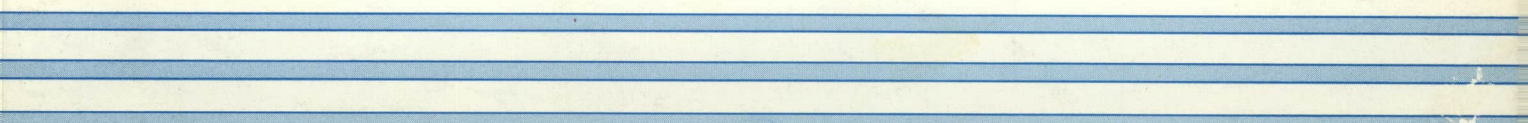
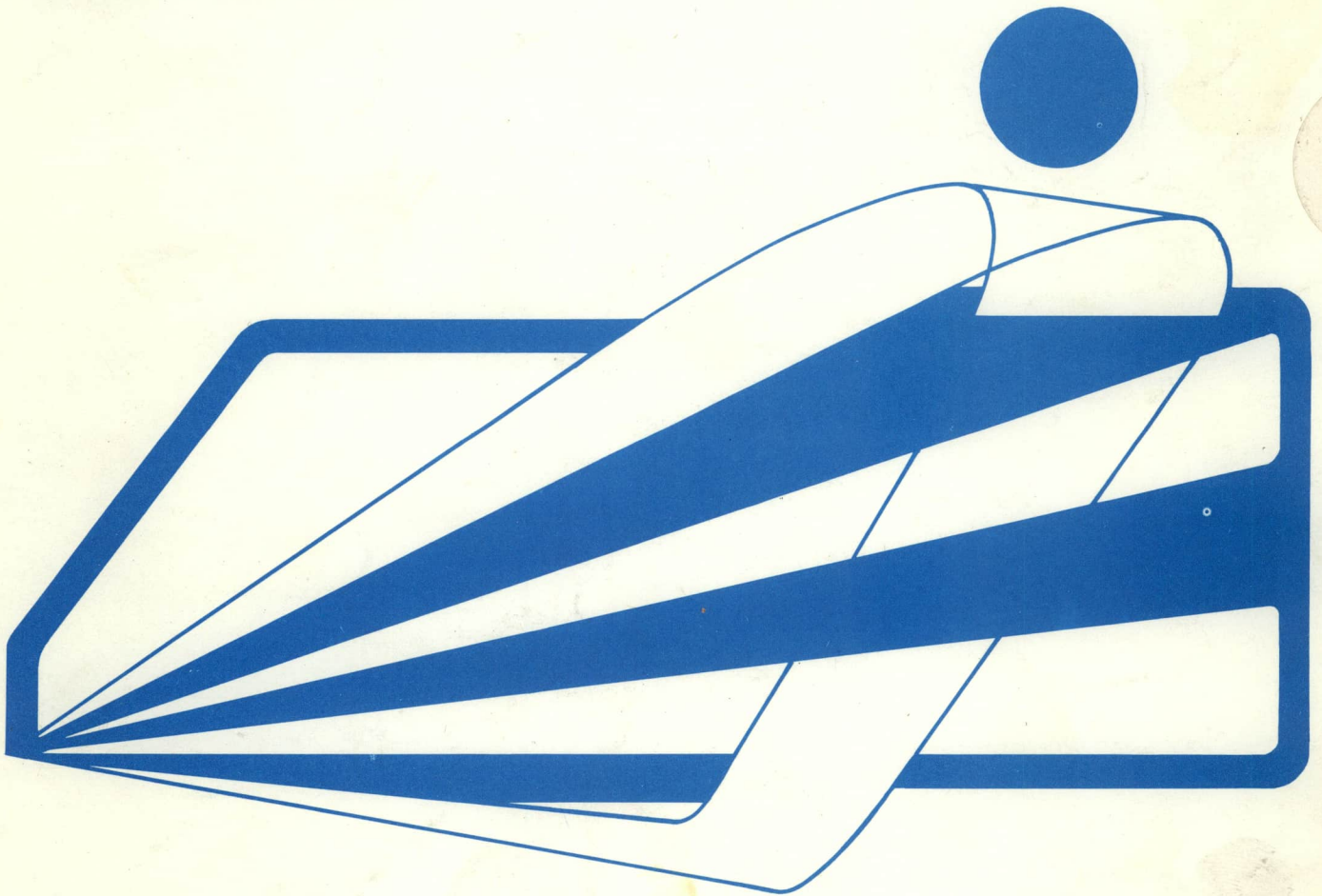


APRIL 1981

LIFELINES[®]

VOLUME ONE No. 11

The Software Evaluation Group: Condor
BASIC Comparisons: SBASIC, Part 2
A Review of PMATE



LIFELINES®

Volume 1 No. 11 April

Editor-in-Chief: Harris Landgarten

Managing Editor: Jane Mellin

CONTENTS		PAGE
Opinion	Editorial Comments	2
	Response	2
Features	A Review of PMATE by Harris Landgarten	3
	The Software Evaluation Group: Condor by Ed Paulette and Steve Patchen	6
	BASIC Comparisons, SBASIC Version 5.3 Part 2 by Bill Burton	13
The CP/M Users Group	CPMUG News	5
Product Status	New Products	16
	New Versions	17
	Bugs and Bug Fixes	18
	Version List	20
Miscellaneous	Tips and Techniques	15
	Readme	12
	Coming	18

* SBASIC is a trademark of Topaz Programming

PMATE is a trademark of Phoenix Associates, Ltd.

* CP/M is a trademark of Digital Research, Inc. The CP/M Users Group is

* not affiliated with with Digital Research, Inc.

Copyright © 1981 Lifelines Publishing Corporation

Lifelines, Volume 1, Number Eleven. Published monthly. The single copy price is \$2.50 domestically, including the U.S., Canada, and Mexico. The single issue price for copies sent to all other countries is \$3.60. A one year's (12 issues) subscription is priced at \$18.00, when destined for the U.S., Canada or Mexico, \$40 when destined for any other country. 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 prepaid. Lifelines is published by Lifelines Publishing Corp., 1651 Third Ave., New York, N.Y. 10028, Telephone: 212-722-1700. Please send all correspondence to the publisher at the above address. Postmaster, send change of address to the above address. Application to mail at 2nd Class postage rates is pending at New York, N.Y.

Editorial Comments

Every so often, I like to examine the current status of microcomputer software and report any trends that I can see developing. Most striking is the dramatic increase in the number of high quality programs entering the market. I think this is attributable in large part to the migration of professional programmers into the field of microcomputers. Evidence of this trend can be seen in the big computer facsimiles that have been appearing lately. In some cases subsets of traditional mini-computer products, especially in the area of database managers, are being introduced. In other cases, the micro versions are superior in both features and speed. A perfect example of this is PMATE, a new text editor which is a fully video version of TECO, an editor familiar to users of DEC equipment. This superiority is no surprise, because single user micros are more powerful than multi-tasking minis from the standpoint of end user performance. The movement of professionals into microcomputer software development is probably due to the potentially large numbers of sales that can be made to a software hungry public.

While assembly language still reigns as the predominant medium for writing low level applications, there is an emerging tendency for programmers to use 'C' or Pascal. Unfortunately, when compared to assembly language substitutes, such programs can only perform on state of the art equipment without noticeable speed degradation. Besides the obvious advantages they offer in programming and maintenance ease, these languages are being used because of their upward source code compatibility to 16 bit machines running UNIX. This trend should build and strengthen as sixteen bit micros grow closer to becoming accepted. Transportable source code applications may very well be the way in which the "chicken-egg" situation so far hindering the progress of the 16 bit evolution will be resolved. Sixteen bit micros should cause an explosion

of features in today's programs. Everyone will have to at least partially rewrite his application, and if there is any truth to the adage, "If I were starting this project over now I'd do things much differently", we will benefit from the result. This in combination with the vastly increased facilities should have a marked effect. Certainly, 8 bit machines will be with us for several more years, so there is little chance of Z80/8080 assembly language programs becoming obsolete in the near future.

One of the nicest side effects of the current UNIX interest is the exposure of the 'C' programming language to the microcomputer public. 'C' is a wonderful language whose limited popularity is due mostly to a lack of publicity compared to Pascal. Even partial UNIX acceptance should change that and give 'C' the chance it deserves. Any qualified enterprising author should grasp this opportunity to write a 'C' tutorial. At this time, other than the Kernigan and Ritchie reference book and internal Bell journals, nothing of substance has been written about 'C'. This language is too important to be ignored.

Harris Landgarten

Response

March 5, 1981

Dear Mr. Landgarten,
I would like to offer a few comments on some issues being discussed in the software industry, which have surfaced recently in "Lifelines."

First, two 'atta-boys' for your editorial in the March issue. I understand the dilemma of the producers of such software as DR's PL/I and Microsoft's BASIC compiler, but the other side of the coin is that we independent software producers absolutely will not allow Digital Research, Microsoft, et al, to be permanent partners in our business. Hence, excellent products, such as PL/I, will not achieve the use (and the profit to DR) that they deserve.

Second, as a software producer, I

understand very well the economic problems of software that can be copied. Software piracy by unscrupulous dealers is certainly a problem, and should be dealt with as severely as possible. However, with the present sophistication (and prices) of good software, I simply do not believe that hobbyist copying and swapping is a major problem. In most cases, the hobbyist is not going to pay five hundred dollars for a piece of business oriented software. If he and a friend split the purchase, the net result to the software producer is one sale instead of none, rather than one sale instead of two. This may well be unethical, immoral, unlawful, etc., but it is a practical fact of life.

