: CP/M Utility disk. Submit replacement, editor, text processor, hard disk backup utility, etc.

NUMBER	SIZE	NAME	COMMENTS
	3K	-CATALOG.081 ABSTRACT.081	Contents of Vol. 81 Abstract of files on volume 81.
81.1	10K	AUTOLOAD.COM	Write initial CP/M command
81.2	3K	AUTOLOAD.DOC	into CCP on disk.
81.3	28K	BACKUP.ASM	Back up hard disk to multiple
81.4	7K	BACKUP.DOC	floppy disks
81.5	7K	BAUDSET.ASM	Set baud rate for
81.6	2K	BAUDSET.DOC	serial board
81.7	25K	EDITM.ASM	Update of CPMUG volume 16
81.8	3K	EDITM.COM	editor, with new
81.9	4K	EDITM.DOC	features.
81.10	16K	FLOPCOPY.ASM	Copy floppy via hard disk
81.11	2K	FLOPCOPY.DOC	when only 1 floppy drive
81.12	42K	POW2.ASM	"Processor of Words" - text
81.13	5K	POW2.COM	processing prog.
81.14	4K	POW2.DOC	" "
81.15	23K	POW2.MAN	Manual on above
81.16	2K	POW2.TST	Test document
81.17	22K	SUPERSUB.ASM	Super submit program,
81.18	3K	SUPERSUB.COM	allowing nested submits,
81.19	15K	SUPERSUB.DOC	etc.
	--K	FILES.CRC	CRC of files on this disk
	2K	CRCK.COM	Produce CRC of files
	5K	U-G-FORM.LIB	CPMUG submission form

type on the printer, restricts the files it backs up (no .BAK, .SYM, etc.), and then does the actual backup. It reports what files are on each disk, etc. Very complete.

BAUDSET.ASM, .DOC is John M. Kodis' program to facilitate baud rate changing on a Cromemco Tu-Art board or a Micromation Doubler board. The program could be modified for other UARTs.

CRCK.COM is Keith Petersen's program - checks all files.

EDITM.ASM, .DOC, .COM is an update of CPMUG volume 16 editor, claimed to be faster than ED.COM, and has additional ability to write arbitrary lines to disk. Useful only if ED is your only editor, and you want a bit more.

FLOPCOPY.ASM, .DOC is Gary Young's program to copy a floppy disk, using a hard disk as intermediate storage. This would be used by someone with a hard disk, but only a single floppy, who wanted to copy a floppy.

POW2.ASM, .COM, .DOC, .MAN, .TST comprise a text formatter, "Processor Of Words". By Herman Watson, from Dr. Dobbs Journal No. 29, page 20. POW2 revised from CPMUG Volume 36 by William R. Brandoni. Enhancements: CP/M-80 base alterable for non-standard systems; new User's Manual; files bigger than memory; bug fixes; viewing the formatted output on a CRT terminal; :MD, :CD, and :CU commands are provided to double strike and underline. Uses C/R overprint technique.

SUPERSUB.ASM, .DOC, .COM is Ron Fowler's replacement for SUBMIT. Allows SUB file nesting, and also an immediate mode where no SUB file need be edited first. Written up in January, '82 *Lifelines/The Software Magazine*.

Abstracts

AUTOLOAD.COM, .DOC is Willis Howard III's program to patch a command into CCP so it will be executed automatically when CP/M-80 boots. The command is patched directly to the CCP image on the boot tracks of your disk. Useful for auto-loading an MBASIC menu program for dedicated

applications, or for loading a special driver, etc. Unless your BIOS is specifically modified to do so, the auto-command will execute on both cold and warm boot.

BACKUP.ASM, .DOC is Gary Young's very complete program for backing up a hard disk to multiple floppies. It prints a master directory by

(continued next page)

Apologies to those of you who experienced panic on not seeing "Macros Of The Month" in the last issue of *Lifelines/The Software Magazine*. Rumors of its death were slightly exaggerated.

The 8086/8088 version of PMATE has arrived in MS-DOS, PC-DOS, and CP/M-86 versions. All share some significant new features, listed below.

- 1-Disk buffering is greatly expanded. This results in faster operation and less disk access.
- 2-The terminal configuration files ("CNF" files) allow assignment of any key to any permanent macro. This had to be done in other versions of "PMATE" by modifying and assembling "IOPATCH.ASM".
- 3-Configuration options allow the editor to come up in Insert or Overtyp mode automatically. Thus a "PMATE" can be configured to (a) come up with menus or operator prompts of an arbitrary and familiar kind, (b) process a file, and (c) exit - without ever entering command mode or requiring that the operator know anything about how to operate "PMATE" on the root level. OEMs take note.
- 4-New single-keystroke commands: Overwrite/Insert mode toggle key, erase line forward/back from cursor, page up and page down, cursor to beginning/end of line.

The PC version of PMATE is particularly interesting and has mated nicely with the IBM keyboard and screen. For example:

- 1-PMATE-PC senses the kind of display in the system and adapts itself automatically to either the monochrome or color display.
- 2-The PC version is optimized for the IBM PC screen and does not rely on IBM's built-in screen-handling routines. This editor is fast!
- 3-All the function keys are implemented as user-defined macros.
- 4-All the keys on the right keypad are implemented. Besides the four cursor motion keys, the assignments are: beginning/end of line (a toggle), forward/back one screen, delete line forward/back from the cursor.
- 5-This version takes advantage of the unique code pairs generated by each key on the IBM keyboard. Backspace and control-H do not necessarily have to perform the same function, nor do control-M and return. Each key on the keyboard has a unique code and can be assigned a unique function in PMATE.

PMATE PC takes advantage of this by configuring the keypad cursor motion keys for geometric cursor motion and the control-keys for line-oriented cursor motion. Both are simultaneously available - in other versions one has to make a choice.

The macros this month are designed to speed repetitive text-processing tasks. One helps you edit a group of disk files quickly; one is designed to speed editing of cursor-addressing statements, and the last auto-loads macros into their execution buffers. These are by Andrew Hughes of Toronto, Canada.

Anybody who has had to make alterations on a group of files in one sitting has probably experienced the drudgery of loading them in one by one, having to repeatedly list their names to see which one is next. This happened to me, and I wrote a macro to do the job. Mr. Hughes wrote a much better one, which is this month's winner, listed below. The macro lists the file names for any drive on the screen and you merely position the cursor on the file name you want to edit. You can also specify the output file name, and it can be on a different drive. The macro loads the file and you proceed with the edit. When done, you can save the edited file and repeat the process. If the processing to be performed on all the files is identical, a macro can be called within a loop to do the work. You could also change this macro to do something to a whole group of (ambiguous filename) files without any operator intervention.

```
; Permanent macro called below to enter a string of characters
^Xi[gEnter: DEL : CR to end$
@k=127 [-d^][@k=13{&}@ki]}

; Permanent macro to load and process all items on a disc
; successively.
;
; Authored by Andrew Hughes of Toronto, Ontario, Canada
;
; Uses buffer 0 permanently, to hold the directory, and variable 0
; permanently, through the whole session. Variable 0 must be 0
; before calling the macro. The first time use 0v0.d$$ to call it,
; and thereafter, just .d$$
;
; Uses buffer 1 during the macro.

^Xd be@0=0 ; on first using the macro, clear buffer 0
[
xk1v0 ; and set variable 0
gEnter input drive or CR$ ; get directory from right drive (CR=A:)
@k&"="B [xsb]
@k&"="C [xsb]
xl$ i ---
$ ; end of initialization
]
lblm zblg ; move NAME to buffer 1 and
; to bottom of Directory list
blea ibtexkxf$ z-dqr ; prepare buffer 1 for execution
gEnter output drive & name or CR$
@k=13[i $@ki.i] .la ; execute buffer 1

; Here the processing program could be appended
; or a macro could be called
```

The second was written to help in editing dBASE II command files, but could be adapted for any language which includes cursor-positioning statements. It simply adds one to the number under the cursor. Suppose you have constructed a screen with cursor-positioning commands, and decide to add another line to the middle of the screen. Simply go down the list

of commands invoking the macro on the command line. This saves much time. Note, however, that the macro can handle only two-digit numbers since the largest digit for cursor addressing on most terminals is 80.

The third allows you to construct auto-loading macros, i.e., macros which are loaded into the buffers for which they were written. All that is necessary is to write the macro with a header and trailer specifying its execution buffer, as in the macro below. The only restriction imposed is that the body of the macro cannot have the string ";@end" in it, and the header must precede any other similar string in the macro. The loader routine should be made into a permanent macro. This very convenient macro allows you to put up to nine macros in a library and load them automatically into their correct buffers.

```
; Macro to add a number to the ascii digit(s) under the cursor
; starting number must be stored in v0

; @b=3 macro executes in any buffer -- 3 used here
@b=0{gYou must store starting count in value register 0 first;$}

[
s@ $ts,$-m#d ; delete old x
@0\va01
g Space to continue, any other character to quit $
(@k<32 ! @k>32)
]
```

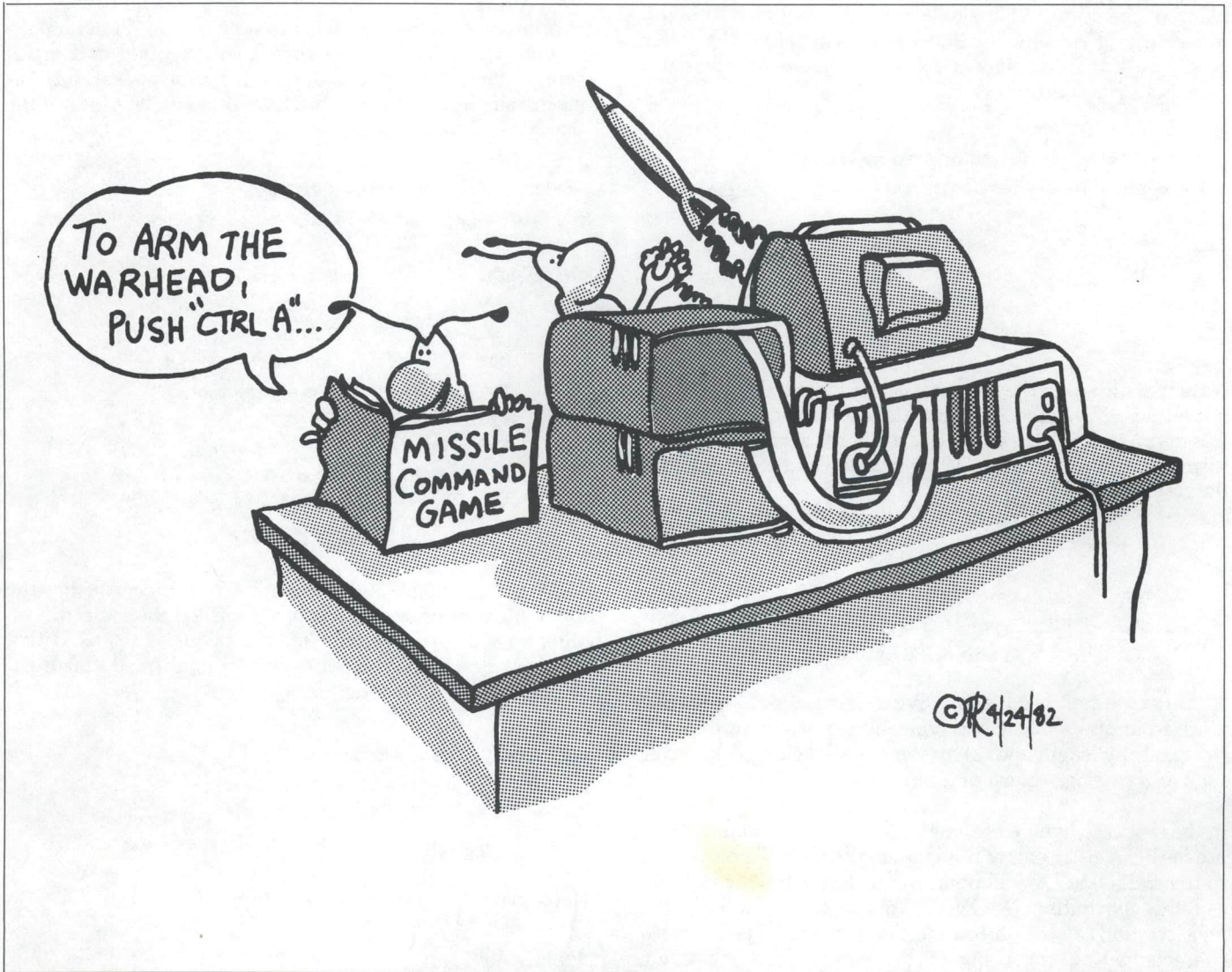
```
0v0
;
;@end
; Macro to auto-load macros

; loads a macro or group of macros into buffers specified by their headers
; format of header: @b=n, where n=0-8, and the first @b in the file is
; the "jump address", or the buffer that will be executed

; the end of each macro group has the line ";@end"

qab9eztxiAa$ ; get the file named by caller into register 9
#s;$@e_-m s;$s@b=$@tv1# ; store the "jump address"
[
es;$@e_-m t s@b=$ ; find beginning of header
@tv0 ; store the buffer number
s;@end$ ; find end of this macro
@0="#0{#b0m^} ; kludge -- this version of pdate doesn't
@0="#1{#b1m^} ; allow b@m were @n is variable
@0="#2{#b2m^}
@0="#3{#b3m^}
@0="#4{#b4m^}
@0="#5{#b5m^}
@0="#6{#b6m^}
@0="#7{#b7m^}
@0="#8{#b8m^}
]
; all moved

@1="#0{.0$^[^} ; now jump to first one
@1="#1{.1$^[^}
@1="#2{.2$^[^}
@1="#3{.3$^[^}
@1="#4{.4$^[^}
@1="#5{.5$^[^}
@1="#6{.6$^[^}
@1="#7{.7$^[^}
@1="#8{.8$^[^}
```



8080 Assembler Programming Tutorial: Subroutines

Ward Christensen

The 8080 instructions have been covered individually, and now it is time to put them to work as a team, doing useful work that no single instruction can perform. Initially I will avoid CP/M-80 specific routines, so that you will get some useful routines which apply to both CP/M-80 and non-CP/M-80 systems. There will be subroutines for data movement, arithmetic, logical, and input/output.

Data Movement Subroutines

A very simple move subroutine might be one that filled a block of memory with a single character, perhaps to blank out a buffer. Typically you know exactly how long the buffer is, so can simply decrement a count as you store the data. Here is such a subroutine:

```
;
;FILL: fills the buffer pointed to by HL,
;with spaces. The buffer length is in BC.
;
FILL  MVI  M,' ' ;store 1 space
      INX  H    ;point to next byte
      DCX  B    ;decrement count
;
```

Note that although BC has been decremented, the 8080 does not set the PSW flag bits to indicate whether or not the result was zero, so it is necessary to test BC for zero. This is performed by ORing B with C. Only if both B and C are zero will the result be zero.

```
;
MOV   A,B    ;get B,
ORA   C      ;OR it with C
JNZ   FILL   ;jmp if B or C isn't zero
RET                    ;return, BC is now zero
```

A more common need is to move a string of data from one place to another. There are typically two ways to define a string: (1) by length and (2) by some special ending character such as a carriage return or a binary 0.

In the first case, where the length is known in advance, I will use the BC register pair to hold the length. HL will point to the source field, and DE will point to the destination field. I will call the subroutine "MOVE". To use it, I just load the registers, and call the subroutine. As an example, let's move a character string consisting of the words "Test message", which is 12 bytes long, to an area called BUFFER:

```
;
;move MESSAGE (12 long) to BUFFER
;
      LXI  H,MESSAGE ;set "from" register
      LXI  D,BUFFER  ;set "to" register
      LXI  B,12      ;set length
      CALL MOVE       ;move the data
      .
      .
      .
```

```
MESSAGE DB 'Test message'
```

I purposely set up the registers this way, so they'd be compatible with the Z80 single-instruction LDIR (load and increment), which allows the Z80 to perform the MOVE subroutine in a single, two byte instruction. But first, let's look at the 8080 MOVE subroutine:

```
;
;Routine to move a string from (HL)
;to (DE), length in BC
;
MOVE  MOV   A,M    ;Get source byte
      STAX D      ;Store at destination
      INX  H      ;Bump source pointer
      INX  D      ;Bump dest. pointer
      DCX  B      ;Decrement count
      MOV  A,B    ;Get B,
      ORA  C      ;Or with C,
      JNZ  MOVE   ;loop if BC not yet zero
      RET                    ;otherwise ret from MOVE
;
```

Back to the Zilog Z80: Although I recommend using the above move routine in your programs (so they will run on either an 8080 or Z80), there may be people who are writing code strictly for their own use, and are confirmed Z80 users. They should simply code:

```
;
;Z80 specific move subroutine
;
MOVE  DB   0EDH,0B0H ;Z80 "LDIR"
      RET                    ;Return after executing
```

If you had a Z80 assembler, you'd code:

```
MOVE  LDIR                    ;do the move
      RET                    ;Return after executing
```


In either case, the "subroutine" is only a two byte instruction and a RET, so it typically would be coded inline rather than as a subroutine.

Variable Length Move

Some languages, notably the "C" programming language, use strings terminated by a special byte, rather than dealing with fixed-length strings. "C" uses a zero at the end of the string. A move routine to move such strings would be:

```

;
;variable length move subroutine.
;(HL) = input string, (DE) = output string.
;input string is terminated by byte of 00. The
;byte of 00 is copied to the output string
;
MOVEZ  MOV    A,M      ;Get source byte
        STAX   D        ;Store at destination
        INX   H        ;Bump source pointer
        INX   D        ;Bump dest. pointer
        CRA   A        ;Is (A) = zero?
        JNZ   MOVEZ    ;No, loop until done.
        RET                    ; then return

```

That's about all there is for data movement. I'll get into some specific uses for these subroutines later (for instance, for moving file names around in preparation for CP/M-80 file-related commands).

Arithmetic Instructions

You will typically be dealing with two kinds of arithmetic in 8080 assembly language: binary and ASCII. Since the 8080 only supports 8-bit arithmetic, and the 16-bit add instruction DAD, subroutines will have to be used for anything more.

Let's start with some basic binary arithmetic: ADD, SUBTRACT, and NEGATE.

```
DAD    D
```

will add the contents of DE to HL, leaving the result in HL.

16-Bit Subtraction

A subtraction is not quite as simple. It requires using an 8-bit subtract on the low bytes, then a subtract-with-borrow on the high bytes:

```

;
;double subtract DE from HL
;
DSUB  MOV    A,L      ;Get low byte of HL
        SUB   E        ;Subtract low byte of DE
        MOV   L,A      ;Put answer back
        MOV   A,H      ;Get high byte of HL
        SBB  D        ;Subtract high byte of DE
                ; with BORROW
        MOV   H,A      ;Put answer back
        RET

```

16-Bit Negate

The 8080 cannot negate a value directly. A "hack" technique for doing it on 8-bits would be to simply subtract the value from zero. Suppose the value in A was to be negated; A might contain a 3, and you might want the value -3:

```

MOV    B,A      ;Save value in B
XRA   A        ;Zero A
SUB   B        ;Subtract value
;

```

(A) now contains the negative of what it originally had.

The 8080 contains a ones-complement instruction, CMA. Ones complement simply means flipping the value of each bit. In binary arithmetic, the term "twos complement" means the same as negative. Ones complement does not produce the negative of the original value, as you can see by the following example:

```

MVI   A,1      ;A=0000 0001
CMA                    ;A=1111 1110
INR   A        ;A=1111 1111
INR   A        ;A=0000 0000

```

The point of this example is that if CMA did produce the negative of the 1 producing -1, then one INR A would have incremented it back to 0. Instead, two INR instructions were required. In general, to produce the negative of a number, a single INR A must follow the CMA. This yields a simple "negate" subroutine:

```

NEGATE CMA                    ;Get ones complement
        INR  A        ;Make twos complement
        RET                    ; (negative)

```

This is not a likely candidate for an honest-to-goodness subroutine, since it is only three bytes, but would take a three byte call instruction to call it. It thus would be called "in line", i.e. by directly coding the instructions CMA and INR A.

A more suitable candidate for a subroutine would be a 16-bit complement. Because DAD adds to the HL register pair, HL may be considered to be a 16-bit accumulator, so typically a 16-bit complement subroutine would be expected to work on HL:

```

;
;return HL=-HL
;
NEGHL  MOV    A,H      ;complement
        CMA                    ; bits in
        MOV   H,A      ; H
        MOV   A,L      ;then
        CMA                    ; in
        MOV   L,A      ; L
        INX   H        ;then make twos
                ; complement

```

The first six instructions produced the ones complement of HL. The final INX then added one to the ones complement, producing a twos complement, or negative of the original
(continued next page)

number. Note that the entire number was incremented (I used INX) and *not* just each 8-bit half of it (via INR).

ASCII Arithmetic

The most common forms of ASCII arithmetic encountered in assembly programming are:

- 1-counting in ASCII.
- 2-converting numbers from ASCII to binary
- 3-converting numbers from binary to ASCII

ASCII COUNTING: A typical use would be to count something, simply incrementing a value each time some condition is met. For example, an extended directory program may wish to count how many files it found. One means would be to maintain a binary count, then convert it to ASCII for printing. However, if you have no other need for such a conversion routine, I'm sure you'll find this simple ASCII increment routine more suitable:

First, let's define the message that is printed. I'll put a special character, '\$', at the end of the message:

```
MESSAGE DB 'There are '
COUNT DB ' zero files$'
```

Now let's define a subroutine called "ADD1", which adds one to the count. To call the routine, point HL to the *low* byte of the ASCII number, and call ADD1. It does "pretty much what you'd expect"; it simply adds one to the digit, and if the sum is greater than nine, makes it a zero, and backs up one column, and increments it. But you might be surprised to note that it simply branches to itself to do the carry, and that a very simple instruction may be used to maintain the leading spaces on the number, so it prints " 3 files" instead of "0003 files":

```
;
;Add 1 to an ASCII number. HL points to the
;units digit of the number to be incremented.
;
ADD1 MOV A,M ;get digit
     ORI '0' ;change possible ' ' to 'zero'
     INR A ;add 1 to it
     MOV M,A ;save it back
     CPI '9'+1 ;is it 1 more than '9'?
     RNZ ; no, just return
;
;got a carry
;
     MVI M,'0' ;set current digit to zero
     DCX H ;back up to next higher digit
     JMP ADD1 ;start all over
```

I think the ORI '0' is a "neat trick", to allow the leading spaces to be easily handled, without even explicitly looking for them. It works because the '0' is a 30H, or 0011 000 binary. Since a space is a 20H or 0010 0000, ORing in the 30H produces a 30H, or '0'. Similarly, if the number has been incremented, say, to 7, it is then 37H, or 0011 0111, and ORing it with 30H leaves it untouched.

ASCII TO BINARY CONVERSION: How would you convert a number from ASCII to binary? How does this sound?

Develop the answer by going a digit at a time, multiplying the previous result by 10 then adding in the next digit, repeating this until there are no more digits.

I qualify this subroutine by stating that it only handles positive numbers. A "-" sign will not be handled. I think with a little imagination, you could see how the start of the routine could test for a "-", and if there was one, remember to call the 16-bit negation routine discussed earlier.

The key part of this routine is the multiplying by ten. I mentioned this back in the "DAD TRICKS" part of the tutorial. It multiplied by 10 by simply using DAD D and DAD H. Here's the entire routine:

```
;
;ASCII to binary routine.
;(HL) points to ASCII input, terminated by any non-ASCII
;character. The delimiter character is returned in (A)
;if you want to check it. The binary answer is in HL.
;Registers BC, DE clobbered.
;
ASCIBIN XCHG ;move pointer to BC.
        LXI H,0 ;init answer
ASCBLP LDAX D ;Get input character
        INX D ;point to next
        CPI '0' ;compare to character zero,
        RC ; and return if less
        CPI '9'+1 ;compare to 1 higher than '9'
        RNC ; return if more or =
        SUI '0' ;turn into binary
        MOV B,H ;copy partial answer
        MOV C,L ; to DE for x by 10
        DAD H ;HL = 2 x original number
        DAD H ;HL = 4 x original number
        DAD B ;HL = 5 x original number
        DAD H ;HL =10 x original number
        ADD L ;add current digit
        MOV L,A ;put back answer
        JMP ASCBLP ;loop until done.
```

Since this is the longest subroutine covered so far, let me go into detail on every instruction to make sure everyone follows. If you understand it without further explanation, skip to "BINARY TO ASCII CONVERSION".

"ASCIBIN" is the name of the subroutine: ASCII to BINARY.

The routine makes use of the DAD instruction, which always adds to HL. Therefore, I decided early on, to make HL free, and thus the XCHG instruction. It swaps the contents of DE and HL. I don't really care about the fact that it swaps, but merely want to move HL to DE. I could have done:

```
PUSH H ;One way to copy DE to HL,
POP D ; but slow
```

and as a matter of fact, this is a very straightforward way to copy a register pair to another. I strongly dislike it for aesthetic purposes however, since it wastes time - stack instructions are always costly. A better way would be:


```
MOV D,H ;a better way to copy
MOV E,L ; DE to HL.
```

This takes two bytes of instructions, the same as the PUSH and POP, but since it uses no stack instructions, it executes much faster. However, if I didn't care about keeping the data in HL when moving it to DE, a simple:

```
XCHG
```

is the best solution, so that is what I used to place the pointer to the input data into DE.

The "answer" must be initialized. I have determined that the answer will be developed in HL, so decided to simply keep it there all along. I initialize it to zero by "LXI H,0", so that if no valid ASCII data is found, a zero is returned in HL.

"ASCBLP" is the label on the main processing loop of the subroutine.

"LDAX D" gets an ASCII character from memory, and "INX D" points to the next character.

The character must be between '0' and '9', to be a valid ASCII numeric digit. Using my old memory aid "C.A.L." (Carry if Accumulator is Lower), I test to see if the character loaded is less than a '0', by "CPI '0'". Since an ASCII '0' has a value of 30H, any character less than that, such as a carriage return (0DH) will cause carry to be set.

The "RC" returns if carry was set, as I have no more ASCII digits to process.

I get to the next instruction if the character was not less than '0'. I now want to test if it is between '0' and '9'. I *cannot* test it via "CPI '9'", because, from C.A.L. rule, carry will only be set if the answer is LESS THAN '9'. However, '9' itself is valid, so really, I want to know if less than or equal to '9'. I could:

```

CPI '9' ;is it less than '9'?
JC OK ; yes, it is OK
RNZ ;return if not = '9'
OK: .
.

```

Again, there is nothing wrong with programming like this - it is what you might directly translate from the thought "less than or equal to '9'". However, a little thinking will show you that if you do the CPI for the character which is one *more* than '9', then carry will be set for values *including* the '9'. You don't have to scurry to find out what character is one more than '9' (it is ':': just for the record). Instead, let the assembler do your work for you, by coding: "CPI '9'+1".

Since we have already returned for any character less than '0', a "RNC" will return if the character is not '0'-'9'.

At this point, the character is a digit from '0' to '9'. The first step in making the binary number is converting this single digit to binary, by a "SUI '0'" instruction. You might see such an instruction commented as "subtract ASCII offset" or "subtract ASCII bias", meaning that the ASCII number for 0, i.e. '0', is 30H, and the offset from "true" zero is thus 30H or '0'.

Now comes the multiply by 10. Since the 8080 instructions can only add, and not truly multiply, it takes the right combination of adds to produce a multiply by 10. It will be necessary to add the number to itself (DAD H), and also to add the original number, after doing some DAD H instructions. Thus, I copy the partial answer from HL to BC, via "MOV B,H" and "MOV C,L".

Then, the sequence "DAD H", "DAD H", "DAD B", "DAD H" converts HL into 10 x HL.

Finally, the new digit is added. Ideally, the digit should be added to L, but since you can only add to the accumulator (or do 16-bit adds to HL), I add L to A instead, via "ADD L", then put L back via "MOV L,A".

Finally, the subroutine JMPs back to the top of the loop via "JMP ASCBLP".

BINARY TO ASCII CONVERSION: Since converting from ASCII to BINARY involved multiplication (by 10), it seems logical that conversion from BINARY to ASCII would involve division. This is true. However, just as the multiplication was really just some adds, the division will be some subtracts.

To simplify things, the output will not be leading-zero suppressed.

Again I'll show the routine, then discuss it in detail. The technique is to essentially divide the number by 10000, then store the quotient as the number of 10000's in the original number, then divide by 1000, then 100, then 10, and finally store the remainder. This will thus develop a 5-digit ASCII number.

```

;
;ASCII to binary conversion. HL = binary number,
;(DE) points to output buffer. Number treated as unsigned.
;
ASCBIN LXI B,-10000 ;compute the ten-
CALL DECDIG ; thousands digit,
LXI B,-1000 ;then the
CALL DECDIG ; thousands,
LXI B,-100
CALL DECDIG ;hundreds,
LXI B,-10
CALL DECDIG ;tens,
MOV A,L
ORI '0'
STAX D ;store final digit
INX D ;bump pointer,
RET ; and return
;

```

This subroutine divides HL by BC, returning the quotient in A, expressed as an ASCII digit. Only useful when the quotient is between zero and 9, as it would be in a binary to ASCII conversion routine.

```

;
DECDIG MVI A,'0'-1 ;init (see details below)
DECLP INR A ;add 1 to quotient
DAD B ;"subtract"

```

(continued next page)


```

JC     DECLP    ;loop if it fit
STAX   D        ;store output ASCII
INX    D        ;point to next char position
MOV    A,B      ;compute the
CMA    ;        negative
MOV    B,A      ;        of
MOV    A,C      ;        the
CMA    ;        BC
MOV    C,A      ;        register
INX    B        ;        pair
DAD    B        ;un-do last subtract
RET    ;        and return

```

Here's a detailed explanation:

"ASCBIN" is the name of the subroutine. The first step is to compute how many 10,000's there are in the number and store that digit. "LXI B, -10000" and "CALL DECDIG" do this. See the details on DECDIG, below.

Repeat the above step for 1000, 100, then 10. Upon return from DECDIG at this time, the final digit, from zero to 9, is left in HL. Thus, "MOV A,L" gets the digit (in binary), "ORI '0'" changes the binary to ASCII, "STAX D" stores it. The "INX D" is not really necessary, but in case the routine calling it wants to put something more into the buffer, it skips the last character stored.

The "DECDIG" subroutine divides HL by BC. Normally, this would be done by subtraction. It could be by a subroutine similar to DSUB show earlier, but since the number being subtracted is a constant, it is easier to "add a negative number" than to subtract a positive number.

Step one is to initialize the quotient. I could just initialize it to zero, then add one every time a subtract was successful. However, this would require some extra JMPS, such as:

```

MVI    A,0      ;init quotient
DECLP  DAD      B        ;"subtract"
JNC    NOMORE
INR    A        ;count quotient
JMP    DECLP
NOMORE: .
.

```

It would be much easier to put the INR A at the top of the loop, but then it would count one too many times. The simple solution is to initialize it to 0FFH, i.e. one less than zero:

```

MVI    A,0FFH  ;init quotient
DECLP  INR      A        ;count quotient
DAD    B        ;"subtract"
JC     DECLP    ;loop if it subtracted
.
.

```

This would then be followed by an "ADI '0'", to make the binary value into an ASCII digit. However, it doesn't really matter where the '0' is added, so why not put it in the accumulator as part of the initialization. Thus, "MVI A,'0' - 1" initializes the accumulator to the '0', but minus one, because of the INR at the top of the loop. Whew!