Meanwhile, the legitimate end user is severely penalized in several ways if the disk is made uncopyable.

1) No backup. Providing two disks is not an answer, nor is an exchange service to replace the original. We all know that in a practical working environment the original is put away once a master copy is made, then working copies are made (and perhaps modified) from the master copy. We also all know, from hard experience, that to do otherwise is to court disaster.

2) If a disk cannot be copied, there is no way to arrange programs as desired, resulting in a great deal of disk swapping to accomplish anything. For example, I would not consider buying an editor or text processor that I could not put on my system disk so as to have it conveniently available. If the system has a hard disk, the situation becomes even sillier! The retail dealer also has a problem in putting together a good smooth package for his customer if he cannot arrange programs from various sources on a disk or disks as desired.

I really think that we should all consider very carefully the ramifications of some of these so-called 'solutions' to admittedly real problems, lest we do a great deal of damage to our infant industry.

David Goodman
President, Piedmont Data Sciences

A Review of PMATE

PMATE is a new text editor that was recently introduced by Phoenix Software Associates for use on 8080/Z80 based microcomputers with CP/M. It is available from Lifeboat Associates for \$195.00. My first reaction was a predictable "who needs another editor", an attitude that was about to abruptly change.

The first thing that strikes you after firing up PMATE is the presence of two cursors. The "command line cursor", represented by the ' ' character, is always positioned on the second line and marks the place where characters will be entered when the editor is in command mode. The regular terminal cursor functions as the "video cursor", and is allowed to move freely about the document being edited, which is constantly displayed on the screen. Characters can be typed directly into the document by entering either "overtyping" or "insert" mode. No matter what mode you are in however, the video screen can always be manipulated by using "instant commands" which are activated by sequences of control codes. There are key sequences to move the cursor by character, word, line or page in either direction; delete characters, words or lines; to insert lines; to move text; to recover items that have been previously deleted; to shift default case (for non-typewriter style keyboards); to redraw and reformat display; to enter either of the three available modes; to edit the command string. The last command mentioned is most useful. Entry of a 'control _' causes the command line to jump into the video area so that it may be edited like ordinary text. After a command string has been corrected, it can be returned to the command line area for execution, thereby eliminating the retyping of long commands. The usefulness of this feature will become apparent after PMATE's command set has been explained. If you move the cursor past the right edge of the screen, the display scrolls smoothly to the left, acting like

a true window for documents exceeding 80 columns. This is the first microcomputer editor I have used which implements horizontal scrolling; a feature which I predict will be added to many existing text processing programs in the near future.

PMATE divides available memory into 11 buffers, a garbage stack, and a permanent macro area. The buffers are labeled "T" for the regular text buffer and 0-9 for the auxiliary buffers. The garbage stack holds deleted items, (characters, lines or text blocks) in a first in last out organization, discarding the oldest items when memory is exhausted. The instant command 'control R' pops the top of the garbage stack into the current video buffer, thus restoring the last item deleted. This is used to correct accidental erasures and to perform simple text moves. More complex text movement utilizes buffer 0, which is also called the special buffer. The 'control T' instant command 'Tags' the current cursor position. A subsequent 'control E' command will cause all text between the current cursor position and the 'Tagged' position to be moved into the special buffer. A 'control Z' will then retrieve the special buffer and deposit its contents in the document. The marked text can be used repeatedly, since the special buffer is only copied, not moved.

More sophisticated editing can be accomplished in the command mode. PMATE command strings vary in complexity from simple cursor movement or string searches to intricate program-like command strings containing logical tests and iterative sequences. All command strings are terminated with two 'esc' characters which echo as dollar signs. After a command has been completed, it may be executed again by merely hitting another 'esc'. PMATE's command set makes up a mini programming language. Along with the usual cursor movement and delete commands, are a variety of sequence control, I/O, file control, and

buffer control commands, as well as 10 numeric variables, a user number stack, numeric expressions, and constants which provide access to all pertinent information about the current state of the editor. These facilities give the user the power to add commands to the editor by writing appropriate macro programs and then storing them in the permanent macro library buffer. The command Cblah\$blew\$ will change all occurrences of "blah" in the text buffer to "blew". It is often desirable to search and replace interactively, stopping after each occurrence of the search string and replacing it only if a positive response is given. A macro to accomplish this task follows:

```
2QA
[
  S^AA$
  GType escape to replace$
  @K=27[-C^AA$^AB]
]
```

The first line sets the number of string arguments required from the calling command to two. The next line searches for the first argument. The 'G' command then gives a prompt, displays the text buffer with the cursor pointing past the next occurrence of the found string, and waits for a response. Since '@K' evaluates to the ASCII value of the key hit following a 'G' command, the '@K=27' condition will only be true if the user responds with an escape, thereby causing the bracketed replacement command to execute. The entire process repeats until either the string is not found or the user hits 'control C' which will abort any runaway macro. If this macro were labeled 'R' and stored as a permanent macro it would be invoked by the command string 'Rblah\$blew\$'. Macros which are less general in nature can be developed in one of the auxiliary buffers and executed directly from there with a command of the form 'b' where b is the appropriate buffer number. A 'Trace and Break Point' feature is included for debugging formidable macro programs. The manual states that the materials are here even

to even construct a monitor program, a boast which I don't doubt.

Another command feature which I liked was the use of the '#' character. When used in place of a numeric command argument, '#' evaluates to the number of items (either characters, words, or lines) between the 'Tagged' position and the current video cursor. In this way large blocks of text can be deleted or moved by simply 'Tagging' one end, moving the video cursor to the other end, and issuing a command with '#' as the numeric argument (e.g. #B2M to move the marked block to buffer 2).

A large assortment of file commands are included in PMATE. Paging large files through the 'T' buffer, file reading and writing, and directory maintenance are all provided for. In addition, the 'XD' command will write a duplicate copy of PMATE, in its current state, to disk, allowing the user to set custom defaults or add permanent macros and then create a new editor with those features.

Those with a real yen for customization will appreciate that almost all of PMATE's features are user redefinable. All instant commands can be easily changed to match the editor you're familiar with. An instant command sequence can contain any number of ASCII codes, allowing the use of terminal function keys which produce multiple characters. Many other editor characteristics are also user selectable. The exact effect of cursor movement commands can be selected on a per mode basis. For example, a cursor down command in overtype mode may move the cursor straight down, while the same command in insert mode may move the cursor to the beginning of the next line. Screen updates can proceed either from top down or from the current cursor position out. The number of lines the cursor is allowed to wander from the center of the screen can be set, as can the number of lines scrolled in a multiple line move command. Those with a basic knowledge of as-

sembly language can implement new instant commands by assigning keystroke sequences to automatically execute stored user macros. Keystrokes can be redefined to mimic others so that inconveniently located keys on your keyboard can be logically moved. A hook is provided for you to insert a command sequence which is executed before command mode is entered. This feature allows you to implement an auto-updating status line or to define a special purpose text processor which runs without user intervention. The latter possibility could be used for a data conversion utility or perhaps to convert between different internal editor formats. Clearly, your imagination and ingenuity are the limiting factors. PMATE can almost be thought of as a special purpose language with which to write the editor you want.

A limited text formatting feature is provided along with the ability to output directly from the screen to a printer. I found the text formatting to be of little use. The printer control is handy for getting quick listings of program segments. To use PMATE for any but the most primitive of word processing tasks would be inconvenient or impossible. PMATE is definitely at its best when working on programs or data files.

Comprehensive documentation is provided by the manual. It begins with a thorough tutorial, continues with a discussion of macro programming followed by a section on customization, and concludes with a summary of PMATE's commands. Many detailed examples of macros are provided. In addition, the section about customization goes into sufficient detail for the moderately experienced computer user to follow without confusion. While no index or table of contents is provided, the command summary should suffice for most queries by semi-experienced users. A quick reference card would also be a useful addition since no on-screen help system is provided.

PMATE will work with any con-

ceivable serial terminal or memory mapped video display. Facilities for handling all exceptional terminals I have ever seen are provided. The flexible way in which the installer may intersperse delays among the cursor commands should serve as a model to others writing terminal dependent software.

I found only two shortcomings in this almost-perfect editor. The first is that the user must manually control the flow of a large file though the editor by using various file in and out commands as memory becomes limited. Once a section has been written to the output file, it cannot be retrieved until the entire file has been flushed through and the edit is reinitiated. This can be a real inconvenience when dealing with very large programs or text files. I am told that the next revision of PMATE, due in three or four months, will have full bidirectional disk to memory automatic scrolling. In the meantime, although PMATE will run in less than 32K, large memory machines will allow the processing of bigger files without restrictive file manipulations. The other shortcoming (some may call it a feature), which is apparently common to all of Phoenix's editors, is the fact that PMATE zaps all of the NULLS in the file it is editing. Although most program sources and ASCII data files do not contain NULLS, there is one notable exception. Microsoft BASIC source files indicate a physical new line that is not the end of a statement with the sequence LF, CR, NULL. Therefore, I do not recommend the use of PMATE on Microsoft BASIC files.

The more difficult the editing task and the more skilled the user, the more valuable PMATE becomes. I have found PMATE to be especially convenient for writing programs in structured languages such as C, Pascal and PL/I. I can develop the main program in the primary buffer while writing procedures in the auxiliary buffers. If I want to refer to another piece of code, it can be read into a spare buffer, inspected, dissected, and moved or discarded

without affecting any of the other text being edited, as long as memory permits. Furthermore, the ability to define default tabs of three or four spaces, and the horizontal scrolling feature, help in managing programs with many levels of indentation. The hours saved on difficult editing jobs (such as the conversion of a program to a different implementation of the same language) have made PMATE one of my favorite software tools. If your editing jobs are anything like mine, I think PMATE will become one of your favorites too.