"DECLP" is the looping label for the subtract. "INR A" counts one quotient.

"DAD B" is the "subtract" itself. If the subtract was able to be made, it produces a carry. This is the opposite of addition, in which carry would indicate the sum was too large. Thus, "JC DECLP" loops as long as the subtract could "fit".

Then, "STAX D" stores the ASCII digit, and "INX D" bumps the pointer to be ready for the next digit.

Now the first problem. The most reasonable way to detect that you couldn't subtract any more, was finding carry no longer set. Trouble is, one of the negative divisors has been added. Since there is no double subtract, it is necessary to do a 16-bit subtract by complementing and adding.

To do the complement, it is the now-familiar "MOV A,B", "CMA", "MOV B,A", "MOV A,C", "CMA", "MOV C,A", and "INX B". That generates the 16-bit complement, then "DAD B" to un-do the last DAD, then "RET".

COMING UP: Next month, I'll present a continuation of the subroutines, showing a MOVE routine which will work fast on either 8080 or Z 80, because it detects which processor it is running under. I'll also get into Input/Output. Specifically, CP/M-80 character I/O.

Future installments will get into CP/M-80 disk I/O, both simple and buffered.

If you have any questions about how to accomplish a task in 8080 assembler, write to me C/O *Lifelines/The Software Magazine*, 1651 Third Ave., New York, N.Y. 10028. Please restrict your questions to those of a limited enough scope to be handled in a subroutine. Examples might be: "How do I convert from EBCDIC (the code used on virtually all IBM machines except the PC) to ASCII?". I will publish the question and an answer in future installments.

A Call For Manuscripts

Maybe you've written for publication before? Or maybe you've always wanted to write? It could be that reading *Lifelines/The Software Magazine* has given you some ideas on what you have to contribute. We're interested in hearing what you have learned, and so are other readers. If you've got experience using software that runs with CP/M-80, UNIX, CP/M-86, MS-DOS, XENIX, or UCSD Pascal we'd like to talk to you. We like to publish both longer essays and those short gems which can hold so much important information. We pay competitively and our current authors will tell you that writing for a magazine like ours is satisfying in many ways.

Send us a brief resume of your software experience, and samples of your previous writing, if you have any. (Don't be shy if you're **not** an experienced writer.) Then we can talk about your work and about payment for your efforts. Write or call: Editorial Dept., Lifelines Publishing Corp., 1651 Third Ave., New York, N.Y. 10028. Telephone: (212) 722-1700.

MicroSpell, MicroProof, And SpellGuard

James K. Mills

Before the widespread availability of microcomputers the word processing market was primarily the domain of minicomputers manufactured by such companies as IBM, Wang, and others. Now, although the minicomputers still hold their place in the market, there are a great many microcomputers dedicated, many on a part-time basis, to word processing. For those of us who are not the best of typists, there are a variety of word processing systems available, including a relatively recent development: proofreading programs to check your spelling.

The readers will recall the article by Robert Van Natta reviewing SpellStar (*Lifelines/The Software Magazine*, April 1982). I would like to compare SpellStar to MICROPROOF, MicroSpell, and SpellGuard. SpellStar has been reviewed by Mr. Van Natta, so I will concentrate on the other three programs.

SpellGuard

SpellGuard, by ISA (Innovative Software Applications) of Menlo Park, California, comes with a softback book of instructions as program documentation. This book gives the user a step-by-step introduction to SpellGuard's use, the help commands, setting up the normal default entries, etc. Appendices include usage tips, technical information, error messages and a glossary of terms. The documentation, in my opinion, is very well done, with many illustrations to guide the new user.

SpellGuard comes on a single diskette containing two .COM files: SP.COM (6K) and MAINTAIN.COM (13K) for spelling checks and dictionary maintenance, respectively. In addition, there is a 33K dictionary file, SP.DIC, and a 54K messages file SP.ISA. The only other file on the disk is LETTER.TXT (3K), a sample text letter for testing SpellGuard.

Running SpellGuard is simplicity itself. The program is self-prompting and

guides the user by the hand through its entire operation. This makes it ideal for the "non-computerist" user, like the office secretary. The user is allowed to have multiple dictionaries for different purposes, such as correspondence, lab reports, etc. However, only one dictionary may be used at any one time. SpellGuard does *not* allow supplemental dictionaries in addition to the "regular" dictionary; this may or may not be a drawback, depending in your particular application. SpellGuard also lets you change its "default table" by using different files on disk for the default table; the table specifies the dictionary to use, among other things.

SpellGuard lets you maintain your dictionary using MAINTAIN.COM. SpellGuard's dictionary, as supplied, contains about 20,000 words. SP.COM allows you to add words to the dictionary as you do your proofreading. MAINTAIN permits you to copy dictionaries, and merge (or add), and subtract dictionaries from one another.

Overall, I'd say SpellGuard is well-suited for use by non-computerists, especially office secretarial personnel.

MicroProof

Microproof, by Cornucopia Software of Walnut Creek, California, will run only on a system equipped with a Z80 CPU. This may rule out MicroProof for some users.

MicroProof's documentation comes in a softbound notebook binder and gives the usual introduction, including a brief history of the development of MicroProof by Mr. Phil Manfield; then the manual gets into actual usage. The documentation is not quite as effective as SpellGuard's, but it is by no means difficult to follow. I had no problem whatsoever in understanding how to utilize MicroProof.

MicroProof can be configured to correct the errors after it proofs your document,

or it can simply list the misspelled words to CRT or printer - then you search them out with your word processor. In the first case, you are prompted with the dubious word and given a choice of several responses: correct the misspelling, ignore the error, display the context, add the word to the dictionary, or quit the program. If you choose to add the word to the dictionary, you may also "code" the word as a verb, noun, adverb, or adjective. The manual also tells you not to enter a plural as an entry, but only the root word. Presumably, MicroProof will search for plurals of root words.

MicroProof is distributed on a single eight-inch diskette, containing several files: MICPROOF.COM (9K) is the proofreading program; CORRECT1.COM (1K) and CORRECT2.COM (2K) are called automatically by the "correcting" version of MicroProof and used for correcting and adding to the dictionary; ADDTODIC.COM (6K) is used to add a list of words to the dictionary; PRINTDIC.COM (3K) is used to print and edit the dictionary, and TEST.COM does some magic to assure you that you have received a valid disk. The dictionary files are DICT1.DAT (34K) and DICT2.DAT (35K) which contain the 50,000 word dictionary. In addition, there is a DICT3.DAT (1K) which is used for storing words added to the dictionary. Finally, there is a sample text file, EXAMPLE (2K). The manual makes reference to some other files that may be on certain diskettes, such as M.COM, an intermediate program to link MicroProof with word processors other than WordStar. There may also be a patch program for similar purposes. One thing that did impress me about MicroProof's manual is that numerous instructions are given for customizing MicroProof to your disk system, whether it be five-inch or eight-inch, or TRS80, etc.

Like most such programs, MicroProof prompts for the input it wants and guides you through its operation. It is

(continued next page)

not as wordy as SpellGuard, but it does nicely.

Overall, I'd say that MicroProof seems to be a well-conceived system, and the user should have little trouble with it.

MicroSpell

MicroSpell, by Bob Lucas (distributed by Lifeboat Associates), comes with its documentation in a three-ring binder; it is 37 pages in length as opposed to MicroProof's 30 pages and SpellGuard's 116 pages (116 is *not* a typo). MicroSpell has more options and variables than the other two programs, which makes it both more versatile and more difficult to learn to use. While the manual isn't as wordy as SpellGuard's, it does cover the various aspects of using MicroSpell concisely, and should be read, and even re-read from time to time, so the user can become familiar with all the options available.

MicroSpell is distributed on two eight-inch diskettes. The first disk contains most of the programs and files used with MicroSpell:

BUILD .COM (13K) used to build LEX files
CUSTOM .COM (13K) customize to your
syst.
EMPTY .COM (2K) create an empty file
INVERT .COM (9K) dumps the LEX files
SPELL .COM (16K) proofing program
UNBUILD .COM (11K) removes words from
LEX files

LEX.1 (34K) dictionary file
(letters A-D)
LEX.2 (32K) dictionary file
(letters E-L)
LEX.3 (33K) dictionary file
(letters M-R)
LEX.4 (32K) dictionary file
(letters S-Z)

In addition to the above, there are some help files, and a demo file. On the second distribution diskette are the following vocabulary files:

ED .VOC (29K)
EXTRA.VOC (35K)
ING .VOC (20K)
RARE .VOC (41K)

By using BUILD and UNBUILD, the user can add to or delete from the dictionary the lists of words contained in the four vocabulary files. The ED,

ING, and RARE files are additional words to be added to the dictionary if you have enough memory to support the bigger dictionary segments. The EXTRA file is a list of words you can delete from the dictionary if you are limited by memory size and cannot run the SPELL program as provided.

MicroSpell does the proofreading in four passes, one pass for each dictionary file. Options upon invoking MicroSpell allow you to skip any or all of the first three passes, suppress creation of the "exception" (bad word) file, suppress context display for bad words, accept all uppercase words, suppress creating the output file (dry run), send words to the exception file, suppress the suffix guessing routines, suppress creation of backup file, accept "RUN-OFF" type mnemonics, mark unfamiliar words in the output, and more. MicroSpell also lets you suppress printing the plurals that MicroSpell accepts, but it is fun to watch the way MicroSpell's algorithms work on plurals. And you will find that occasionally the words that MicroSpell accepts as plurals and suffixes are not correct - the English language not always being logical.

In addition to all of this, there is a learn mode (very helpful in view of the complexity) to help you become familiar with MicroSpell. Like most of the programs reviewed here, MicroSpell also has an appendix with a table of commands and options (also called INFO.HLP on disk).

Whew! That's a mouthful, but maybe it will give you an idea of the flexibility of MicroSpell as compared to other programs.

When actually doing the proofreading, errors are reported to the user, who is then allowed to correct the error, add new words to the dictionary, display the context (automatic unless suppressed by user), quit, and probe the dictionary for more words (i.e., look for CR??TE, where the ?? are wildcards). But here's MicroSpell's *big* feature: MicroSpell "guesses" what it thinks the misspelled word should be by looking (automatically or by command) for similar words in its dictionary. Typically, MicroSpell will come up with one to four guesses for misspelled words. Each guess is numbered, and all you have to do is select that number to replace the word in the file with that

guess. This is actually quite helpful, and makes using MicroSpell quite entertaining. Watching it make guesses for your bad words and watching it find root words that match words with (sometimes) complex suffixes is quite interesting. And even though you may know the correct spelling of a misspelled word, you're saved the trouble of keying it in.

Opinions And Conclusions

Well, it will probably come as no surprise to the reader that I, personally, like MicroSpell better than the others. It is like comparing a text editor to a word processor - there are similarities, but you can do so much more with the more complex program that gives you more options. (I once swore foolishly that I'd never use WordStar, I was happy with my copy of WordMaster, but now I write these articles with WordStar!) The only item left open and uncovered is speed of execution. No one likes to wait all day for a program to finish. The table provided should give you all the data you need. Note that MicroProof is not specified - I am currently using an 8080 CPU, not the Z80 required for MicroProof. However, I am in the process of upgrading, and I plan to publish a followup article next month to indicate how all these programs do on a 4 MHz Z80. The system used for these comparisons (this month) is an 8080 running at 2 MHz with 64k of memory, and a Tarbell single density floppy disk controller. The file checked was this article (prior to any corrections).

Final Conclusions

The reader will have the best idea of what is best for his or her purposes. SpellGuard is well-suited for an office environment where speed is the primary factor, although you may encounter objections from personnel who have to "bypass" all the false errors (this is true of all the proofreading programs). One of the things that Mr. Van Natta pointed out in a previous article is that the secretaries don't want to spend the time and effort to run the spelling check program and do the necessary subsequent editing, whether part of the proofing program or a separate editor. It does take time and effort.

If you are more concerned with amount of output as opposed to quality of output, so be it. If you are writing for pub-

lication, or for any purpose for which you want the neatest and most professional appearance, you will want to

take the time and make the effort to use a proofreading program, whichever variety is best for your needs.

CRITERIA	SPELLSTAR	MICROSPELL	SPELLGUARD	MICROPROOF
time to proof:	3:30	n/a	00:45	next month
time to correct:	11:30	n/a	10:00	next month
total time:	15:00	8:30	10:45	next month
# words in dict.:	21026	?	20000	50000
# words in file:	1704	1704	1704	1704
# different words:	587	587	587	587
# words misspelled:	79	53	86	next month
total misspelled:	168	98	?	next month
actual misspelled:	12	12	10	next month
false errors:	156	86	81	next month
Memory required:	?	48K minimum	32K minimum	next month
Processor required:	any *	any *	any *	Z80 only

* "any" means 8080, 8085, or Z80.

? MicroSpell does not tell you how big its dictionary is.

SpellGuard does not inform you of the total number of mismatched words, only the number of unique mismatched words.

SpellStar must run under WordStar, which will not run on a 32K system. I'm not sure of the minimum memory required.

Software Notes

Patches For MAGSAM

Micro Applications Group has released two patches to correct bugs in its products. The first problem occurs in all versions of MAGSAM/E and PRISM/ADS. When MAGSAM is run unbuffered it may during a key search look at the buffer contents as overflow, when in fact the buffer contains index data. As result, a pointer may point to itself, causing a loop from which MAGSAM never returns. This only occurs during key searches (RK, SK, RG, SG, WA, SA, KD, SD, DR).

If the loop has occurred and the program has been aborted, the index structure may be corrupted. It should be rebuilt by sequentially reading the data file and with each record executing a WA into new index and overflow files.

To cure this problem, make the following changes to MAGSAM.BAS. Then recompile all programs using MAGSAM.BAS. (New code is underscored.)

```

61220 MAGSAM%(0)=VAL(MID$(MAGSAM$(12),MAGSAM%(14)+MAGSAM%(4),6))
      MAGSAM%(7)=MAGSAM%(7)+1:RETURN
61230 IF MAGSAM$(6)<>"B" THEN GOTO 61237
      IF MAGSAM.BIO$="O" AND MAGSAM.BUF%=MAGSAM%(0) THEN \
        GOTO 61239
      MAGSAM.BUF%=MAGSAM%(0):MAGSAM.BIO$="O"
61237 READ#MAGSAM%(3),MAGSAM%(0);MAGSAM$(12)
61239 RETURN
  
```

The second bug occurs in all versions of MAGSAM/E, in versions 4.2 and higher of MAGSAM III, and in versions 1.1 and higher of MAGSAM IV. During a reorganize involving duplicate keys, MAGSAMRO may fail to write the last overflow record. This may result in a CBASIC error EF (End of File) on subsequent access of the index structure, and the overflow file may be missing the last record after the first reorganize. The index structure, which may be corrupted, should be rebuilt in the fashion described above.

Apply the changes below to MAGSAMRO.BAS and recompile the programs which use it, including MAGSAMRX. New code is underlined.

```

64700 IF LEN (MAGSAM$(16))>1 THEN GOSUB 64800:MAGSAM%(16)=MAGSAM%(16)+1
      MAGSAM%(11)=MAGSAM%(16)-1
      IF LEN(MAGSAM$(11))>1 AND LEFT$(MAGSAM$(11),1)="O" THEN GOSUB 65400
      IF MAGSAM%(18)<>0 THEN GOSUB 64750
  
```


A Detailed Description Of PLAN80, Part 2

Raymond J. Sonoff

PLAN80 was initially discussed by the author in the May 1982 issue of *Lifelines/The Software Magazine*. This second installment review article discusses how PLAN80 might be used.

Certainly, most PLAN80 applications relate to financial modeling. After all, depreciation and internal rate of return functions are built into PLAN80's library of functions. An example of a Procedure file that illustrates several types of depreciation over varying periods of time, and its associated printout, are shown in Tables I and II.