Harris Landgarten

CPMUG News

=====

VOLUME 49 IS AVAILABLE

=====

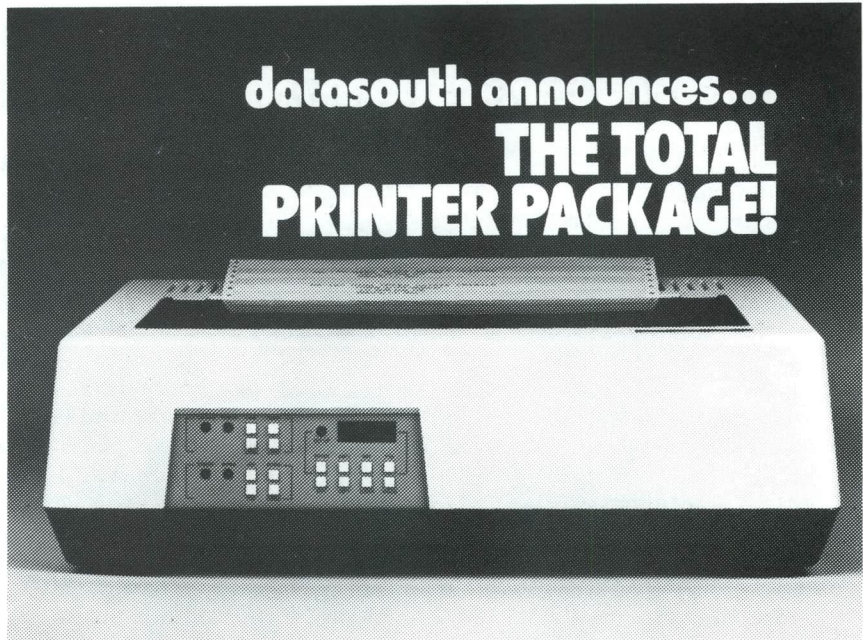
See next month for more detail. This month, we have an exciting new feature The CPMUG BUG prompted by a letter from Rich Leary of Phoenixville, Pa. re "DI" on Volume 40. Rich writes that:

Apparently, the ONLY means of exiting the directory read loop is by detection of an 0E5H in both the 1st and 2nd bytes of a directory entry. Sam Singer's comments state that he reads "groups" 0 and 1; in fact, he sets "group" to 0 and then increments "sector" to read successive directory records. NO test seems to have been made to ensure that only 64 directory entries (ie: 16 records or 2 groups) are read. If the directory is or has been full on a diskette, the program will not exit the read loop (this may be why Sam says that the output is sometimes "scrambled").

I cannot re-assemble Sam's program as I do not have MAC. I did verify the changes listed below by inserting a patch and testing them. They allowed me to use the program with disks that previously could not be used.

Here are the changes:

```
DIR6:  LHL  OUTB
        XCHG
        LHL  INB
        MVI  A,0E5H
        CMP  M
```



datasouth announces... THE TOTAL PRINTER PACKAGE!

The DS180 matrix printer provides the total package of performance features and reliability required for applications such as CRT slave copy, remote terminal networks and small to mid-range systems. Not a "hobby-grade" printer, the DS180 is a real workhorse designed to handle your most demanding printer requirements. And pricing on the DS180 is hundreds of dollars below competitive units.

High Speed Printing—Bidirectional, logic-seeking printing at 180 cps offers throughput of over 200 lpm on average text. A 9-wire printhead life-tested at 650 million characters generates a 9x7 matrix with true lower case descenders and underlining.

Non-volatile Format Retention—a unique programming keypad featuring a non-volatile memory allows the user to configure the DS180 for virtually any application. Top of form, horizontal and vertical tabs, perforation skip-over, communications parameters and many other features may be programmed and stored from the keypad. When your system is powered down, the format is retained in memory. The DS180 even remembers the

line where you stopped printing. There is no need to reset the top of form, margins, baud rate, etc...it's all stored in the memory. If you need to reconfigure for another application, simply load a new format into the memory.

Communications Versatility—The DS180 offers three interfaces including RS232, current loop and 8-bit parallel. Baud rates from 110-9600 may be selected. A 1K buffer and X-on, X-off handshaking ensure optimum throughput.

Forms Handling Flexibility—Adjustable tractors accommodate forms from 3'-15". The adjustable head can print 6-part forms crisply and clearly making the DS180 ideal for printing multipart invoices and shipping documents. Forms can be fed from the front or the bottom. If you would like more information on how the DS180's low-cost total printer package can fill your application, give us a call at Datasouth. The DS180 is available for 30-day delivery from our sales/service distributors throughout the U.S.

datasouth computer corporation

4740 Dwight Evans Road • Charlotte, North Carolina 28210 • 704/523-8500

```
>>>> Delete the following lines:
        JNZ  DIR8
        INX  H
        CMP  M
        JZ   SORT
```

```
>>>> and change the JMP to a JZ:
        JMP  DIR12
```

The last instruction in the DIR12 routine is JMP DIR4 to read another block from the directory.

Delete this JMP and insert:

```
CPI    8           ;see if 8
JNZ    DIR4       ;loop if not
XRA    A          ;clear A
STA    S          ;and then sector
LDA    G          ;get group
INR    A          ;bump it
STA    G          ;save it
```

```
CPI    2           ;see if too big
JNZ    DIR4       ;loop if not
JMP    SORT       ;else done
Note: you only have 3 bytes (the
JMP instruction) free at this
location (DIR12) if you are going
to hot patch the code. You
should put a JMP INSERT at that
point (in place of JMP SORT) and
use the DDT "A" command to put
the INSERT routine at the end.
```

I would like to thank Rich for sending in this patch. Sam Singer's program is quite useful and I hope you can use this data. We welcome your contributions. Please send them to CPMUG, 1651 3rd Ave., NYC NY 10028.

THE SOFTWARE EVALUATION GROUP: CONDOR

by ED PAULETTE & STEVE PATCHEN

Name of database package:
CONDOR SERIES 20/DBMS LEVEL I & II
Author: CONDOR COMPUTER CORPORATION
3989 RESEARCH PARK DR.
ANN ARBOR, MICHIGAN 48104

Series 20 from Condor Computer Corporation is currently available in two editions, Level I and Level II. Both Level I and Level II utilize the "Relational Model" - the same one used in IBM's recently released SQL system (known before release as SYSTEM R). This is the first system we have evaluated which actually follows a previously described model. One of the major advantages of this system is that it has a consistent underlying conceptual structure. Of the three commonly described models for database systems the Relational is the simplest and at the same time probably the most elegant. In database circles it is usually discussed as potentially the most "user friendly".

While both Condor database systems follow the relational model, only Level II can be said to be "fully" relational. This is almost academic nit-picking, since the less expensive Level I includes a very well-chosen set of commands which allow the implementation of almost any necessary set of many-to-many relationships among files.

It's not possible to describe the nature of the relational data model in any significant detail in this review, so I encourage consultation of James Martin's (almost classic) introductory text on database systems, Principles of Database Management, for more information. C.J. Date's Introduction to Database Management Systems goes into more detail on relational systems. Both are referenced in the second article in this series.

Using the relational data model, data is viewed by a user as falling into two dimensional tables in which the columns correspond to fields and each row corresponds to a separate record in conventional systems. Each of these tables is called a "relation", because it defines the relationship between the different data values within a record. In most writing about relational systems, a record is termed a "tuple" and each column is called an "attribute". These terms are part of the elegant (but unnecessary) mathematics underlying the system, "tuple"="row"="record" and "attribute"="column"="field".

Under Series 20 a "relation" is called a "dataset". A collection of one or more datasets logically describing your data structure would be called a "database".

If all we could do was implement and operate on a single datafile at a time, we would have nothing exceptional (although this is the extent of the

capabilities of some of the least expensive "database management system" packages). A relational system goes well beyond this, allowing one to operate on one or more relations to produce entirely new relations!

A trivial example of producing a new relation is the SELECTION of a subset of the records in a dataset, producing a new relation which contains only those records meeting specified criteria. Less trivially, one can specify that only certain fields be included in a result dataset (the PROJECT operation), and further, that this newly-created dataset be "JOINED" with another dataset, using some arbitrarily chosen set of matching fields (called "KEYS"). And all of this can be done interactively, on the fly, without having specified that any of those operations would have to be performed before they were actually needed. (In Series 20 matching fields must have the same name. This limitation may be removed in the future.)

The ability to directly, interactively specify such powerful operations after the entire database has already been defined, makes a relational database system very attractive for many systems where a more rigid database is impractical. For new applications, for ill-defined applications, for applications where the database structure is volatile or where the demands of users for information change rapidly, the relational model can make the difference between success and failure. Because each relation (dataset) is independently specified, it's possible to implement even a very complex database a little at a time, so it's also modular.

Focusing now on the particulars of CONDOR's implementation of the relational model:

Installation goes quite quickly - if your terminal is compatible. One of the authors has used the package on a TRS-80 Model II with LB CP/M, and an ALTOS ACS8000-5 multi-user system with an IBM 3101 under both CP/M 2.2 and MP/M 1.12. (Only Level II works under MP/M.) In each case, the system installed quickly and worked smoothly.

After loading the database command processor (which remains resident thereafter) and identifying your terminal type to the system, one can begin definition of a new dataset by simply issuing the "FORMAT" command. One then "draws" the desired data entry screen, using control keys and typing the text one wants to appear during data entry. Field names are specified by typing them on the screen surrounded by brackets, "[" and "]".

One is next automatically prompted for definitions of the fields named on the screen. Extensive defaults are provided. (For example, the system will count the number of dashes typed on the data entry screen next to a given field name.) Maximum, mini-

mum and default values can be specified to provide minimal data validation.

As soon as this step is complete, one can immediately enter data to the dataset, retrieve the data and relate it to other already existing data. For seven or eight fields, the entire process can easily take less than fifteen minutes.

Level I and Level II are completely compatible; databases established under one of the two can be read and accessed by the other (within the capabilities of each). Level I includes all the basic utilities and operations which allow one to define and modify the definition of datasets: to enter, display, modify and sort data in the datasets; to select subsets of datasets; to compare two datasets ("RESTRICT" in relational terminology) and isolate the matching or non-matching records; to list the data and to read and write the data from external files. To these very extensive capabilities Level II adds the powerful JOIN and PROJECT operations as well as several very useful utilities. The Level II INDEX command allows the selection and sorting of indices to data records without actually writing out a new copy of the selected records. Subsets selected using the INDEX facility can be accessed by the other commands in Series 20.

One small departure in the Condor package from the standard relational features is that Condor does not currently support weeding, the elimination of duplicate tuples. This feature is most important when a relation has been created by a series of joins and projects, so that it contains many duplicate entries. As with many of the small problems we noted in this package, Condor has told us they will correct this one with a weeding option (in conjunction with the sort function). This feature will be an update to both Levels I and II.

Level II also adds compatibility with MP/M and the ability to automatically initiate a CP/M submit file upon leaving the Database system, allowing access to non-DBMS programs requiring the entire TPA and CCP address space.

With minor differences, Level I and Level II appear almost to be powerful extensions to CP/M, MP/M or CDOS. CP/M utilities can be executed directly, as can most user written programs which don't depend on specific locations in CP/M or access to more than 32K of user memory. (If more is needed both Level I and Level II can be "unhooked" then reloaded from a submit file.) When a series of commands is to be regularly executed, those commands can be stored in a CP/M textfile and the usual CP/M facility used to execute them. Level II allows interrupted batch files to be restarted (cautiously!) in case of power failures or other catastrophes.

A general ledger comes with the system as an example application and is used in the manual to familiarize the user with the various operations in the system. As a test application, we developed a check

entry system in addition to this ledger. It is one of the evaluation problems, and one of the authors desperately needed an automated bookkeeping system with a checking account balancing act.

The other problems have been successfully implemented for production databases exceeding 600Kb (for one of the authors' clients), but they will not be discussed here except to note that they can be performed conveniently with this system.

Series 20 is the first true database system that we have reviewed so far. Its price reflects this stature, however. It is not without problems, unfortunately. First, we will discuss the system and the problem setup and then we will point out some of the good features and some of the inconvenient ones in using this system.

The accompanying illustrations are examples of the printed form of the basic parts of the Condor Database System. A menu selection feature is available (see Figure 1), but all commands can be entered interactively. The menu can execute any valid CP/M command but its most common use is to execute the SUBMIT command with a batch processing file. The batch files are, in a minimal sense, programs. They can be created with any text editor and given a name with a ".SUB" extension. The menu itself is likewise created with a text editor and given an extension of ".HLP". It is loaded with the command "HELP name". The part of each line in brackets is not displayed on the screen but is executed as a CP/M command if that line is selected. In my example I have changed the SUBMIT to DO to save typing in interactive mode.

Figure 2 is a listing of the data dictionary in tabular format. It can also be printed on the individual screen forms used to enter each dataset into the data dictionary. The last four lines are the datasets added to implement the checks entry system for the ledger.

Figures 3 and 4 are the screen forms for the checks entry datasets. They are followed in Figures 5 and 6 by the corresponding data definitions. The field numbers on the forms are found by scanning from left to right and from top to bottom. Enough underlines must be supplied for each entry on the form to accept the number of characters required for the "MAX" entry specified for each field. The data types are "A" for alpha characters, "N" for integer numbers, "AN" for alpha or numeric, "DATE" for dates entered in the "mm/dd/yy" format and stored as a Julian date starting from 1/1/00. The "MIN" & "MAX" ignore the sign of the number when defining the number of bytes required to store it, but the ranges specified are enforced during data entry.

The last illustration, Figure 7, is the batch submit file for the checks entry. As we discuss the strategy for the checks entry we will also describe some of the commands of the Condor system. Each line in the batch file is a valid CP/M command line. The first word on the line is a DBMS command

which is loaded and executed as an independent program. If the DBMS operating system has not been loaded already, the command terminates with an error message telling the user of the oversight. The start-up procedure can also be submitted as a batch. The DBMS operating system loads below the CCP for a 48K system; it does not load any higher in memory if a larger CP/M system is used.

Description of the Checkbook problem:

After the checks and deposits are entered using the data entry routine "ENTER", they are sorted by the ledger balance account numbers. The result is compared to the general ledger to produce a list of entries to non-existent accounts. The "TITLE" command sends a form feed to the printer and prints the specified title at the top of every page until a different title is specified. The check entries are compared to the bank account balance records to insure the checks and deposits are being made to existing bank accounts. This result is also printed and moved to a temporary file. The comparison is made again to select the good entries. The temporary file is also compared to the ledger to select entries with good account numbers. The result again is moved to the temporary file and is appended to the audit file. The result is printed and posted to the ledger.

Next the debit and credit fields in the balance records are zeroed and the temporary file is posted to the bank accounts. In order to post the bank accounts to the ledger the credit and debit fields have to be exchanged so they will balance with the previously posted individual entries. The debit and credit are moved to the checks and deposits fields, and the checkbook balance is updated; then the debit and credit are exchanged, while reloading, to the debit and credit fields. Note the unnecessary "+0"s required by the syntax of the system. After the bank accounts are posted, the year to date amounts in the ledger are updated. In order to update the balance proof the unprocessed checks are selected from the check audit file and posted to the debit and credit fields. Then useful information is printed.

General Comments:

You will notice that the checks entry procedure follows a very "straight line" strategy. As those of you who use CP/M's SUBMIT facility are well aware, no mechanism is provided to change the flow of control within the SUBMIT file. In some situations this is a considerable inconvenience. Given the non-procedural nature of the command language provided by Series 20, this is not such an inconvenience as it might otherwise have been. In the current example, however, note that there is no method for determining whether errors in the input data were corrected; nor is it possible to stop the procedure in order to allow the operator to make corrections. You must either provide an error correction step or require that the operator watch the processing and attempt to halt the process in midstream by holding down CONTROL-C until the batch

is interrupted. (Control-C will not interrupt the batch under many implementations of CP/M.)

If a method were provided to check a dataset for the presence or absence of data records and conditionally to skip the next SUBMIT file command or terminate batch processing, one could implement it much more intelligently. Since the SUBMIT facility is a well-known CP/M feature and Series 20 commands are .COM files, implementation of this facility might be an early contribution by some public-spirited member of a Series 20 users group.

The print function is currently very restrictive, but is due for expansion. The examples in this article illustrate what is currently available as a format. (except for the "STAX" function which prints out a total, an average, the minimum and maximum of a selected field for all records in the dataset.) Promised extensions to the LIST, PRINT, and STAX commands for both Levels I and II will provide selection of records to be printed, the specification of fields to be counted, totaled, averaged, etc and multiple levels of control breaks (e.g. for subtotalling). A Report Writer feature has also been promised for release in the near future as part of Level II, but no details were available regarding its capabilities.

A significant current limitation is the inability of Series 20 to sort files larger than 128K. Condor has promised that the sort capacity will be significantly increased or the problem eliminated in the near future. A new feature available in Level II (but not in the version tested), is the ability to generate indices of records in a manner similar to the generation of a subset by the SELECT command. Since one can sort these indices and use them to access the files for subsequent processing by other commands, the sort file size limitation should be considerably lessened for Level II users. The INDEX command is obviously useful in many contexts.

Interactive examination and updating of information is easy in the Condor system. The database integrity is well maintained and most losses involve only the most recent entry. ("Hard" system crashes due to static or power failures have occasionally caused the loss of all data entered since the file was closed.)

Review Summary

Good points for use of this package:

This is a very flexible package which closely follows the relational model in the capabilities it provides and the view it presents to the user. It has proven quite reliable in production use with databases of substantial size (up to 600Kb). It can handle very complex data base structures in a modular fashion without overwhelming the user. With care it can be used in a multi-user environment (MP/M). The operations it provides have been thoroughly studied by database theorists and several texts for use in addition to the documentation provided by Condor Computer are available de-

scribing general strategies.

It is very easy to learn to use and works very quickly considering its generality and flexibility.

Bad points for use of this package:

Series 20 is among the more costly packages available which puts it out of the price range of most persons personally financing their software habits. As a business package, it suffers most from lack of a report generation package of the caliber of that provided with CBS and Selector IV and from the inability of the SORT command to sort files greater than 128K. (Larger files can be segmented and recombined using the SELECT command.)

The SUBMIT file facility as currently implemented can make certain types of tasks tricky to program. (See comments above.) Data validation during data entry is quite limited (somewhat more so than CBS). It would be very desirable to be able to enforce the uniqueness of primary record keys, and to allow only specific values in certain fields where data is entered in coded form.