By virtually eliminating the drudgery previously associated with calculating, recalculating, checking and ultimately producing high quality reports, you should find that PLAN80 will enable you to save time, perform much needed sensitivity analyses, and provide (via simple PRINT commands) one or more finished reports incorporating those parameters you have selected as appropriate for your particular multivariate models, sets of equations, etc.

In short, not only will such computing power provide you with the capability to solve problems readily and accurately while certainly enhancing your own understanding, but any of the mass storage files associated with your efforts can, if so desired, be retrieved, reformatted, or subjected to further "What if...?" considerations.

Features Of PLAN80

PLAN80 statements are divided into sections (TITLES, COLUMNS, ROWS, DATA, AND RULES). Each statement is described below.

TITLES. The TITLES section introduces up to nine lines at the top of the report. Each line may be up to 60 characters wide and be centered, left-justified, or right-justified.

COLUMNS. There must be at least one COLUMN.

ROWS. ROW specifications define the structure and certain printing characteristics associated with a report's ROWS.

DATA. DATA may be entered into any row or column, and the range of rows applying to entry by column (or the range of columns applying to entry by row) is under user control. Values can be placed into any part of a PLAN80 application.

INTERACTIVE. The INTERACTIVE statement indicates where PLAN80 should begin to recalculate after values are changed in the DISPLAY mode.

RULES. The RULES statements indicate calculations.

OPTIONS. The OPTIONS section lets you specify characteristics of a printed report.

DISPLAY. The DISPLAY statement allows you to view results on your computer screen or to print the report.

Some of the more positive features of PLAN80 that have been noted to date are these:

1-PLAN80 is forgiving. It auto-initializes data fields to zero upon startup; it "fits" files compatibly when GET statements are used; when run-time errors in the RULES section of the Procedure file are skipped via the <C> "continue" key, the associated data values will be set to zero.

2-PLAN80's FOR, GET, PUT, INCLUDE, PRINT, ... STATEMENTS allow for direct selection, limiting, combining, or other appropriate RULES, DATA, and DISPLAY sections of a given Procedure file being

operated on by the user. These STATEMENTS prove easy to implement.

- 3-The strict adherence to syntax, and the 36 error codes provided greatly aid in Procedure file program debugging.
- 4-Files can be readily called from or stored on user-specified disks, or output can be to a printer or the console.
- 5-The INTERACTIVE STATEMENT provides the user with the capability to perform multiple computations for various data entries or combinations of data values. A simple recalculate command is all that is required after making these changes while in the Data Display Mode of PLAN80.

Where PLAN80 could be improved, at least from the user's present experience with this software package, is as follows:

- 1-A built-in "mini-editor" would save considerable time exiting from PLAN80, calling up an editor and the procedure file to be edited, making the changes thought appropriate, storing the edited file, calling up PLAN80, and inputting the edited procedure file for execution to see if no further syntax errors have been made, etc.
- 2-Many more examples and greater elaboration of how the various STATEMENTS can be utilized effectively and efficiently in Procedure files would be helpful.
- 3-The establishment of a PLAN80 USERS GROUP with a periodic mailing program would be a great idea. Numerous PLAN80 software users could contribute to its contents.

4-A printable DUMP of encountered error codes with their locations to aid in debugging of Procedure files could be offered.

Final Comments

PLAN80 has proven more than a little difficult for me to get used to. Essentially, you set up each procedure file as indicated in the FEATURES section of this article. However, the fact that you compose your file using algebraic relationships under the RULES portion of the overall file, introduce DATA values in ROW and COLUMN positions, and ultimately run a program to DISPLAY the results of your file creation activity does differ considerably from normal word processor-oriented operations (including to some extent T/MAKER II which I reviewed for *Lifelines/The Software Magazine* also). Now, after having spent the time actually doing a number of these operations, I am beginning to appreciate and to feel comfortable with PLAN80.

So, if your experience is anything like mine, be prepared to spend time getting used to the PLAN80 "system". It is worth it!

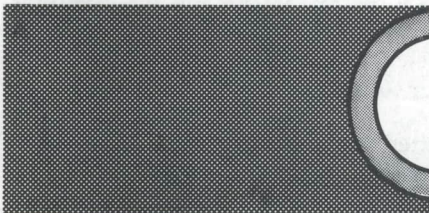


Table I

```

:TITLES
  1 "PLAN80 EXAMPLE #1"
  2 "Multi-Year Depreciation Options Modeling"
  3 "(Cost and Depreciation, in Thousands of Dollars)"
:COLUMNS
  Y1982 "1982"
  Y1983 "1983"
  Y1984 "1984"
  Y1985 "1985"
  Y1986 "1986"
  Y1987 "1987"
  Y1988 "1988"
  Y1989 "1989"
  Y1990 "1990"
  Y1991 "1991"
:ROWS
  MACHINE1 "Machine #1 (5-yrs.)"
  MACHINE2 "Machine #2 (7-yrs.)"
  MACHINE3 "Machine #3 (7-yrs.)"
  MACHINE4 "Machine #4 (6-yrs.)"
  MACHINE5 "Machine #5 (10-yrs.)"
  SL1 "Straight Line #1"
  SOD2 "Sum-of-Digits #2"
  DB3 "Declin'g Bal. #3"
  DB4 "Dbl.Decl. Bal.#4"
  SOD5 "Sum-of-Digits #5"
:DATA
  MACHINE1 = 100
  MACHINE2 = 200
  MACHINE3 = * 200
  MACHINE4 = * * 400
  MACHINES5 = * * * 800
:INTERACTIVE
:RULES
  SL1 = @SL(MACHINE1,5,0.5)
  SOD2 = @SOD(MACHINE2,7)
  DB3 = @DB(MACHINE3,7,1.25,1)
  DB4 = @DB(MACHINE4,6,2.0,1)
  SOD5 = @SOD(MACHINE5,10)
:OPTIONS
  ROWWID(20)
:DISPLAY

```

Table II
Multi-Year Depreciation Options Modeling
(Cost and Depreciation, in Thousands of Dollars)

	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991
Machine #1 (5-yrs.)	100	-	-	-	-	-	-	-	-	-
Machine #2 (7-yrs.)	200	-	-	-	-	-	-	-	-	-
Machine #3 (7-yrs.)	-	200	-	-	-	-	-	-	-	-
Machine #4 (6-yrs.)	-	-	400	-	-	-	-	-	-	-
Machine #5 (10-yrs.)	-	-	-	800	-	-	-	-	-	-
Straight Line #1	10	30	50	70	90	100	100	100	100	100
Sum-of-Digits #2	50	93	129	157	179	193	200	200	200	200
Declin'g Bal. #3	-	36	65	89	109	125	139	150	159	166
Dbl. Decl. Bal. #4	-	-	133	222	281	321	347	365	377	384
Sum-of-Digits #5	-	-	-	145	276	393	495	582	655	713

Software Notes

Pseudo-Relocatable Subroutines, Part 2

Gregory A. Knott

Last month I presented a method of creating 'pseudo-relocatable' assembler language code using Digital Research's ASM. If you recall, this feat was accomplished using a 'relocator' constant. What this constant actually did was cause ASM to generate reference addresses to variables and labels not according to Hoyle (or Kildall if you prefer). The addresses generated were actually displaced to another location in memory. This allowed a program to be created that could be relocated to a pre-determined location other than the Transient Program Area (TPA). This relocatability was also accomplished using the distributed ASM and LOAD utilities, thus negating the need for purchasing a relocatable assembler.

This technique is most suitable to subroutines that will be called from some main program, like MBASIC. These subroutines typically have to be loaded in memory prior to MBASIC being initiated. In Part 1, I demonstrated how to load this subroutine into memory using DDT. Now, DDT is kind of fun to play around with, but I don't get my jollies jumping in and out of DDT just to load a subroutine that one of my productive programs needs. I knew there must be a better way!

The Dream

I thought it would be great to run one program that would place my subroutine into memory at the proper location and properly transfer control to MBASIC. It would be even better if MBASIC could start right off on its productive program. This would mean that to start any application, the user would simply have to enter one command - and it's off to the races. The user wouldn't have to know anything about DDT or even MBASIC or be concerned in the slightest that there was a slick little subroutine floating around in memory somewhere. What I would have would be "user friendly" (my definition: someone who has used your software extensively for one year and still calls you a friend)!

The Loader

The program presented here will do just that. It is written in such a manner that it uses the relocator constant and thus will allow a better understanding of how it is employed. The program is attached in Figure 1. Let's look at it.

The very first two program statements are EQUates that define two absolute memory addresses. The first, SUBADDR, is the location where the subroutine will be loaded. This must match the 'Pseudo-Org' location of the compiled subroutine. The subroutine should be placed in the highest possible memory location so that MBASIC can use as much memory as possible. It should, however, be placed below the FDOS. To figure out the highest location you can use, look at locations 0006H and 0007H. CP/M-80 uses this address to tell programs where not to go. You best not go there either. (If you're using DDT to read that location you will find that DDT will adjust that location downward as DDT actually sits just below the FDOS).

The second, LOADADDR, is the location where a piece of this program will be moved. It is necessary to move this portion of the program out of the way because later MBASIC comes flying right into this area of memory and would overlay the actual loader routines and cause all kinds of havoc. Make sure this address is beneath SUBADDR by at least 200 bytes or the subroutine will overlay the moved portion of the program.

The first part of the program, after getting control from CP/M-80, moves the loader portion to the location LOADADDR. Once the major part of the program has been relocated up into high memory, control is transferred to this portion.

At this point, the stack is set up and the subroutine is loaded into memory by the statement CALL GETPROG+Z. Note here the use of the relocator

constant. Since this part of the program is no longer relative to the TPA address, it was necessary to fake out ASM and have it generate an address for this label that we knew it could find. If you can understand why the +Z was added you'll have no trouble with this relocatability nonsense.

After our subroutine is loaded, we then CALL GETPROG+Z again, but this time we are looking for the main program MBASIC. Note that the subroutine was loaded in high memory but MBASIC was sent to the TPA. Also note that it is not only necessary to refer to labels of statements with the relocator constant but also for labels of variables such as SUBFCB. However, we don't add the relocator constant to the absolute addresses that aren't relocated like TPA and SUBADDR.

Once MBASIC is loaded we set up the command tail line which it reads from CP/M-80's default buffer area. In this case the parameter information we want to pass is to execute the program PRINTEST.BAS and not allow any memory usage above B100H (where our subroutine is now residing).

After this is accomplished we simply transfer control to location 100H, the TPA (where MBASIC is now).

If there is some problem with opening a file it is assumed to be absent from the disk and an error message is printed.

In actual use the program is executed like this:

```
A>LOADSUB
```

It will also require the following files to be on drive A:

```
MBASIC.COM  
PRINTHI.COM  
PRINTEST.BAS
```

Some Caveats

When writing pseudo-relocatable subroutines you must be very careful to

The largest selection of software from the world's largest software publisher.

LIFEBOAT'S PRODUCT LIST NO. 22 1/2

NEW — 16-Bit Software Available

for the IBM PC, plus . . .

System Tools:

Emulator/86 (the CP/M emulator)
EM80/86
PMATE-86
UT86

Telecommunications:

ASCOM

Languages:

Lattice C Compiler

DataBase Management Systems:

T.I.M. III

Disk Operating Systems:

MS-DOS — soon available configured for CompuPro Sweet 17 and Software Development System. Currently available for OEM license.

Media & Formats

IBM Personal ComputerG1
GodBout.....E1
SeattleE1
TecMarE1

8-Bit Software Available

System Tools:

BUG and uBUG
DESPOOL
DISILOG
DISTEL
EDIT
EDIT-80
FILETRAN
IBM/CPM
MAC
MACRO-80
MINCE
PANEL
PASM
PLINK
PLINK II
PMATE
RAID
Reclaim
SID
TRS-80 Model II Cust. Disk
Unlock
WordMaster
XASM: 05, 09, 18, 48, 51, 65, 68, F8, 400
ZAP80
ZDT
Z80 Development Package
ZSID

Precision BASIC
BD Software C Compiler
CBASIC-2
CIS COBOL (Standard)
CIS COBOL (Compact)
COBOL-80
FORTRAN-80
KBASIC
muLISP/muSTAR-80
Nevada COBOL
JRT Pascal
Pascal/M
Pascal/MT
Pascal/M +
Pascal/Z
PL/I-80
STIFF UPPER LISP
S-BASIC
Timin's Forth
Tiny-C
Tiny-C Two
UCSD Pascal
Whitesmith's C Compiler
XYBASIC

Language and Applications Tools:

BASIC Utility Disk
DataStar
FABS
FABS II
Forms 1 for CIS COBOL
Forms 2 for CIS COBOL
MAGSAM III
MAGSAM IV
MAGSORT
MSORT for COBOL 80
Programmer's Apprentice
PSORT
QSORT
STRING/80
STRING BIT
SUPERSORT
ULTRASORT II
VISAM

Word Processing Systems and Aids:

Benchmark
DocuMate/Plus
MicroSpell
Letteright
Magic Wand
Spellguard
TEX
Textwriter III
WordIndex
WordStar
WordStar Customization Notes

Data Management Systems:

CÓNDOR
Formula
HDBS
Hoe
Microseed
MDBS
MDBS.DRS, QRS, RTL
dBASE II
PRISM/LMS
PRISM/IMS
PRISM/ADS
T.I.M. III

General Purpose Applications:

CBS
CBS Label Option
Selector III-C2
Selector IV

Mailing List Systems:

Benchmark Mailing List
Postmaster
Mailing Address
MailMerge for WordStar
NAD

Financial Accounting Packages:

BOSS Financial Accounting System

Peachtree Financial Packages
Univair 9000 Series
General Ledger Accounting
Structured Systems Group Financial Packages
GLector

Numerical Problem-Solving Tools:

T/MAKER II
fpl
PLAN80
Analyst
Microstat
muSIMP/muMATH
Statpak

Professional And Office Aids:

Angel
American Software Property Management Package
Cornwall Apartment Management
Datebook
GrafTalk
Guardian
Professional Time Accounting
Property Management
PAS 3 Medical
PAS 3 DENTAL
Sales Pro
Torricelli Author
Univair 9000 Series Family Medical Management
Univair 9000 Series Family Dental Management
Univair 9000 Series Insurance Agency Management
Univair 8000 Medical Management
Univair 8000 Dental Management
Wiremaster
Univair 9000 Series
Legal Time Accounting

Books, Periodicals, Accessories

APL—An Interactive Approach
Accounts Payable and Accounts Receivable-CBASIC
The CP/M Handbook (with MP/M)
The C Programming Language
8080/Z80 Assembly Language Techniques For Improved Programming
Executive Computing
Fifty BASIC Exercises
General Ledger-CBASIC
H.W.Sams Crash Course in Microcomputing
Introduction to Pascal
Lifelines
Pascal User Manual and Report
The Pascal Handbook
The Pascal Primer
Payroll with Cost Accounting —CBASIC
Structured Microprocessor Programming
Using CP/M—A Self-Teaching Guide
Smartmodem
DC Data Cartridges
Flippy Disk Kit
Floppy Saver
Diskette Drive Head Cleaning Kits
Vari Clean Cleaning Kit

Disk Operating Systems

Software Bus Family
SB-80
CP/M-80
MP/M

Hard Disk Integration Modules

Telecommunications:

ASCOM
BSTAM
BSTMS
MicroLink-80
RBTE-80

Languages:

ALGOL-80
APL/V80
BASIC Compiler
BASIC-80
baZic II

Media & Formats

This list of available formats is subject to change without notice. If you do not see your computer listed or are uncertain, call to confirm the format code for any particular equipment.