Recommendations and potential application:

This system is especially well suited to use by novices and where requirements are likely to vary quickly over time. Because it can handle very complex databases, it should also find utility in situations where flexible access is of paramount importance.

Since it is possible to move easily from the currently available Level I to the coming Level II (both will be supported indefinitely), users who are uncertain can begin with one and shift to the later one should the utility of the additional commands become apparent. Some of the newest features of Level II will make it a very attractive package indeed.

Ed Paulette is currently using Level II in conjunction with a similar relational database management system on a time-sharing system in which data entry, and editing are performed locally and more general processing is performed centrally. The general nature of the relational model allows the implementation of equivalent databases on both machines. Connect time is thus minimized, and additional software development was minimal.

References:

See the introductory articles for details on terminology and the evaluation format used. These articles appear in the following issues of Life-lines¹ Volume 1: No. 4, No. 5, No. 6, and No. 7.

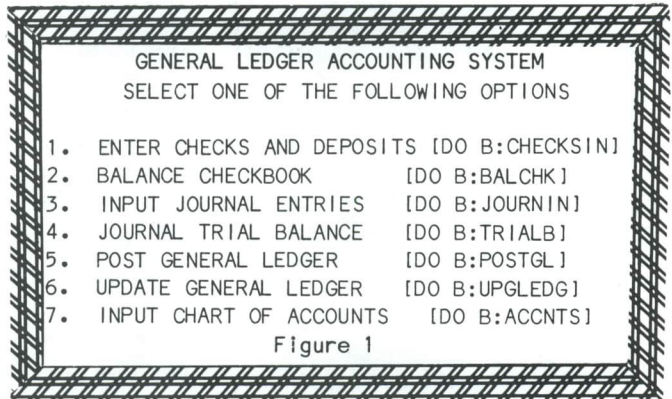
* Neither of these database systems would run on my system without modification. They did not like talking to my Digital Equipment VT52 terminal. The terminal does not have line wrap around. The software does not have carriage returns or line feeds in the screen forms. I was able to get the Level I system working by inserting CRLFs into each form at the ends of the lines in place of the last two

characters. This fix was defeated by the Level II system. Harris and Steve are working on an article dealing with battles over terminal interfacing. We hope to make useful recommendations to both authors and users. (SP)

Editor's note: The authors of Condor have informed me that they are planning to offer the source of the terminal drivers as an option for those with difficult to implement terminals (many terminals will fit this description). This should be at least a partial solution for those with a knowledge of assembly language.

Ed Paulette is Associate Executive Director of The Michigan Evaluation Resource Center, where he is engaged in the implementation and support of financial applications and client data base systems for human service organizations. His address at The Center is 338 South State St., Ann Arbor, MI 48104.

Steve Patchen is a data engineer who provides hardware and software support for small business computer users in the southeastern Michigan area. His address is: LAB Data Systems, 255 Chippewa, Pontiac, MI 48053.



DATA DICTIONARY.

Title	Data	Form	Definition
GLEDGER	GLEDGER	GLEDGER	GLEDGER
GLTEMP	GLTEMP	GLEDGER	GLEDGER
JOURNAL	JOURNAL	JOURNAL	JOURNAL
JOURAUDT	JOURAUDT	JOURNAL	JOURNAL
RESULT	\$.DAT	\$.FRM	\$.DEF
CHECKS	CHECKS	CHECKS	CHECKS
BALANCE	BALANCE	BALANCE	BALANCE
CHKTEMP	CHKTEMP	CHECKS	CHECKS
CHECKAUD	CHECKAUD	CHECKS	CHECKS

GENERAL LEDGER ACCOUNTING SYSTEM
CHECKS AND DEPOSITS

NAME : _____ (Bank
account name)
DATE : _____ (Date check or de-
posit)
CHECKNO : _____ (Check number or
blank for deposits)
DEBIT : _____ (Amt. check or
blank for deposits)
CREDIT : _____ (Amt. deposit)
ACCOUNT : _____ (Ledger account
to debit/credit)
DESCRIPTION : _____
PROCESSED : _____ (Flag to track pro-
cessing, leave blank)
Figure 3

GENERAL LEDGER ACCOUNTING SYSTEM
BALANCE RECORD

ACCOUNT : _____ (Bank ledger
account number)
NAME : _____ (Bank
account name)
MONTH : _____ (Accounting month)
DATE : _____ (Date of last
statement)
STATEMENT : _____ (Statement
balance)
CHECKS : _____
DEPOSITS : _____ (From check-
book)
CREDIT : _____
DEBIT : _____ (Post to ledger)
BALANCE : _____ (Current
checkbook balance)
PROOF : _____ (STATEMENT+CRE-
DIT-DEBIT)
Figure 4

CHECKS DATA ITEM DEFINITIONS:

FIELD	TYPE	SIZE	MIN	MAX
1	AN	20	0	20
2	Date	3	1	36524
3	N	5	0	2147483647
4	\$	4	0	2147483647
5	\$	4	0	2147483647
6	N	4	0	2147483647
7	AN	50	0	50
8	AN	1	0	1

Figure 5

BALANCE SHEET DATA ITEM DEFINITIONS:

FIELD	TYPE	SIZE	MIN	MAX
1	N	4	0	2147483647
2	AN	20	0	20
3	AN	8	0	8
4	Date	3	1	36524
5	\$	4	0	2147483647
6	\$	4	0	2147483647
7	\$	4	0	2147483647
8	\$	4	0	2147483647
9	\$	4	0	2147483647
10	\$	4	0	2147483647
11	\$	4	0	2147483647

Figure 6

CHECKSIN.SUB: THE BATCH PROGRAM TO ENTER CHECKS
INTO THE DATABASE

```

ENTER B:CHECKS
SORT B:CHECKS BY ACCOUNT
COMPARE B:CHECKS B:GLEDGER NOT MATCHING ACCOUNT
TITLE 'CHECKS NOT ENTERED TO EXISTING ACCOUNTS',L,L,'DATE ',DATE,L
PRINT RESULT BY NAME CHECKNO DEBIT CREDIT DATE ACCOUNT
COMPARE B:CHECKS B:BALANCE NOT MATCHING NAME
TITLE 'CHECKS & DEPOSITS TO UNRECOGNIZED BANK ACCOUNTS ',DATE,L,L
PRINT RESULT BY NAME CHECKNO DEBIT CREDIT DATE ACCOUNT
COMPARE B:CHECKS B:BALANCE MATCHING NAME
EMPTY B:CHKTEMP OK
APPEND B:CHKTEMP.DAT $$$$ .DAT
COMPARE B:CHKTEMP B:GLEDGER MATCHING ACCOUNT
TITLE 'CHECKS TO BE POSTED, ON DATE ',DATE,L,L
PRINT RESULT BY CHECKNO DEBIT CREDIT DATE ACCOUNT
EMPTY B:CHKTEMP OK
APPEND B:CHKTEMP.DAT $$$$ .DAT
APPEND B:CHECKAUD.DAT $$$$ .DAT
POST B:GLEDGER RESULT BY ACCOUNT AND ADD DEBIT,CREDIT
COMPUTE B:BALANCE ST DEBIT=0+0
COMPUTE B:BALANCE ST CREDIT=0+0
POST B:BALANCE B:CHKTEMP BY NAME AND ADD DEBIT,CREDIT
COMPUTE B:BALANCE ST CHECKS=DEBIT+0
COMPUTE B:BALANCE ST DEPOSITS=CREDIT+0
COMPUTE B:BALANCE ST BALANCE=BALANCE+DEPOSITS-CHECKS
COMPUTE B:BALANCE ST CREDIT=CHECKS+0
COMPUTE B:BALANCE ST DEBIT=DEPOSITS+0
POST B:GLEDGER B:BALANCE BY ACCOUNT AND ADD DEBIT,CREDIT AND REPLACE MONTH
COMPUTE B:GLEDGER ST YTDAMT=DEBIT-CREDIT
SELECT B:CHECKAUD ST PROCESSED NE Y
COMPUTE B:BALANCE ST CREDIT=0+0
COMPUTE B:BALANCE ST DEBIT=0+0
POST B:BALANCE RESULT BY NAME AND ADD DEBIT,CREDIT
COMPUTE B:BALANCE ST PROOF=STATEMENT+CREDIT-DEBIT
TITLE 'OPEN CHECKS AS OF ',DATE,L,L
PRINT RESULT BY CHECKNO DEBIT CREDIT DATE ACCOUNT
TITLE ',BANK BALANCES AS OF ',DATE,L,L
PRINT B:BALANCE BY NAME ACCOUNT STATEMENT BALANCE PROOF
HELP B:RESTART
    
```



TABLE I
Facts & Figures

Package or Version name:
CONDOR SERIES 20/DBMS LEVEL I & II

Price: Level I	\$695
Level II	\$995
Upgrade I to II	\$350

Systems available for:
CP/M (1.4 and up)
CDOS (2.36)
MP/M (Level II only)

Required supporting software:
Operating system with SUBMIT utility

Hardware requirements:
Z80 cpu
48K user RAM (includes CP/M, excludes MP/M)
2 disk drives with total capacity of 500Kb

Utility programs provided:
NONE

Record size & type limits:
Fixed length, binary used internally,
2 to 1024 bytes, 1 to 127 fields/record,
1 to 127 bytes/field, 1 to 10 digits, up
to 32767 records/file

External record types:
Fixed length ASCII w/o delimiters
Variable length ASCII w/delimiter
(readable by most BASICS)
Fixed length ASCII w/inter-record delimiter

Portability:
GOOD PORTABILITY among CP/M systems
except for some terminal problems. Good
application design portability to other
relational systems.