ADDS MultivisionRT
ALSPA 8 in.A1
Altair 8800B1
AltosA1
Apple CP/M 13 SectorRG
Apple CP/M 16 SectorRR
Archives 1SG
AVL Eagle IIST
BASF System 7100RD
Blackhawk Micropolis Mod IIQ2
BMC IF-800SR
California Computer Sys 8 in.A1
CDS Versatile 3BQ1
CDS Versatile 4Q2
Columbia Data Products 5.25 in.A1
Commodore CBM/PET w/SSES4
Box + 8050C2
Commodore CBM/PET
w/Madison Z-RAM + 8050C4
COMPAL-80Q2
Computer Ops N.C. HQS2
Control Data 110A1
CPT 8000A1
Cromemco System 3A1
Cromemco System 2 SD/SSR6

Cromemco System 2 DD/SSRX
Cromemco System 2 DD/DSRY
CSSN BackupT1
Datapoint 1550/2150A1
DEC VT 18XSD
Delta SystemsA1
Digi-Log Microterm IIRD
Direct OA1000M2
DTC Micro 210ASC
Durango F-85RL
Dynabyte DB8/2R1
Dynabyte DB8/4A1
Exidy Sorcerer +
Lifeboat CP/M-80Q2
Exidy Sorcerer +
Exidy CP/M-80 5.25 in.RW
Exidy Sorcerer +
Exidy CP/M-80 8 in.A1
EXOA1
FindexP6
Heath H8 + H47A1
Heath H89 + Magnolia CP/M-80P7
Heath H89 + Heath CP/M-80P7
Helios IIB2
Heurikon MLZ, SSSN
Heurikon MLZ, DSSO
Hewlett-Packard 125, 5.25 in.SB
Hewlett-Packard 125, 8 in.A1
IBEX 7100RQ
ICOM 2411 Micro FloppyR3
ICOM 3712A1
ICOM 3812A1
ICOM 3812A1
IMSAI VDP-40/VDP-42R4

IMSAI VDP-44R5
IMSAI VDP-80A1
Industrial Microsystems 5000RA
Industrial Microsystems 8000A1
Intel MDSS SDR1
Intertec Superbrain DOS 0.5-2.xRJ
Intertec Superbrain DOS 3.xRK
Intertec Superbrain QDRS
ISC Intercolor 8063/8360/8963A1
Lexitron VT 1303 DS/DDS8
Lexus Alphasprint Model S1S1
Meca Delta-1 5.25 in.P6
MICOM 2001B3
MICOM 2001EB4
MICOM 3003M1
MicromationA1
MicroMega 85SC
Micropolis Mod 1Q1
Micropolis Mod IIQ2
MITS 3200-3202B1
Monroe OC 8820, DD/SSSW
Morrow DiscusA1
MostekA1
MSD 5.25 in.RC
MULTI-TECH-IQ2
MULTI-TECH-IIQ2
Nascom (Gemini drives)R3
Nascom II with Lucas DrivesSL
NCR 8140/9010A1
NEC PC-8001RV
Nicolet Logic Analyzer Model 764SX
NNC-80/80WA1
North Star SDP1

North Star DDP2
North Star QDP3
Northern Telecom 503SM
Nylac Micropolis Mod IIQ2
Ohio Scientific C3A3
OKI IF-800 + MSA CP/M-80SP
OKI IF-800 + OKI/Lifeboat
CP/M-80SR
(Above OKI entries replace catalog
entry for OKI IF-800 format code RZ)
Percoc PCC 2000A1
PET/CBM w/Small Systems
Engineering Box + 8050C2
PET/CBM w/Madison Z-RAM +
8050C4
Philips MICOM 2001 8 in.B3
Philips MICOM 2001EB4
Philips MICOM 3003M1
Processor Technology Helios IIB2
Quay 500RQ
Quay 520RP
RAIR DDRE
Research Machines 5.25 in.RH
Research Machines 8 in.A1
Sanco 7000 5.25 in.RQ
Sanyo MBC 1000SY
Sanyo MBC 2000SS
Sanyo MBC 3000A1
SD Systems 5.25 in.R3
SD Systems 8 in.A1
SpacebyteA1
Tarbell 8 in.A1
TEI 5.25 in.R3

TEI 8 in.A1
Televideo DD/DSS5
T.I.P. (Alloy Engineering, Inc.)T3
Toshiba T200SF
Triumph Adler AlphatronSV
TRS-80 Model 1 +
Shuffleboard 8 in.A1
TRS-80 Model IIA1
Vector MZQ2
Vector System 2800A1
Vector System B/VIPQ2
Vista V-80 5.25 in. SDR8
Vista V200 5.25 in. DDP6
WangwriterSE
WORDPLEXSZ
XEROX 820, 5.25 in.S6
XEROX 820, 860 8 in.A1
ZEDA 580SH
Zenith Z89 + Magnolia CP/M-80P7
Zenith Z89 + Zenith CP/M-80P7
Zenith DD/SSSK
Zenith DD/DSSJ

Program names and computer names are generally trademarks or service marks of the author or manufacturing company.
All Lifeboat 8-bit software requires SB-80 (or other CP/M-80 compatible disk operating system) unless otherwise stated.
All products are subject to terms and conditions of sale.

Send for full Lifeboat Associates Software Desk Reference with descriptions of all the above plus a whole lot more.

LIFEBOAT HAS THE ANSWER

with software, service and support from its offices in the U.S.A., U.K., Switzerland, W. Germany, France, and Japan.

LIFEBOAT ASSOCIATES • 1651 Third Ave., N.Y. 10028 • (212) 860-0300
TWX: 710-581-2524 (LBSOFT NYK) • Telex: 640693 (LBSOFT NYK)



An Introduction to Microcommunications

Davis Foulger

The first thing I did when I woke up this morning was turn on my computer. By any but the most computer-freakish of standards, that is a somewhat radical action. My wife, for one, considers it positively weird, although she is getting used to it. In a few years, however, the action may be about as commonplace as opening a newspaper over the breakfast table. Indeed, that's exactly what I was doing.

About two minutes after I turned my IBM Personal Computer on, my microcommunications software was loaded. That done, I dialed up the local Telenet access port and accessed Dialcom, a Silver Spring, MD based electronic mail and computer timesharing service where I have an account. The "greeting" I received on the service informed me that two pieces of mail were waiting for me in my electronic mailbox. Tucking that piece of information away for future reference, I typed in UPI and started a controlled journey through the "newspaper's newspaper". As is my habit, I approached the news selectively, entering in a set of keywords that would take me on a journey through that subset of the day's news which would be of greatest interest to me.

When I finished reading this "electronic newspaper", I went to check the mail. In reading the messages in my electronic mailbox I found that one was a reply to a letter I had sent almost a week ago. The other, however, was a response to a message I sent last night. I filed one message, replied to and deleted the other, and got off Dialcom. The entire transaction had taken about fifteen minutes. Dialcom returned me to Telenet, from which I then accessed the computers at the University of Wisconsin, the site of a computer conference in which I have been participating.

The conference was actually four distinct sessions, each of which attracted different people. One, entitled HCT*ICA, was a discussion of how the Human Communications Technology Interest Group of the International Com-

munications Association could use computer conferencing to advantage. A second, called PARTY*LINE, was an open ended "conversation" among participants in the conference. The topic bound rules of HCT*ICA were nowhere to be found in PARTY*LINE, and the topics ranged from requests for (and offers of) information to discussions of exotic, yet-to-be-built communications equipment like "Feel-A-Phone" (a telephone with a bionic extension that allows people to shake hands and do other "handiwork" long distance). A third, which I had started in the middle of the conference, was a forum where graduate students were discussing the electronic communication-related dissertations they were writing. It was called DISS*DISC. Finally, a fourth session, called COMM*TECH, was a sort of open national seminar about communication technologies.

As I was involved in all four conferences, I accessed them all, one by one, saving a disk-transcript of the messages I received. The transcript would allow me to come back later and read the comments more carefully. Connect time is too expensive to waste reading messages selectively. As I read the new messages in the sessions, I prepared replies to various messages and when it came my turn to write messages to the sessions, I transmitted both the messages I had just written and some extended messages I had prepared the night before. Twenty-five minutes after starting, I was finished. I had received about fifty new messages and sent five.

That night, after work, I received a call from a friend. We talked for a while, and then wanting to exchange some files we had written to our computers, switched over to our modems and "talked" through our computers. It was a local call and we let it go for most of an hour. About that time my wife called me for dinner.

A few years ago, that sequence of

events would have been Science Fiction. At best, it would be seen as being a few years distant. At worst, it would have been seen as fanciful. But that is exactly what I did do today. In fact, I follow a similar routine almost every day, checking out the news and the mail via my computer in less time than it used to take just to read the newspaper. Indeed, when I finish this article, I will probably submit it to *Lifelines/The Software Magazine* the same way - saving time, effort and (maybe) money for both of us.

Why communicate?

Few people would buy a microcomputer as a substitute for the telephone or newspaper. Truly the greatest value of micros is their capacity to accomplish a range of useful tasks without a communications link to other computers. Word processing, spreadsheet mathematics and other applications software more than pay back the cost of the computer for most users. Still, one out of every ten microcomputer owners have bought communications capabilities for their microcomputers. And even with the microcomputer market growing rapidly enough to guarantee rapid growth in the microcommunications market without any increase in the percentage of microcomputers that also microcommunicate, that percentage is increasing and is likely to reach 25% or more by the end of this decade.

Although there are variations in the ways that people microcommunicate and the applications they use the microcommunications medium for, there are basically two reasons for microcommunicating:

- (1) *Communication with other microcomputer users*
- (2) *Access to remote timesharing computers*

This article will explore both of these general applications of microcommunications, looking at the applications

that different users put the medium to, the motivations behind those uses, and the costs involved, particularly when compared to comparable communications systems. An article in a future issue of *Lifelines/The Software Magazine* will look at the necessary and ideal characteristics of microcommunications software.

Communications With Other Users

A friend of mine has wisely noted that the first uses of microcommunications are not likely to be the ultimate ones. In the long term, microcommunications is likely to develop into a dominant mode of interpersonal communication. In business settings, this will happen within the next few years, because some of the advantages that I'll discuss can have a positive impact on both productivity and the quality of work. But as voice recognition moves from promise to reality, microcommunications may actually supersede the telephone as the dominant mode of mediated interpersonal communication in the U.S. Although some users already use a mode of interpersonal communication, the single largest application to which users currently seem to put microcommunications is program exchange. This application is a very practical one, as it solves one of the larger problems that face microcomputers today.

Overcoming Disc Incompatibility: Users who wish to exchange the programs they write between different machines are inevitably faced with the problem of disk incompatibility - the fact that very few microcomputers are capable of reading disks written for other microcomputers. Even when programs share common operating systems and systems use the same size and type of disk, system software will often prevent disk exchange. Users who wish to exchange programs between different machines not only face the differences between five and eight inch disks, hard sector and soft sector disks, single density and double density disks, and single sided and double sided disks, but the less tangible variations in the way information is packed on the disk.

IBM's *de facto* standard for eight inch disks eliminates disk compatibility problems for many eight inch machines,

but even among eight inch machines that standard is not universal. In fact, buying the software/firmware that allows some eight inch machines to read IBM-format diskettes can cost more than microcommunications does.

The number of bytes stored on a soft sector, single sided, double density disk can fall anywhere along a continuum that ranges from the 160 Kbytes stored on an IBM Personal Computer PC-DOS format diskette to the nearly 700K bytes stored on some S-100 bus machines. In the five inch world, there simply aren't any standards, although some are beginning to talk about using the IBM Personal Computer's PC-DOS disk format as a *de facto* standard in the world of double density storage. A single set of universally applied standards for exchangeable storage media is not, moreover, likely to appear. There are simply too many highly successful machines operating in too many disk formats to allow any easy movement toward standards.

Microcommunications offers what is probably the single most important way out of this disk incompatibility problem. Users who are equipped for microcommunications simply dial each other up on the telephone and "talk" through their computers, exchanging files, including programs, in the process. The "equipment" needed to make this kind of program exchange over the telephone is expensive compared with the cost of the \$5.00 disk it replaces, but costs don't count when disks cannot be exchanged. There are, moreover, other advantages to exchanging files this way.

Speeding The Software Development Process: Key among these advantages is the speed with which software can be exchanged. There are basically three ways in which users can exchange software. They can mail it to each other, exchange disks in person, or microcommunicate it. No matter how close two people live, the mail will take at least 24 hours. More often than not, it will take longer than a week. Exchange in person requires that people coordinate their schedules so that they can be in the same place at the same time. That coordination may not be difficult, but it does slow the exchange and will often require one or both of the participants to go somewhat out of their way.

By contrast, microcommunicating the program doesn't even require the software developer to leave his or her computer. This can be particularly valuable in the program development process, as it allows the developer to quickly try out new features on other users. All you need is another user with microcommunications capabilities and a machine that can run the software. If, moreover, the other user has an operating auto-answer modem and host communications software, they don't even need to be at their computer to receive your software. Indeed, the ease and speed with which users can microcommunicate software is such that the microcommunication of software would probably be widespread even if the problem of disk incompatibility were solved.

A New Mode of Conversation

This ease and speed also makes microcommunications a highly enjoyable way of conversing with someone. Indeed, microcommunications represents the bare bones of a mode of conversation that may well revolutionize the way people talk to each other within twenty or thirty years. Even now, the microcomputer has some major advantages over the telephone as a conversational medium.

A first advantage has already been implicit to the discussion of program exchange. Microcommunicators can save files that are sent to them by other microcommunicators. While these files are often programs, they can be anything, including the text of this article. In truth, literally anything that can be stored in a computer can be microcommunicated to another computer and saved there.

Once saved, moreover, it exists in an electronic form which can not only be read at any time, but which can be easily edited and revised by the recipient. This is particularly valuable if the people who are communicating are working together over long distances (and even short distances) on projects which require a document as the final product. People who are, for instance, collaborating on the writing of an article, can rapidly shuttle easily revised electronic drafts of documents to one

(continued next page)

another; this speeds up the writing, and ultimately improves the quality, of the final product.

Saving An Electronic Transcript: This ability to save information that is microcommunicated is not, moreover, limited to saving the files that have been prepared and sent by the other party. The microcommunicator can easily save an electronic transcript of his or her conversations with others through the computer.

The problems of keeping a record of what has been said is one of the biggest problems with conversation on the telephone. After talking with someone for twenty minutes, it is not always easy to reconstruct what has been said. You often cannot remember what you said, what the other person said, or what it was that took the conversation down a given path in the first place. The tape recorder does, of course, provide a means of keeping a record of a telephone conversation, but it is not a convenient or easy-to-transcribe means. This problem is solved in computer communication because you can easily keep a transcript of the conversation by simply saving what is being said to disk. It is easy to do, does not generally slow down the conversation, and is saved in an electronic form that is easy to read, edit, and apply to other purposes.

Sexual Microcommunications? Another advantage of the medium is its potential for simultaneity. If your microcommunications software is written to allow it, communication can become literally simultaneous, with one person responding to the other's comments even as the other person is writing them. This is made possible by the nature of the microcommunications carrier signals and the flexibility with which the microcomputer can be programmed to transmit and display information. This kind of communication capability is close to impossible in any other communications media except face to face interaction. Even there, the only conditions under which a comparable level of simultaneity can be reached is in the sex act. This kind of feature is not universally applied in existing communications software packages, but it can be. It is, plain and simple, a function of software.

The World Of Microcommunications Games

One area in which users will appreciate this capacity for simultaneity is in the playing of multi-player computer games. The kinds of games which would be suited to microcommunications are not yet being marketed, but there are several potential characteristics of such games that could make them highly popular. First, the machine reverts to a battleground for a contest in which there can be clearcut winners and losers, where strategy becomes more flexible. The two major problems with most person *versus* machine computer games are the less-than-clear definition of what constitutes winning and the singularity of the strategy which ultimately beats any given game. Games with this characteristic can already be found in play on mainframe computers at schools across the country, but are uncommon on microcomputers.