User skill level required:
A novice could implement useful systems, an
expert could implement very complex ones.

System upgrade policy:
Nominal cost for upgrades within Level
Difference in price plus modest charge for
upgrades from one level to another

TABLE II
Qualitative Factors

Rating*

Documentation **	
organization for learning	5
organization for reference	5
readability	5
includes all needed information	4

Ease of use	
initial start up	5
conversion of external data	7
application implementation	5
operator use	6

Error recovery	
from input error	7
restart from interruption	6
from data media damage	5

Support	
for initial start up	6
for system improvement	6

* Ratings in this table will be in a 1-7 scale
where:
NT = not tested
1 = clearly unacceptable for normal use
4 = good enough to serve for most situations
7 = excellent, powerful, or very easy depending
on the category

** Relational systems are well described in the
references cited in this review. If some sort of
supplementary text is used to gain a broad under-
standing of the system, the documentation should be
very adequate. An attempt is made to provide enough
information to allow novice users to implement
simple databases without having to use any outside
materials. It has been successful in this for at
least one such novice.

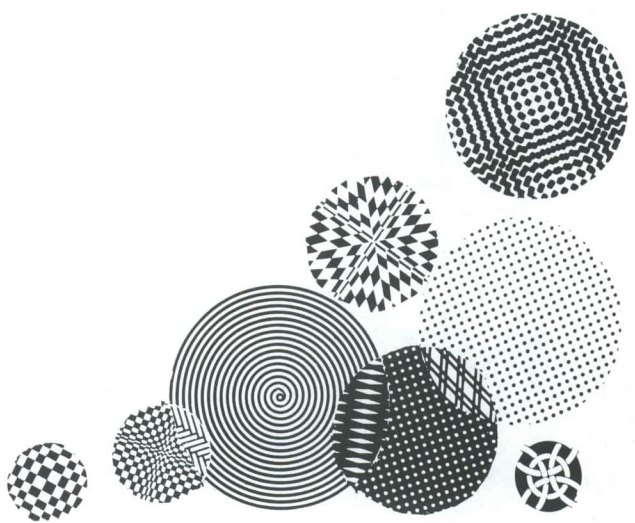


TABLE III
Data Management Capabilities

A. Underlying Data Model

1. Data Types char, integer, dollars, date
2. Relationships full relational n:m

B. Functions Provided

1.a. Data dictionary maintenance:

All functions in Series 20 are performed with the same commands used to input, edit and manipulate ordinary datasets (except the screen formatting and initial file definition). This lends simplicity and elegance to the system which makes it easier to learn. However, the lack of special facilities for dictionary listing and field specification modification are sometimes inconvenient. Dictionary maintenance is more than adequate but could stand some improvement.

b. Data reorganization & conversion:

This is one of the strongest features of Series 20. A special command is provided in Level I (and II, of course) to allow the addition or deletion of fields from existing datasets in a very straight forward manner - data already entered is reformatted, new fields are padded.

A number of formats are provided for reading and writing external data. It should be relatively simple to convert data from or to Series 20 datasets. Formats are provided which are compatible with typical BASIC (both sequential and random), FORTRAN, RPG and COBOL I/O packages, as well as formats appropriate for transmission to record oriented host systems in distributed processing applications.

2.a. Data entry and editing:

Development of screen formats is very convenient allowing "form-oriented" data entry. Multiple "views" can be provided onto the same datafile by definition of alternate screen formats. Data validation is limited during data entry, but batch processes can be used to perform arbitrarily complex validation. Editing of data in the sense of updates and corrections is extensively supported with predicate selection of records to be modified. A user can also page through a file and modify randomly chosen records.

b. Report generation:

Report generation is quite limited at present, promised upgrades will first (and soonest) provide subtotalling capability, and later a more complete report generator. (They've suggested that this will not involve a price increase, but such things have a way of changing.)

TABLE III (continued)

3.a. Data selection by predicate:

And how! Up to 32 terms can be specified in the predicate using the usual set of relational operators. The major current limitation is that only 128 characters may be input as a single command.

b. Data joining & relating multiple data sets:

The JOIN is officially only available in the Level II version, but the POST command which is functionally equivalent to the UPDATE facility in CBS, allows one to achieve the equivalent in a much more round about fashion.

The ability to relate multiple datasets is the sine qua non of the relational model. Condors Level I provides most of these facilities, Level II supplies the rest.

c. Calculations on data:

The COMPUTE command provides the four basic arithmetic functions between fields or between a field and a constant. No precedence is maintained and parentheses do not change the order of operations. This function is sufficient for most business applications, but not striking.

4.a. Data independent application interface:

The READ and Write commands are the only mechanism currently provided. This is barely sufficient and is certainly not data independent. Some mechanism for direct access to datasets is needed which uses the definition file to provide automatic conversion on request. Something like this might be possible using MP/M's interprocess communication c1, but this is beyond the capabilities of most microcomputer users.

readme

Last month our charter subscribers received a letter and a survey--the former to remind them that renewal time is around the corner, the latter to elicit their opinions on Lifelines. It's not too early for those of you who began with us last June to renew your subscriptions. We don't want you to miss a single copy.

When you renew or change your address please enclose one of your current mailing labels, or a photocopy of it. That way we can serve you as promptly and efficiently as possible. We've already lost one or two of you who haven't reported address changes (but then you aren't reading this).

S-BASIC, VERSION 5.3 PART 2-BILL BURTON

Author's note: Part 1 of this product review was mostly devoted to some first impressions of S-BASIC. This installment will deal primarily with the structure oriented features of that language. Again, I would like to thank Gilbert Ohnysty, the author of S-BASIC, for his help and insights.

The term, structured, is rather ambiguous as it describes not only the design of certain languages but also an approach to programming which helps one to create easily readable and maintainable code. This is a most desirable objective especially in the case of large, complex programs. To some extent, a programmer's competence can be gauged by the degree of structure with which he or she writes. Structured techniques are virtually mandated when several programmers collaborate on a single project because each programmer's code must interact properly with that written by others. Under such circumstances, it is common practice for programmers to debug troublesome segments of each other's work. This is a most efficient approach, but one which requires that code produced by members of a programming team all be written in a single, readable style. This style, of course, is structured.

BASIC and FORTRAN are essentially unstructured and the programmer using either of these languages must adhere to special disciplines in order to implement even marginal program structure. One technique for maintaining structure is called top-down programming, which means simply that no backward references are made via jumps or subroutine calls. In fact, some purists insist that proper 'top-down' programming does not even allow backward references as returns from subroutine calls. In other words, each program line executed should be physically closer to the logical end of the program than the previous line executed. As is the case with unstructured programs, there may be one or more lines which are never executed due to variables presented at run time; more than one logical exit from a program may exist. While it is possible to write

clear, structured code in either BASIC or FORTRAN, it is not easy, especially in the case of large or complicated programs.

Structured languages, (often called Block and Procedure oriented languages), such as Pascal, ALGOL, C and PL/I are popular amongst professional programmers as they have been designed to minimize confusing and potentially troublesome logic design. Despite the fact that these languages do offer unconditional jumps equivalent to the GOTO statement in BASIC (or the GO TO of FORTRAN), many good programmers deliberately avoid such jumps entirely. The recently coined term 'GOTOless programming' refers to what some consider the essence of good program structure.

FORTRAN 77 was conceived in large part to improve upon the structural deficiencies of earlier FORTRANS, and it includes many features of the more traditionally structured languages.

CBASIC was the first widely used microcomputer BASIC to support readable, structured programs with features such as long variable names, WHILE/WEND, multi-line user-defined functions, free use of 'white space' and relaxed line numbering requirements. S-BASIC has taken this idea a lot further by incorporating not only these features but also many others which until now were only available in the purely structured languages.

PROCEDURES, FUNCTIONS AND BLOCKS

Figures 1a and 1b represent a simple BASIC program. Figure 1a illustrates this program using the closest approximation of structure available in standard BASIC. Figure 1b demonstrates that same program using the more structured approach of S-BASIC.

Two major differences between examples 1a and 1b are apparent. In S-BASIC, all variables must be declared by name and type (variable declaration was discussed in the last installment); also procedures are used in place of subroutines. 'Fall-through' of program logic

from top to bottom is obvious in 1b. Note also that the main portion of the S-BASIC program exists (or may exist) as a block. Blocks will be explained shortly.

As regards variable declaration, it is not a requirement of standard BASIC. Nonetheless, many programmers are in the habit of declaring variables at the beginning of a BASIC program. This, however, is usually done to order the variable lookup table for increased execution speed rather than for structure.

Procedures differ from subroutines in that they must be declared closer to the beginning of a program than any statement which accesses them, and procedures are accessed by name rather than by line number. The way to declare a procedure is; (elements in square brackets are optional):

```
PROCEDURE <name> [(argument.def
[;argument.def])]
  <body of procedure>
END
```

Figures 2a and 2b, which represent functionally identical programs, illustrate the difference in usage of subroutines and procedures.

User defined functions are declared as follows:

```
FUNCTION <name> [(argument [, ar-
gument])]=<type>
  <body of function>
END=<expression>
```

In the above example, <expression> is the value which will be returned when the function is called.

A block consists of any number of program statements which appear between matching BEGIN and END statements. Blocks logically correspond to a single program statement. A block structure might appear as:

```
REM LINE NUMBERS FOR
REFERENCE ONLY
1: BEGIN
2: [declare variables]
3: <some BASIC statements>
4: [[BEGIN
```



```

5:  [declare variables]
6:  <some BASIC statements>
7:  END]
8:  <more BASIC statements>]
9:  END

```

In this example, an optionally nested block is represented by lines 4-7 and line 8 is meaningful only if the optional block exists. Blocks may be nested to any level, with memory being the only limiting factor.

GLOBAL AND LOCAL VARIABLES

Any variable declared within a block or which appears as an <argument> to a procedure or function will be local only to that single structure. As soon as a structure's END statement is processed, the memory space used by variables local to that structure can (and will) be reclaimed by S-BASIC. When a local variable is created, it is considered global to any blocks nested within the outer block in which the variable was originally declared.

In standard BASIC, all variables are global. This gives the programmer an unenviable choice; either to assign unique variable names for each new program element, thus wasting memory and compromising execution speed, or conversely reusing selected variables. The latter choice decreases program readability in proportion to the number of reassigned variables and is often the very way in which serious, hard to find bugs first find their way into a program.

In some of the classical structured languages (especially Pascal), the compiler may have to resolve a conflict which arises when a global and local variable share the same symbolic name. In such cases, the local will generally have precedence, thus also modifying the global. This can produce disastrous results which neither the compiler nor the object program recognize as errors. In S-BASIC the 'standard' has been redefined so that global variables have precedence, with subsequent attempts to declare conflicting local variables being unacceptable to the compiler. Apparently, this has occasioned a few complaints from users who are accustomed to doing things the other

way, but this variance from the standard seems consistent with the S-BASIC design philosophy of only attempting change for good reason.

Mr. Ohnysty suggested a method for ordering the elements of an S-BASIC program which, in addition to promoting good structure, will help the programmer avoid the kind of problems which can result if there is an assignment conflict between global and local variables. The recommended method requires declarations to be made as follows (note: any of items 1-5 below might be optional):

DECLARE:

- 1: Variables global to program, procedures and/or functions
- 2: COMMON variables
- 3: PROCEDURES [and their local variables]
- 4: FUNCTIONS [and their local variables]
- 5: Variables local to program only
And only then:
- 6: The beginning of the main program.

In the course of answering some of my questions about S-BASIC, Mr. Ohnysty explained at length the rationale behind setting up an S-BASIC program **exactly** as shown. It is beyond the scope of this article to list the (technical) reasons why this approach is so strongly recommended; but, since this advice is offered courtesy of S-BASIC's author, it should be copied and attached to each user's manual! Incidentally, although the manual is good, I feel that it would be much better if it included more helpful hints, such as the important one above.

CONTROL STRUCTURES

S-BASIC offers the widest variety of control structures of any BASIC offered to date. Those listed below can be used exactly as in other BASICs:

```

GOTO/GO TO
GOSUB
ON GOTO
ON GOSUB
IF-THEN [ELSE]
FOR-NEXT [STEP]
ON ERROR GOTO

```

Interestingly, the ability to de-

clare character type variables permits statements of the form:

```

FOR LETTER = 'A' TO 'Z'
<body>
NEXT LETTER

```

The ON ERROR GOTO statement typically selects one of several user-written error handling procedures, depending on the value of an error code which is generated at 103H when non-fatal errors occur at run time. Fatal run time errors require the source program to be corrected and recompiled.

S-BASIC offers the REPEAT-UNTIL and WHILE-DO commands which like the more familiar FOR-NEXT are used for iteration. The use of these commands is shown in figures 3a and 3b. Standard BASIC code which would produce the same results is shown in figures 4a and 4b.

In examples 4a and 4b the loss of top-down structure occurs as a result of backward jumps. Even if REPEAT-UNTIL or WHILE-DO are used as structured alternatives, it is likely that backward references are being internally generated by the compiler, this however should be of no concern to the programmer.

The CASE-OF statement is another useful S-BASIC implementation. CASE-OF accomplishes the same thing as IF-THEN:GOTO (or possibly ON-GOTO) in standard BASIC. See figures 5a, 5b and 5c.

Using the CASE-OF structure avoids all the jumps which are evident in examples 5b and 5c.

Except for the ON ERROR GOTO statement, it is possible to write substantial programs in S-BASIC without using numbered lines. Still, S-BASIC offers line numbering, or more accurately, line labeling. Any unreserved ASCII character may be used within a line label as long as the first character is numeric. I would recommend choosing a single digit, such as '0' to place at the beginning of each line label, thereby making the S-BASIC restriction insignificant. See figures 6a and 6b.

As illustrated in example 6b, S-BASIC's line labels support highly readable code. There is also ano-

ther important benefit; REM statements used to clarify GOSUB and GOTO references in standard BASIC are no longer needed when self explanatory label names are used. Using GOSUBs and GOTOs (except with the ON ERROR statement) is not the ideal approach to programming in S-BASIC, but it is worth noting that these capabilities have been implemented most intelligently.

S-BASIC includes many user oriented convenience features which will provide the principal focus of next month's installment. One such feature, which allows the user to create formatted screen displays on the very first try, is so useful that I will preview it here.

TEXT <device>,<delimiter>

For example, a menu screen might be set up like this:

```
REM DEVICE 0 IS THE CONSOLE
TEXT 0,&
ENTER:
    1: To create a new file
    2: To erase an old file
    3: To create index file
&
```

Everything between the delimiting (&) characters will appear exactly as entered. PRINT statements, enclosing quotation marks and TAB statements are no longer needed! In addition, the TEXT command is fast!

Author's note: After just one month's exposure to S-BASIC, I feel confident saying that we are dealing with an exceptional product. In order to prepare this review, I have written many short test programs (and some not so short test programs), which were designed to isolate potential problems. To date, S-BASIC has performed flawlessly in all of these tests. Naturally, further testing with even larger programs will be required to confirm or disprove [my] initial positive reaction to S-BASIC. But, since first using XYBASIC some four years ago, I have not been encouraged to this degree by the apparent reliability of any new BASIC.

Figure 1a
A 'structured' program
in standard BASIC

```
-----
BEGINNING OF PROGRAM
DEFINE FUNCTION
<some code>
CALL DEFINED FUNCTION
CALL SUBROUTINE
<more code>
LOGICAL END OF PROGRAM
BEGINNING OF SUBROUTINE
<body of subroutine>
RETURN FROM SUBROUTINE
PHYSICAL END OF PROGRAM
-----
```

Figure 1b
A true structured program
in S-BASIC

```
-----
DECLARE VARIABLES
DEFINE PROCEDURE
DEFINE FUNCTION
BEGINNING OF PROGRAM [block]
<some code>
CALL DEFINED FUNCTION
INVOKE PROCEDURE
<more code>
END OF PROGRAM [block]
LOGICAL & PHYSICAL END OF PROGRAM
-----
```

Figure 2a
Using subroutines - standard BASIC

```
10 INPUT "ENTER DEGREES FAHR. "; DF
20 GOSUB 100
30 PRINT "DEGREES CELSIUS = "; DC
40 STOP
100 DC=INT(((DF-32)*5)/9)
110 RETURN
```

Figure 2b
Using procedures - S-BASIC

```
-----
VARIABLE DGS,F=REAL
-----
PROCEDURE FAR.TO.CEL
INPUT "ENTER DEGREES FAHR. "; DGS.F
PRINT "DEGREES CELSIUS = ";
PRINT INT(((DGS.F-32)*5)/9)
END
-----
BEGIN
FAR.TO.CEL
END
-----
```

Figure 3a
THE REPEAT - UNTIL COMMAND

```
-----
REPEAT
BEGIN
A=A+X
END
UNTIL A>100
-----
```

Figure 3b
THE WHILE - DO COMMAND

```
-----
WHILE A<=100 DO
BEGIN
A=A+X
END
-----
```

Figure 4a
Duplicating REPEAT - UNTIL

```
-----
100 A=A+X
110 IF A>100 THEN 130
120 GOTO 100
130 {rest of program}
-----
```

Figure 4b
Duplicating WHILE - DO

```
-----
100 A=A+X
110 IF A<=100 THEN 100
120 {rest of program}
-----
```

Figure 5a
THE CASE - OF STATEMENT

```
-----
CASE REPLY OF
A:RET.TO.MENU
B:REENTER.FILENAME
C:ABORT
END
-----
```

Figure 5b
DUPLICATING CASE - OF

```
-----
10 REM USING IF-THEN:GOTO
100 INPUT "ENTRY ?"; R$
110 IF R$="A" THEN 1000
120 IF R$="B" THEN 2000
130 IF R$="C" THEN 3000
140 GOTO 100
1000 REM MENU
2000 REM REENTER FILENAME
3000 REM ABORT PROGRAM
-----
```

Figure 5c
DUPLICATING CASE - OF

```
-----
10 REM USING ON/GOTO
100 INPUT "ENTRY ?"; R$
110 IF ASC(R$)<65 THEN 100
120 A=ASC(R$)-64
130 ON A GOTO 1000,2000,3000
140 GOTO 100
1000 REM MENU
2000 REM REENTER FILENAME
3000 REM ABORT PROGRAM
-----
```

Figure 6a
NUMERIC LINE LABELS - STD. BASIC

```
-----
100 REM CLOSE FILES
110 GOSUB 1236
120 REM GO BACK TO CP/M
130 GOTO 9999
-----
```

Figure 6b
ALPHANUMERIC LINE LABELS - S-BASIC

```
-----
GOSUB 0.CLOSE.FILES
GOTO 0.BACK.TO.CPM
-----
```

Tips and Techniques

Editor's Note: Below is our winning tip; in addition, we are publishing two other contributions.