A second characteristic could prove more interesting, however, as each player, with a microcomputer at his or her disposal, can potentially use that microcomputer as an assistant. In playing chess, for instance, the players could ask their computers for advice on moves. In playing a super star game, the computer could be programmed to calculate vectors, keep track of friendly and enemy ships, give warnings of impending problems and opportunities, and even take care of minor operational decisions. In this kind of game, programming skill could quickly become as important as reflexes; and strategy and planning would become an even more important component of computer gamesmanship.

Cutting Communications Costs

If all this sounds expensive, think again. Several factors work to keep the cost of microcommunications low. Under many circumstances, a conversation via a microcomputer is less expensive than a telephone call. Under some special conditions, it can even be cheaper than a first class letter.

Consider, for instance, the amount of information that can be exchanged in a

given period of time when talking on the telephone and when microcommunicating. The rate at which most people speak generally ranges between 120 and 160 words per minute. It is difficult for most people to talk faster than that and still be understood. Contrast with that a typical microcommunications system equipped with a 300 baud modem (baud is the communications word for bit - 300 baud means 300 bits per second or roughly 37.5 characters per minute). The 300 baud microcommunications system is capable of transmitting and receiving information at a rate of about 350 words per minute - two to three times the speed of typical voice communications. At 1200 baud, another common communications speed, that rate quadruples, and voice communication pales by comparison.

These speed increments can be deceiving, as most people cannot type at speeds that even approach the 120 words a minute at which almost anyone can talk. More typical typing speeds run at about twenty to forty words per minute. Microcommunications can still translate substantial savings on communication costs, though. If messages are prepared off-line before the call is made, typing rates become unimportant. The communication can be sent as a file at 350 words a minute or more, cutting communication costs by at least 50% and often considerably more. If the message is reasonably short, moreover, it can sometimes be transmitted for less than the cost of a first class letter. A late night call between New York and Los Angeles on ITT's Citi-Call long distance telephone network costs well under 10 cents a minute. Microcommunicators using such a connection at 300 baud might easily send a two- to four-page letter for less than twenty cents.

Access To Remote Timesharing Computers

Many microcommunicators enter the microcomputer world after years of using mainframes and minicomputers in school and business. Thus, although the range of applications and the potential advantages of microcommunicating directly with others account for a large portion of the users who currently microcommunicate, there are many microcommunicators who never engage in this kind of direct user-to-user

communication. These users, for the most part, use their microcomputers to access remote timesharing computers. This second use of microcommunications is becoming increasingly important as the range and quality of information, communication and computing services available on these centralized services increase and improve.

The world of timesharing and quasi-timesharing is a cluttered one. Even in an article which restricted itself to exploring only commercial services, the number of companies that offer timesharing services to users would be too long to give many adequate mention. In general, the range of services that offer timesharing and quasi-timesharing services can be divided into four broad classes. First among these are commercial services which generally restrict their marketing efforts to business, government and academic audiences. Few microcommunicators will find themselves regularly accessing these services except as a function of their work. They are generally too expensive for individual users.

Consumer Timesharing Services: Individual users are generally targeted by the consumer timesharing services. These services, which currently include *The Source*, *CompuServe* and *Dow Jones*, offer users a more limited range of services than are available on industry-oriented timesharing services, but offer those services at realistic prices.

Individual users may also have access to timesharing computers at school and at work. Most universities have some sort of timesharing computer system that is available for use by students. Although students will often face restrictions on the circumstances under which they can use university equipment, the option is often a quick means to low-cost timesharing access for the single user group that can probably least afford commercial services. Access to business computers can be much more restrictive, but many companies encourage professionals and executives who use company computer resources at work to maintain such access at home.

Quasi-Timesharing Via Bulletin Boards: A final source of what might be called "quasi-timesharing services" can be found in the hundreds of electronic bulletin boards that already dot

the U.S. These services, which are often available to users at low or no cost, often consist of little more than a modem, a microcomputer and some bulletin board software. As many of these systems can only communicate with one user at a time, they are not true timesharing systems, but many of the applications and services that are available to users of commercial and consumer timesharing services can be found on these electronic bulletin boards.

It should be noted that there is no special reason why a user needs a microcomputer to use these services. Most are geared to terminal users. The microcomputer will frequently lower the cost of such access, however, by allowing the user to do a large percentage of his or her message preparation offline, and by allowing the user to store the information received from the timesharing computer for later examination. A microcomputer will also allow the user to change the microcommunications software employed, to suit the particular application and the desires of the people involved.

Applications Of Timesharing

The wide range of available timesharing services reflects, in part, the diversity of applications offered. Among the user applications obtainable on various remote computer timesharing services are *electronic communications*, *electronic publishing*, *information retrieval* and *extended computing resources*. Not all of these services are offered by all timesharing computers. Indeed, many commercial timesharing companies have concentrated their efforts on offering only a limited subset of these services. All, however, are offered by the major consumer timesharing services.

Extended Computing Resources: A friend of mine once commented that a microcomputer can do almost anything, if you give it long enough. Still, the speed and memory capacities of the typical 64K microcomputer limit its applications. Many software packages simply cannot be fully implemented on the typical microcomputer and many software distributors are reluctant to even try adapting their software to the

microcomputer market. Given this backdrop, it should come as no surprise that one of the biggest reasons for using remote timesharing services is the extended computing resources they offer the microcommunicator.

A wide range of applications software is available to users through commercial timesharing services, and although the options are more restricted on consumer timesharing services, the microcommunicator will find a wide range of software options available. These options include advanced statistical packages, portfolio managers, financial planning software, mineral exploration and management programs, and engineering and research and development software.

Another resource that will be of interest to some microcommunicators is the games programs that can be accessed on the consumer timesharing services. Although many of the games that are available on these services can be implemented on microcomputers, a whole new class of multi-user games can be expected to be implemented in the future. In these games, the microcommunicator will be able to program his or her microcomputer as an electronic assistant turning games of skill and rapid reaction into games of strategy. Users are not limited to pre-packaged software, moreover, and may find value in implementing their own extended programs on the remote timesharing system.

Electronic Publishing: One of the most promising new services that timesharing microcommunications introduces is the publication of newspapers, magazines, newsletters, articles, and even whole books without paper. Today that promise is reality. Microcommunicators are not only capable of reading the *United Press International* wire service, as I did this morning, but *The New York Times*, *The Wall Street Journal*, *The Washington Post*, *The San Francisco Chronicle* and a number of other newspapers and regular publications. These well-known publications are just the tip of the electronic publication iceberg, however; microcommunicators will find electronic publications devoted to a wide range of topics. While many are electronic versions of "hard copy" publications, some can be found nowhere else.

(continued next page)

Electronic publishing remains highly experimental. Good solutions to the problems of transmitting graphics, displaying advertising, getting people to the information they want and distributing revenues and costs are far from being found. But powerful incentives to find those solutions are the advantages of electronic publishing in terms of saving natural resources, increasing the responsiveness and accessibility of publications and making it possible to publish a far wider range of information than has ever before been possible. Even at this experimental stage, however, many microcommunicators will find the already-available electronic publications a convenient alternative to hard copy.

Information Retrieval: Even those who aren't interested in reading publications on-line will find timesharing a useful adjunct to their reading if they spend much time looking for specific kinds of information. Just about any index of publications that can be found in a library can be accessed through a computer - and with considerably greater facility. A search that might take several hours of ploughing through twenty or thirty different library volumes can be completed in a few minutes through a database timesharing service like *Lockheed Dialog* (currently alone in offering their services on terms that most microcommunicators can afford). The service is still expensive, running \$60 or more an hour, and will not be required by every microcommunicator, but for those who do a great deal of library research, it can be invaluable.

Bibliographic databases are not the only kind of information retrieval services available on timesharing services, however, and other users will find other kinds of timesharing database services. These services include chemical and econometric data, the full text of newspapers going back over periods of six months or more, and software that permits users to build their own timeshared databases.

Electronic Communications: However useful the above applications of remote timeshared microcommunications may be, the most useful applications of remote timesharing may be in the communication itself. Indeed, microcommunicators will find that timesharing offers both greater flexibility in

communication and a way to further shave the costs of long distance microcommunications. There are already four distinct modes of microcommunication available through various time-sharing services. These modes - *computer conversation, electronic mailboxes, computer conferencing and electronic bulletin boards* - offer users the opportunity to tailor the communication resources they utilize to their particular needs.

Electronic mailboxes are exactly what they purport to be, a place to put letters that people send to each other electronically. The biggest value of the electronic mailbox is that it is, in fact, a mailbox. To get the message, you don't have to be there when the message is left. Remembering the message is the job of the timesharing service. The fact that the mailbox is electronic leads to other values, moreover. Electronic mail can be opened from anywhere you can interface a terminal or microcomputer to a telephone. My mailbox on *Dialcom* is just as accessible to me from Boston, New Orleans or Los Angeles as it is from my home in Connecticut. This feature is a tremendously convenient one, especially for people who travel frequently on business. A secretary or associate can easily leave a message in an electronic mailbox with confidence that the message will get where it is supposed to go quickly.

This speed is another value of the electronic mailbox. Remember that one of the two messages I found in my electronic mailbox this morning was a response to a letter I sent last night. As my mother puts it: "Your grandchildren can open their Christmas presents in the morning and you can read their Thank you's in the afternoon". This savings in time can more than pay back the cost of an electronic mailbox all by itself, even if electronic mail were really more expensive than regular mail.

Computer conferencing and time-shared *electronic bulletin boards* can both be thought of as *public* variations on the electronic mailbox. Both store the comments of many people in a single file that can be read by many different people. The intent and organization of the two is quite different, however, and they are appropriate for rather different sets of tasks. The computer conference is organized sequentially as a kind of asynchronous small group

meeting. Participants enter and leave the meeting pretty much as they please, with the computer keeping track of how much of the meeting they have read and any comments they add to the meeting. Computer conferences can be just about anything you might want them to be. They can be tightly structured and focused on a specific issue or left open to a general discussion. The conference sessions I participated in this morning included conferences that operated at both of these extremes and at points between.

The computer conference is a valuable alternative to the conventional business meeting, especially when the speed with which the group must reach a decision is less important than the quality of the decision. Computer conferences are, understandably, slow. Participants move in and out sporadically and it may take a week to generate the amount of conversation that might be generated in an hour of face-to-face discussion. It is not uncommon, moreover, to find few times when two group members are actually accessing the conference at the same time. But that slowness has value. Participants have time to digest one another's comments and are not pushed into making statements in haste. Ideas tend to become more important than emotion and it becomes difficult for group members to exert control over the meeting. As a friend has commented, "it is difficult to have an argument in a computer conference".

By contrast, electronic bulletin boards tend to be topically organized, with users choosing the things they want to read and write about by selecting options from menus of alternatives. More will be said about bulletin boards below, as they are commonly implemented as quasi-timesharing systems.

Computer conversation is really not appreciably different than direct user-to-user microcommunications, and unless portions of the conversation are written before the user accesses the timesharing service, it can be the most expensive of the microcommunication options that are available through a time-sharing service. It can, however, be considerably cheaper than a direct microcomputer-to-microcomputer connection. A one-hour telephone microcommunications connection between New York and Los Angeles via ATT,

even during the lowest rate periods, will cost over \$13.00. That same connection can be made for less than half that amount (almost two-thirds less) using CompuServe or The Source during those same periods.

These savings multiply the speed-related savings on connect costs that microcommunicators can already gain over a voice telephone conversation. If messages are prepared off-line and transmitted at night using a 300 baud modem and a timesharing service like The Source or CompuServe, \$13.00 worth of late night telephone conversation can be microcommunicated for a price that will range somewhere between \$1.60 and \$3.00, depending on how fast you talk and the service you use.

Quasi-Timesharing Electronic Bulletin Boards

Community electronic bulletin boards offer microcommunicators another class of service with many of the same options available on consumer timesharing services. As these services are often implemented on microcomputers, a microcommunicator may have to make use of several bulletin boards to match the range of services offered by the most limited timesharing service. Still, they offer the user many timesharing service options and the availability of these services at low or no cost makes them an important option for cash-strapped microcommunicators.

Operated by local computer clubs, businesses and other organizations that want to establish a central source of information exchange, these local electronic bulletin boards are established for a variety of purposes. Computer clubs start bulletin boards to facilitate program exchange, prepare and distribute newsletters, and perform other communications functions. Businesses start the bulletin boards as a way of advertising themselves and their products, and as a way of taking orders from customers. Other organizations provide a wide range of information through bulletin boards.

Although these bulletin boards are often available to microcommunicators without cost, it should be noted that

"free" is not necessarily cheap. Use of a community electronic bulletin board will still require the microcommunicator to pay telephone charges that can be more expensive than consumer timesharing costs unless the call is a truly local one. A one hour late night phone call to a person just two towns removed from the one where I live (about ten miles away) costs me \$3.66, most of the cost of access to The Source or CompuServe during that same period. Calls to towns that are further away quickly escalate in cost to the point where it is cheaper to communicate via the Source or CompuServe than it is to make a phone call. Of course, these economics change if you must access the consumer databases via Tymnet or Telenet (if the access number for Tymnet, Telenet or the consumer database is not a local call for you). But the point remains - "free" microcomputer-based local electronic bulletin boards do not necessarily save money for users.

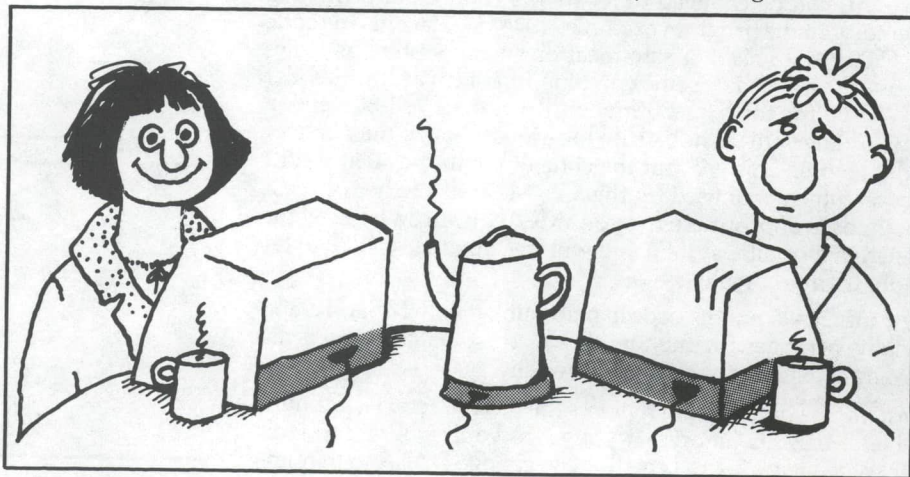
Listings of various electronic bulletin boards are available from a number of sources. One source of this information is an electronic bulletin board operated by Novation, the modem manufacturer. Novation lists electronic bulletin boards under item 18 on their bulletin boards menu. It can be accessed by dialing (213) 881-6880 with a 300 baud modem and using the password CAT. Another information source is AMRAD (524 Springdale Avenue, McLean, VA 22101), which will mail you a hard copy directory of community electronic bulletin boards for \$1.00.

In Microconclusion

Even if you bought (or are considering buying) your microcomputer primarily as a standalone work station, there are

a lot of good reasons to give it microcommunications capability. The hobbyist benefits from the ability to quickly and easily exchange the software written with friends. The computer games player gains access to a way of playing new kinds of interactive games. The software developer benefits from the ability to try out innovations in software on test audiences, without having to go to the trouble of meeting that person and without the delay of shipping software through the mail. From the convenience of home, professionals gain access to the computerized files and databases they keep at work. Executives and managers can stay on top of important correspondence from wherever they find a phone and terminal.

Microcommunications has something to offer just about anybody who uses a microcomputer. And because it doesn't cost much compared with some of the alternative modes of communication currently available, it really has something to offer just about anybody who does much communicating over the telephone, or who finds that the speed and reliability of first class mail leaves something to be desired. Properly used, microcommunications can be faster, less expensive and more convenient than a telephone call; in some circumstances, microcommunications can even beat a first class letter for cost and reliability. Other characteristics of microcommunications can make the medium both more convenient and more forgiving than other media. Cost effective microcommunications depends on the use of good communications software. The necessary and ideal characteristics of such software will be examined in a coming issue of *Lifelines/The Software Magazine*.



Software Notes

Modifying Control-C In MBASIC

Bill Norris

A frequently recurring MBASIC question is "How can I keep my program from being aborted when a ^C is typed on the console?" Although this problem may be solved by several different methods, each suffers from one or more drawbacks. The best solution would be for the interpreter to have a command which would redefine or disable ^C checking. (If you wait for this though, you'll probably see complex variables implemented in FORTRAN first.) Another method would be to modify the system so that a ^C is translated or ignored. This can be done by changing the BIOS or by using a keyboard redefinition program. This won't help someone using your program on another system, and what do you now do when a ^C is really needed? The method described here should work with any version of MBASIC, including the compiler, although the variable names would have to be shortened to be compatible with pre-5.0 interpreters. It has one limitation, however, in that the operating system under which it runs must have implemented its BIOS as described in Figure A. That is, the CONIN vector in the BIOS JUMP TABLE must point to either a jump or a call instruction which branches to the real console input routine. As most of the BIOS implementations which I have seen utilize this method (including those done by Lifeboat Associates), the program should work for most of you. Line 45 will notify you if this is not the case.

The rest of the program works as follows:

- 1) Line 15 is a subroutine which receives a variable (TEMP) containing an 8080 style two byte address. It produces as output two variables (TEMP1 & TEMP2), each containing the value of one of the address bytes.
- 2) Line 35 is a DATA statement which contains two machine language subroutines. The first one converts the ^C, and the second restores the system to it's original state. The routines are listed in figure B.
- 3) Line 40 gets the value (CONIN) of the address pointed to by the BIOS console input vector. This is the address that MBASIC will call for console input. The jump table vector is ignored by MBASIC after its initialization.
- 4) Line 45 does some testing to see if the routine won't work. If the error message on this line doesn't get printed and yet the program still doesn't work, then the first instruction in the console input routine probably is a call to a console status check. Fortunately, I haven't seen this yet.
- 5) Line 50 determines where the machine language routines go. This can be changed to locate the routines anywhere in memory. In the program example, 1024*64-256 puts the code at FF00 hex. This is a safe location in my system as it lies above the BIOS. For some combinations such as the Lifeboat CP/M-80 for the TRS-80 model II, using CP/M-80 version 2.25 or later, this is not a safe location. Change the value of ADDR to 8. This will put the code down in the 8080 restart region which is unused by this CP/M-80. If neither of these methods is appropriate for you, MOVCPM can be used the generate a smaller system and will free up at least 1000 bytes more than you really need.
- 6) Lines 55-85 put the code in place, link it with MBASIC and finally, arranges for the undoing of these patches when the program exits to the operating system.

Last note: This program replaces the ^C character with a null, which MBASIC ignores. If you want your program to look for someone trying to type ^C, just change DATA item number 8 (between 62 and 201) from 0 to some other value.

```
10 GOTO 35 ***** BASIC-80 routine to filter control-C *****
***** Written March 15, 1982 - Bill Norris. *****
15 TEMP$=HEX$(TEMP) :
TEMP1=VAL("&H"+MID$(TEMP$,3)) :
TEMP2=VAL("&H"+LEFT$(TEMP$,2)) :
RETURN
35 DATA 205,0,0,254,3,192,62,0,201,33,0,0,34,1,0,33,0,0,34,0,0,195,0,0
40 TEMP=PEEK(2)*256+10 : CONIN=PEEK(TEMP)+PEEK(TEMP+1)*256
45 IF PEEK(CONIN)=195 OR PEEK(CONIN)=205 THEN GOTO 50
ELSE PRINT "Not safe to patch CONIN." : STOP
50 ADDR=1024!*64!-256! : CONIN.LOW=PEEK(CONIN+1) : CONIN.HIGH=PEEK(CONIN+2)
55 FOR I=0 TO 23 : READ TEMP : POKE ADDR+I, TEMP : NEXT I
60 POKE ADDR+1, CONIN.LOW : POKE ADDR+2, CONIN.HIGH
65 POKE ADDR+16, CONIN.LOW : POKE ADDR+17, CONIN.HIGH
70 TEMP=CONIN+1 : GOSUB 15 : POKE ADDR+19, TEMP1 : POKE ADDR+20, TEMP2
75 TEMP=ADDR : GOSUB 15 : POKE CONIN+1, PEEK(1) : POKE CONIN+2, PEEK(2)
80 POKE ADDR+10, PEEK(1) : POKE ADDR+11, PEEK(2)
85 TEMP=ADDR+9 : GOSUB 15 : POKE 1, TEMP1 : POKE 2, TEMP2
90 NEW : REM Most of you will want to change NEW to a CHAIN statement.
```

Figure A

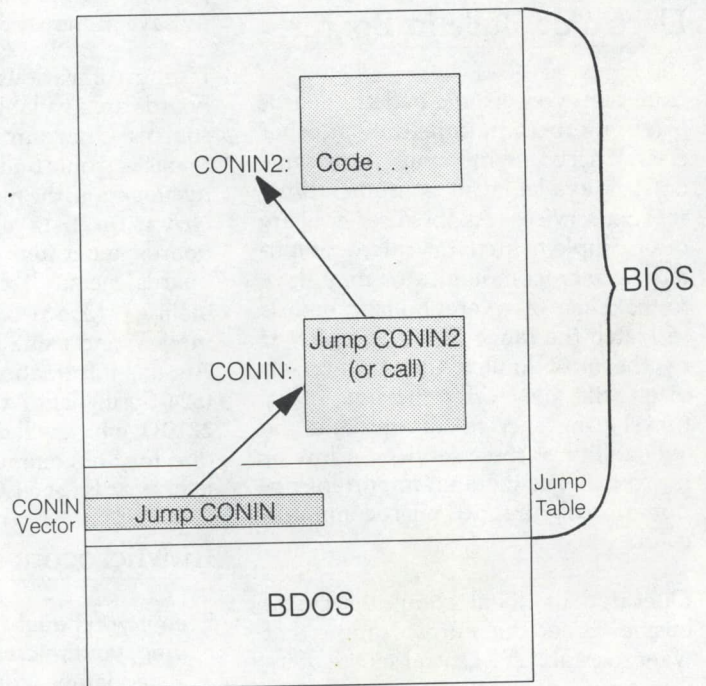


Figure B

```
CALL CONIN2 ; Get character
CPI 3 ; Check for ^C
RNZ ; Return if not
MVI A,0 ; Else convert
RET ; and return

LXI H,WBOOT ; Restore the
SHLD 1 ; jump at 0
LXI H,CONIN2 ; Undo the patch
SHLD CONIN+1 ; made in BIOS
JMP 0 ; Back to system
```


Product Status

Reports

New

Products

The products described below are available from their authors, computer stores, software distributors and publishers.

BACKUP

TRI-L Data Systems

This program is designed for users of CP/M-80 2.x who have large-capacity winchester disk drives. This utility permits data to be copied from winchester drives onto more than one smaller capacity floppy disk. BACKUP replaces a second hard disk drive, or tape drives for archiving disk data.

The FinalWord

Mark of the Unicorn

This word processor, written in C, incorporates such features as multiple line spacing, automatic word wrap, automatic insert mode, global search and replace, justification, super- and subscripts, multiple fonts, underscore, move and delete blocks; cursor positions by word, sentence, line, paragraphs, beginning and end of text are supported.

In addition, The FinalWord allows the user to set up headings and a table of contents automatically, to footnote, and to create index entries. These functions are implemented when the user embeds certain codes in the text. Two files can appear on screen at once, while one is edited, and one file can be printed while another is edited. True proportional spacing is supported. User-defined commands can be created, and the product can recover deleted text.

The FinalWord requires CP/M-80, CP/M-86, or IBM PC DOS, along with a 56K system.

LP Disk

Agricultural Software Consultants, Inc. This linear programming problem solver handles up to fifty variables and

fifty constraints (maximize, minimize, less than, more than, equal to). The algorithm is written in machine language to enhance speed; problems may be printed, saved to disk. Data may be altered; the solution includes price ranges and shadow prices, along with an appraisal of the maximum error the solution may contain.

LP-Disk requires CP/M-80, 48K of memory, one disk drive.

Mr. EDit

Micro Resources Corporation

This screen-oriented text editor is designed for non-memory mapped video display terminals; the author intends it to be user-configurable for any such terminal. Commands can be given by text or command keys, and can be English language or abbreviations. The command keys remain active in command mode, and the cursor is maintained in the screen text at all times. A single key can be defined as a series of commands. Insert and overwrite are supported, as are word wrap and paragraph fill.

Such commands as search/replace, print by line or area and other cursor control and delete commands are included. Horizontal windowing, indent levels for structured programming, primary and secondary input and output, command file handling, and iteration macros are some of the features included in this product.

Mr. EDit runs with 8080 or Z80 CPU's, requires 24K or larger transient program space (TPA), a terminal with a 12 by 64 display, and CP/M-80 or MP/M-80 2.x.

SID-86

Digital Research

This general purpose debugger can be used to debug software or configure Digital Research 16-bit operating systems for 8086/8088 computers. It can read compiled programs in any high or low level language running under CP/M-86, concurrent CP/M-86, or MP/M-86. It has basically the same features as the 8080-based SID. It sets up to 16 permanent breakpoints with

associated pass counts, has high level trace, symbolic assembly and disassembly, and expression handling.

Toricelli Author

the Answer in Computers

This product is a tool primarily designed for creating a Computer Aided Instruction (CAI) course. A course contains up to 250 "pages" or screens, with quizzes, test answers, and closing statistics (test results, pages completed) where desired. A built-in full screen editor allows manipulation of data; the course author may use WordStar-like or WordMaster-like commands. The answers to quiz questions and response paths of the student can be determined by the teacher. Blank pages may be inserted, pages deleted or copied.

This product requires CP/M 2.2x, an 8080 or Z80 CPU, total disk capacity of 32K, 48K RAM, a cursor addressable terminal; if a printer is used, it must be an 80-column one.

New Publications

Periodical Guide For Computerists

The 1980-1981 version of this index has been released, listing articles which have appeared in *Lifelines/The Software Magazine*, *BYTE*, *Dr. Dobbs*, *Kilobaud*, *Microsystems*, and twenty other publications. They are classified in broad subject headings which are further divided into more specific subject categories (i.e., Languages, Pascal). The author, article name, magazine, issue and page are listed and the article is classified according to whether it's a book review, editorial, review, etc.

Practical Pascal Programs

By Greg Davidson and Lon Poole
This book contains a collection of Pascal programs classified under the categories of finance, management decision, statistics, science, and math. Instructions are given to allow users to modify the programs. Sample runs and practice problems are also included.

Software Vendor Directory

Micro-Software Services, Inc.
This directory lists more than 1800 software vendors, 123 hardware ven-

(continued next page)

dors, and twenty-two operating systems. 12,300 are categorized and indexed. The directory is available in book form or on disk, running under CP/M-80 with a database utility for finding information.

New

Versions

JANUS

Version 1.4.3

This new version includes some minor bug corrections and adds Integer Exponentiation to the features provided by JANUS. A library of string handling procedures has now also been appended, and the assembler has been sped up.

Microstat

Version 2.08a

The MBASIC version now contains a message warning the user not to mix Single and Double precision numbers.

PLAN80

Version 2.3

This update contains several improvements and bug fixes:

- 1- In an :OPTIONS section the options ZERO, DASH or BLANK may be specified to control the way in which zero values are shown; if there is no specification, dashes are used.
- 2- The "W", when used in the DISPLAY mode, now properly handles numbers in which the total number of digits before and after the decimal point is greater than 7.
- 3- An IF statement following an assignment to a cell or as the first statement of a RULES section no longer yields an Error 27.
- 4- The shift function has been expanded to work with :FOR statements. It is used in row or column rules to compute re-

sults from values in columns or rows other than the one in which the result is to be placed. For example, Cash Collection might be set to 60% of prior month's plus 40% of current month's sales. The :FOR statement may be used to tell PLAN80 to ignore certain columns or rows.

PANEL Version 3.0

The new system offers a number of improvements over version 2.2, of which the following is a summary:

- 1-Wide-screen terminals are supported.
- 2-The terminal definition file now includes specifications for up to 16 highlighting types, and also caters for additional cursor addressing methods (including ANSI standard).
- 3-New field-definition attributes are provided for right-justified, numeric, and currency fields, all handled automatically by the library subroutines.
- 4-Subroutine names and calling sequences have been simplified and rationalized, but not to such an extent that a few minutes work with a good text editor cannot change your existing programs.
- 5-Screen panel designs can now be optionally loaded at run-time, instead of being linked into the program.
- 6-The system is supplied linked and ready-to-run. The TAILOR program now includes a menu of predefined screens as well as allowing even more flexible screen and keyboard customizing.
- 7-A multi-key record retrieval program is supplied which sets up an instant 'electronic filing system' to match any screen panel layout.

Plink-II

Version 1.14

This version implements several bug fixes and some important enhancements. Here are the problems which

have been remedied:

- 1-Plink-II no longer delivers a false error #71 (program too large) message when a program using Microsoft BASIC version 5.3 or above of more than 180 Hex bytes in data area is linked.
- 2-A false error #85 is no longer generated when linking a Whitesmiths' C program.
- 3-A bug #190 was occasionally occurring when a DEFINE <symbol1> = <symbol2> command was used; this has been fixed.
- 4-The last sector of an overlay an even number of 128 byte sectors in length now loads properly.
- 5-Plink-II no longer crashes when it encounters Pascal/Z programs with 8 character identifiers.

The following are new features included in this update.

- 1-Microsoft .REL files are now linked more quickly.
- 2-Microsoft COBOL version 4.6 .REL files can now be linked by Plink-II; COBLOC and COBLBX files must be on disk.
- 3-Microsoft M80 version 3.44 .REL files containing link-time arithmetic expressions are now handled by Plink-II; if they are improperly formed an error #99 will occur.
- 4-Files produced by the new Whitesmiths' C compiler are handled by Plink-II; the header file is automatically included in the program, and the library is searched automatically. The free memory area is set to the proper address and the uninitialized data area appears as a concatenated common block.
- 5-Three error numbers have been added to signal a 4-byte integer, a relative fixup having been specified, and a loadable .0 file given as input; old and new formats are detected but cannot be used in the same program.
- 6-Most BDS C programs can now be linked as is. The run-time support program C.CCC must be on disk and is included in the program.
- 7-Four new statements have been added, specifically to identify

Phoenix Software Associates, Microsoft, Whitesmiths' or BDS C formats. If these statements are used, and they have no arguments, Plink-II does not have to identify a file's format itself.

8- Plink-II can output a linkable Entry Point File containing nothing but the absolute addresses of all global symbols defined in the program being linked. So no memory space is reserved and the entry file may be linked into a second program to permit that program to access symbols defined in the first program. Entry Point Files are useful when it becomes necessary to create a program whose executable code resides in several different files which are linked separately.

Quic-N-Easi
Version 1.4

This update offers several new features, the most important being a report

writer. The report writer allows up to twenty columns, permits breaks, totals, subtotals, and a floating dollar sign. A column can come from file data, be a constant or calculated results. The specifications for a report are entered by filling in blanks on a formatted screen, and can be saved. In addition, this new version includes dimensioned variables and color graphics. All applications developed during earlier versions will run with this one.

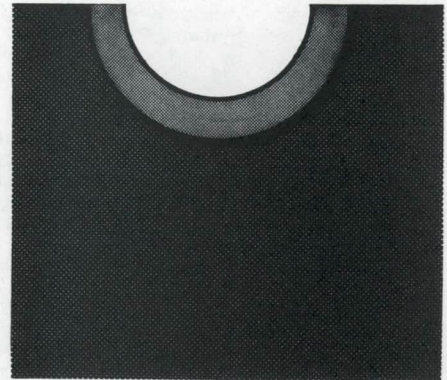
T/MAKER-II
Version 2.5.3

This update features three enhancements. A new function called DATA can be used to label data files with a drive name. A default data drive can be specified. T/MAKER will continue to look for all its program files on the currently logged drive, even if the data statement is used with a different drive.

The Tally function now accepts an option called ALL which will cause it to tally all non-blank lines in a file even

when they do not include a plus sign in the first column. It may not work correctly unless the file has been created with T/MAKER, however.

The Print function has been modified so that a few shortcuts may be taken for underlining and boldface when a full line is involved.



Software Notes

Notes On dBASE II, Version 2.3B

Michael Olfe

The Do case . . . enddo construct

"Do case . . . enddo" operates exactly as described in the manual, which may be disappointing to those expecting a "case" construct à la Pascal. The first "endcase" encountered will pair with the first "do case" statement, whether or not there were intervening "do case" statements. That is, this procedure does not nest.

With `x <> '0'` and `x <> '1'` the following sequence will always execute the inner "DO CASE" statement, and never print "x=2" or "x=3" (see Figure 1).

A note of warning on variable names

Strange and wonderful things can happen if you do not take care in naming database fields which have boolean data types. Take the case of a database called "database" with a field named "DONE", of type boolean, being opened with the loop in Figure 2.

If the first record in the database has a true value in the field 'Done', the program will terminate immediately with no prompting on the screen. The loop termination condition will be true if either the memory variable called 'Done' or the database field called 'Done' is true.

Moral: reserve sets of variable names for control loops e.g. "loopdone", "loopdone1".

```

stor f to done
use database
do while .not. done
  accept "Press Q to quit or D to do something" to c
  if !(c)='Q'
    stor t to done
  else
    < do something >
  enddo

```

Figure 2

Figure 1

```

do case
  case x='0'
    ? 'x=0'
  case x='1'
    ? 'x=1'

  do case
    case y='0'
      ? 'Y=0'
    otherwise
      ? 'Y<>0'
    endcase

  case x='2'
    ? 'x=2'
  case x='3'
    ? 'x=3'
  endcase

```

(continued next page)

VERSION LIST

May 7, 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
MR Memory Required

New Products and new versions are listed in boldface.

Product	S	P	MR	
ACCESS-80	1.0	8080/Z80	54K	
Accounts Payable/Cybernetics				Needs RM/COBOL. Runs w/CP/M-80, OASIS, UNIX
Accounts Payable/MC	1.0	8080/Z80	56K	For CP/M-80 2.2
Accounts Payable/Structured Sys	1.3B	8080	52K	w/It Works run time pkg.
Accounts Payable/Peachtree	07-13-80		48K	Needs BASIC-80 4.51
Accounting Plus		8080/Z80	64K	
Accounts Receivable/Cybernetics				Needs RM/COBOL. Runs w/CP/M-80, OASIS, UNIX
Accounts Receivable/MC	1.0	8080/Z80	56K	CP/M-80 2.2
Accounts Receivable/Peachtree	07-13-80	8080	48K	Needs BASIC-80 4.51
Accounts Receivable/Structured Sys	1.4C	8080	56K	w/It Works run time pkg.
Address Management System	1.0	8080		Requires 2 drives
ALGOL 60	4.8C	8080	24K	
ANALYST	2.0	8080	52K	Needs CBASIC2, QSORT/ULTRASORT
APL/V80	3.2	Z80	48K	Needs APL terminal
Apartment Management (Cornwall)	1.0	Z80		Needs CBASIC2
ASCOM	2.01	8080		
ASCOM/86	2.01	8086		Specify operating system: IBMPC/CPM-86/MS-DOS
ASM/XITAN	3.11	Z80		
Automated Patient History	1.2	8080	48K	
BASIC Compiler	5.3	8080	48K	
BASIC-80 Interpreter	5.21	8080	40K	w/Vers. 4.51, 5.21
BASIC Utility Disk	2.0	8080	48K	
BaZic II	03/03	Z-80		
Benchmark Word Processor	2.2			Give Name & Model #'s of the video terminal
Benchmark Mail List	1.1			Give Name & Model #'s of the video terminal
BOSS Financial Accounting System	1.08	8080	48K	Needs 2/3- drives w/min 200k each, & 132-col. printer
BOSS Demo	1.08	8080	48K	
BSTAM Communication System	4.5	8080	32K	
BDS C Compiler	1.46	8080	32K	w/'C' book
Whitesmiths' C Compiler	2.1	8080	60K	
BSTMS	1.2	8080	24K	
BUG/uBUG Debuggers	3.20	Z80	24K	
CBASIC2 Compiler	2.08	8080	32K	w/CRUN(2,204P, & 238)
CBS Applications Builder	1.33	8080	48K	Needs no support language
CIS COBOL Compiler	4.4.1	8080	48K	
CIS COBOL Compact	3.46	8080	32K	
FORMS 1 CIS COBOL Form Generator	1.06	8080		
FORMS 2 CIS COBOL Form Generator	1.1,6a	8080		CP/M-80 1.41 or 2.XX
Interface for Mits Q70 Printer				
COBOL-80 Compiler	4.6	8080	48K	
COBOL-80 PLUS M/SORT	4.01	8080	48K	
CONDOR II	2.06	8080	48K	
CREAM (Real Estate Acct'ng)	2.3	8080	64K	CBASIC needed
Crosstalk	1.4	Z80		
DATASTAR Information Manager	1.101	8080	48K	
Datebook-II	2.04	8080	48K	Needs 80x24 terminal, N/A for CDOS, CP/M-80 1.4, MP/M-80
dBASE-II	2.3B	8080	48K	
dBASE-II Demo	2.3B	8080	48K	
Dental Management System 8000	8.7A	8080	48K	Needs CBASIC
Dental Management System 9000	2.0	8080	48K	Needs CBASIC
DESPOOL Print Spooler	2.1A	8080		
DISILOG Z80 Disassembler	4.0	Z80		Zilog mnemonics
DISTEL Z80/8080 Disassembler	4.0	8080/Z80		Intel mnemonics, TDL extensions
Documate/Plus	1.4	8080	36K	
Documate/Plus/Demo	1.5	8080		
EDIT Text Editor	2.06	Z80		
EDIT-80 Text Editor	2.02	8080		
EM 80/86	1.01	8086		Specify operating system: IBM PC/CPM-86/MS-DOS
FABS-I	2.7	8080	32K	
FABS II	4.15	8080/Z80	48K	
FILETRAN	1.20		32K	1-way TRS-80 Mod I, TRSDOS to Mod I CP/M-80
FILETRAN	1.4		32K	Needs TRSDOS. 2-way TRS-80 Mod I, TRSDOS & Mod I CP/M-80
FILETRAN	1.5		32K	1-way TRS-80 Mod II, TRSDOS to Mod II CP/M-80
FinalWord	1.0	8080	56K	Runs under CP/M-80, CP/M-86 or IBM PC DOS
Financial Modeling System	2.0		48K	
FORTH (Timin)	3.5	8080	28K	
FORTTRAN-80 Compiler	3.44	8080	36K	
FPL 56K Vers.	2.6	8080	56K	
FPL 48K Vers.	2.6	8080	48K	
General Ledger/Cybernetics				Needs RM/COBOL. Runs w/CP/M-80, OASIS, UNIX
General Ledger/MC	1.0	8080/Z80	56K	Needs CP/M-80 2.2 or MP/M-80
General Ledger/Peachtree	07-13-80	8080	48K	Needs BASIC-80 4.51
General Ledger/Structured Sys	1.4C	8080	52K	w/It Works Package
General Ledger II/CPAids	1.1	8080	48K	Needs BASIC-80 4.51
GLECTOR Accounting System	2.02	8080	56K	Use w/CBASIC2, SELECTOR III
GLECTOR IV Accounting System	1.0	8080		Needs SELECTOR IV
GrafTalk	1.0			Requires 180Kb/drive. Available for certain Terminals Printers, & Plotters.

VERSION LIST

Product	S	P	MR	
HDBS	1.05A	+	52K	
HOE	2.1	8080	48K	
IBM/CPM	1.1	8080		CP/M 1.4 only!
Insurance Agency System 9000	1.08	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/Z80	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	52K	w/It Works Package
JANUS	1.4.3	Z80/8080		Also runs w/8086
Job Cost Control System/MC	1.0	8080/Z80	56K	Requires CP/M-80 2.2
JRT Pascal System	1.4	8080	56K	
LETTERIGHT Text Editor	1.1B	8080	52K	
LINKER		Z80		
LP-DISK	1.0	8080/Z80	48K	Also for TRS-80 I/III
MAC	2.0A	8080	20K	
MACRO-80 Macro Assembler Package	3.43	8080/Z80		
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	32K	
MAG/sam3	4.2	8080	32K	
MAG/sam4	1.1	8080	32K	Needs CBASIC
MAGSORT-C	1.0			For CBASIC
MAGSORT-M	1.0			For MBASIC
MAGSORT-R	1.0			For Compilers — BASCOM, FORTRAN-80, PL/I-80
MAILING ADDRESS Mail List System	07-13-80	8080	48K	
Mail-Merge	3.0	8080		
Master Tax	1.0-80	8080	48K	
Matchmaker		8080	32K	
MDBS	1.05A	+	48K	
MDBS-DRS	1.02	+	52K	
MDBS-QRS	1.0	+	52K	
MDBS-RTL	1.0	+	52K	
MDBS-PKG		+	52K	w/all above MDBS products
Medical Management System 8000	8.7a	8080		Needs CBASIC
Medical Management System 9000	2.0	8080		Needs CBASIC
Microcosm		Z80		CP/M-80 2.X or MP/M-80
Micro-SEED	B.10G	8080		
Microspell	4.3	8080	48K	
Microspell Demo	1.0	8080		For Dealers Only
Microstat	2.08a	8080	48K	Needs BASIC-80, 5.03 or later
Microstat for Apple	2.0	Z-80		
Mince	2.6	8080	48K	
Mince Demo	2.6	8080	48K	
Mini-Warehouse Mngmt. Sys.	5.5	8080	48K	Needs CBASIC
Money Maestro		8080/Z80	48K	CP/M-80 1.4 or 2.2
MP/M-I	1.0			
MP/M-II	2.0	8080	48K	Needs MP/M-80
Mr. EDit	2.0	8080/Z80	24K	Needs 24K TPA, 12 x 64 column terminal
MSORT	1.01	8080	48K	
Mu LISP-80/Mu STAR Compiler	2.12	8080		
Mu SIMP / Mu MATH Package	2.12	8080		muMATH 80
NAD Mail List System	3.0D	8080	48K	
Nevada COBOL	2.1	8080	32K	
Order Entry w/Inventory/Cybernetics Panel	3.0		44K	Needs RM/COBOL
PAS-3 Medical	1.78	8080	56K	Also for MP/M-80
PAS-3 Dental	1.64	8080	56K	Needs 132-col. printer & CBASIC
PASM Assembler	1.02	Z80		Needs 132-col. printer & CBASIC
Pascal/M	4.02	8080	56K	CP/M 2.4 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/Cybernetics, Inc.		Z80		
Payroll/Peachtree	07-13-81	8080	48K	Needs RM/COBOL
Payroll/Structured Sys	1.0E	8080	60K	Needs BASIC-80 4.51
PEARL SD	3.01	8080	56K	w/It Works run time pkg.
PLAN80 Financial Package (Z80/8080)	2.3	8080/Z80	56K	w/CBASIC2, ULTRASORT II
PLAN80 Demo	1.2	8080		Specify Z80/8080
PL/I-80	1.3	8080	48K	
PLINK I Linking Loader	3.28	Z80	24K	
PLINK-II Linking Loader	1.14	Z80	48K	
PMATE	3.02	8080	32K	
PMATE-PC	1.04	8088		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/Z80	56K	Needs BASIC-80
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

VERSION LIST

Product	S	P	MR	
Property Manager	1.0	8080	48K	Needs CBASIC
PSORT	1.3	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	
RM/COBOL Compiler				w/Cybernetics CP/M-80 2, OASIS, UNIX
RAID	5.0.2	8080	28K	
RAID w/FPP	5.0.2	8080	40K	
RECLAIM Disk Verification Program	2.1	8080	16K	
SBASIC	5.4a	8080	48K	
Scribble	1.3	8080		Needs CBASIC
SELECTOR-III-C2 Data Manager	3.24	8080	48K	Needs CBASIC
SELECTOR-IV	2.17	8080	52K	
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/Z80	32K	Needs Word Processing Program
Standard Tax	1.0	8080	48K	Needs BASIC-80 4.51
STATPAK	1.2	8080		Needs BASIC-80 4.2 or above
STIFF UPPER LISP	2.8	8080	48K	
STRING BIT FORTRAN Routines	1.02	8080		
STRING/80 bit FORTRAN Routines	1.22	8080		
STRING/80 bit Source	1.22	8080		
SUPER SORT I Sort Package	1.5	8080		Max. record=4096 bytes
SELECT		8080/Z80	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		
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/Z80		24x80 CRT
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	48K	
Wiremaster	3.12	Z80		Needs 180K/drive
Wordindex	3.0	8080	48K	Needs WordStar
Wordmaster	1.07A	8080	40K	
WordStar	3.0	8080	48K	
WordStar w/MailMerge	3.0	8080	48K	
WordStar Customization Notes	3.0	8080		
XASM-05 Cross Assembler	1.05	8080	48K	
XASM-09 Cross Assembler	1.07	8080	48K	
XASM-51 Cross Assembler	1.09	8080	48K	
XASM-F8 Cross Assembler	1.04	8080	48K	
XASM-400 Cross Assembler	1.03	8080	48K	
XASM-18 Cross Assembler	1.41	8080		
XASM-48 Cross Assembler	1.62	8080		
XASM-65 Cross Assembler	1.97	8080		
XASM-68 Cross Assembler	2.00	8080		
XYBASIC Extended Interpreter	2.11	8080		With EDIT features
XYBASIC Extended Disk Interpreter	2.11	8080		Requires the XYBASIC w/EDIT features to create SOURCE
XYBASIC Extended Compiler	2.0	8080		
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		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.

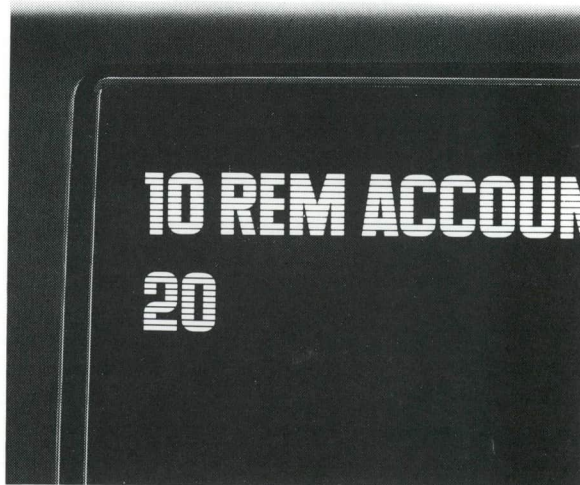
CREATE new databases in minutes, or JOIN databases that already exist.

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.

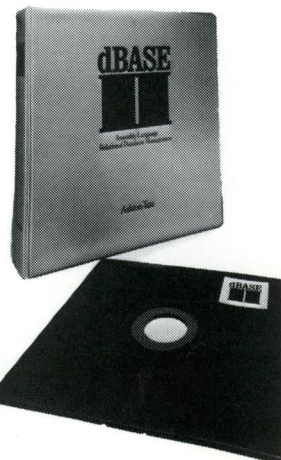
Call for our Dealer Plan and OEM run-time package prices, then take us up on our money-back guarantee. Send us your check and we'll send you a copy of dBASE II that you can exercise on your CP/M® system any way you want for 30 days.

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

Also available from Lifeboat Associates.

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

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



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