I am one of those poor people with only a 64 character-wide video display (Polymorphics VTI). Since my first days with my new CP/M system, I was very unhappy with the way the D(ump) command of DDT program outputs to the screen. Apparently, the Digital Research authors only considered those fortunate people with 'standard' 80-wide displays when they developed this command output format.

In order to more easily read the hex and ASCII format, the space that is placed between each hex value can be removed to reduce the characters per line to under 64



for displays like mine. Since the same technique is used in both DDT and SID, I have given the patches for DDT (versions 1.4 and 2.2) and SID (version 1.4). I have not seen ZSID; so if the output is similar, I hope someone will supply the appropriate patch.

Both DDT and SID are written to relocate themselves to the highest available memory after initial loading at hex address 100. We can use this non-relocated copy to install the necessary patches. The patching process is very simple. Enter DDT or SID without any operands.

For DDT Vers 1.4 or 2.2

1. List address A16 which should be a JNZ 805.
2. Change this instruction to JNZ 808.
3. Exit from DDT (Control C) and save 19 DDT2.COM.

For SID Vers 1.4

1. List address AA4 which should be a JNZ 893.

2. Change this instruction to JNZ 896.
3. Exit from SID (Control C) and save 28 SID2.COM.

Re-execution of DDT2 or SID2 will call in the corrected versions. Upon verification of the change you may wish to then rename these new versions back to the original names.

Tom Cochran

The following tip is a time-saver for users of Wordstar. It concerns the print command ("P" or "ctl. KP") and permits you to more quickly and easily accept default responses to the queries which follow the command ("DISK FILE OUTPUT?", etc.). Normally when you wish to print a file and accept the defaults it is necessary to carriage return through all the questions. A faster way to accomplish this task is to depress the "ESCAPE" button on your keyboard, immediately after naming the file to be printed.

Anonymous



New Products

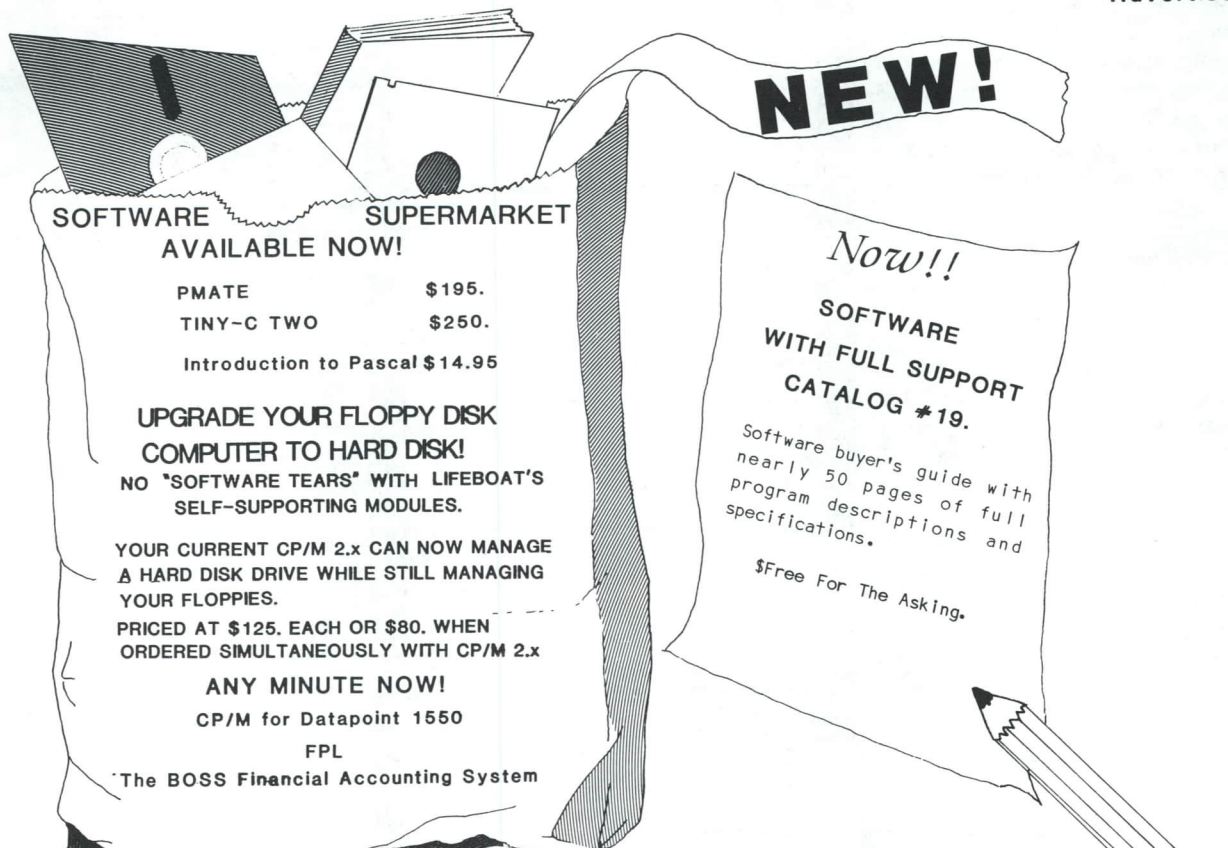
BOSS Financial Accounting System by Balcones

This system includes Accounts Receivable, Accounts Payable, and General Ledger. BOSS is on line at all times; it supports a total of 9,999 customers and vendors. Statements, aged reports, checks and mailing labels can be produced with the various journals.

In addition, BOSS includes the following among its most important attributes: statements of changes in financial position are supplied; financial ratio analysis for management, reporting and planning is handled; loan payments are calculated; amortization and depreciation schedules

(continued next page)

Advertisement



SOFTWARE SUPERMARKET AVAILABLE NOW!

PMATE	\$195.
TINY-C TWO	\$250.

Introduction to Pascal \$14.95

UPGRADE YOUR FLOPPY DISK COMPUTER TO HARD DISK!

NO "SOFTWARE TEARS" WITH LIFEBOAT'S SELF-SUPPORTING MODULES.

YOUR CURRENT CP/M 2.x CAN NOW MANAGE A HARD DISK DRIVE WHILE STILL MANAGING YOUR FLOPPIES.

PRICED AT \$125. EACH OR \$80. WHEN ORDERED SIMULTANEOUSLY WITH CP/M 2.x

ANY MINUTE NOW!

CP/M for Datapoint 1550

FPL

The BOSS Financial Accounting System

NEW!

Now!!

SOFTWARE WITH FULL SUPPORT CATALOG #19.

Software buyer's guide with nearly 50 pages of full program descriptions and specifications.

\$Free For The Asking.

CALL OR WRITE
(212) 860-0300
TELEX 640693

LIFEBOAT ASSOCIATES
1651 THIRD AVENUE

NEW YORK, NY 10028

are implemented. Departments can be assigned separate income statements and balance sheets. Programs and data are checked with each reading; this helps keep the user alerted to hardware and/or disk malfunction.

BOSS has three levels of password security and immediate posting. At the top security level changes and corrections may be made within a given accounting period, without an adjusting entry.

Other journals which can be produced include: Cash Receipts, Cash Disbursements, General Journal, Transaction Journal, Purchases Journal, Sales Journal, and Check Register. They may be printed for a whole period or according to user-selected dates. The journals can be sorted by: reference number; transaction, entry or due date; check, invoice, or purchase order number; customer or vendor number. Data entry is simplified, using a facsimile of manual accounting methods, employing word processing-like controls.

A configuration program allows keyboard customization. This package was written using Microsoft's BASIC-80 compiler. It requires a 48K CP/M, two floppy drive, a 132-column printer, and an 80 by 24 CRT.

CONDOR

by Condor Computer Corporation
A detailed review of this database manager appears in this issue, pages 6-13. It requires a CRT with an addressable cursor and clear-screen functions, as well as a printer which will handle form-feeds.

PAS-3 Medical

PAS-3 Dental

by Artificial Intelligence, Inc.
The Patient Accounting System is a video-oriented system for account-management and billing in individual or group practice. PAS-3 Dental differs from Pas-3 Medical only in details. For instance, when charges are entered for treatments, the dental user is asked by a screen-displayed message to supply tooth numbers and surface numbers. Screen prompts of this kind are a constant feature of both PAS-3 versions; both are also menu-referenced.

Reusable formats for both patients' and insurance bills are created by the user. Item headings can be omitted or placed anywhere on the printed sheet. A library of "blank" insurance billing forms can be produced; at billing time PAS-3 will call the appropriate form by code number from each patient's record, and complete it.

Ten different types of patients' accounts are supported ("no bill", "hold bill", "welfare", "insurance budget plan", "cash", "insured with no bill to patient", etc.) and preauthorized treatment charges can be included on insurance bills. If the user wishes, PAS-3 will calculate service charges for overdue amounts and add it to patients' bills.

Current charges are entered as incurred, and records are reviewed on screen and may be printed out. PAS-3 consults a recall file to make a set of mailing labels for patients who should be reminded of impending appointments. Mailing labels are produced for all other purposes by a program that picks out records using a key-search formula set up by the user from such data categories as date of last visit, amount greater than..., age over..., and others.

Built-in routines produce (and then print out or screen-display) arrears aging reports, an "undefeatable" delinquent charges list, a producer (practitioner) activity report, a new accounts list, summaries of monthly and daily charges and receipts, and procedure profitability analyses. A query program allows the user to format special analytical summaries using two search-keys simultaneously.

The privacy of PAS-3 records is protected by a password user-access system. You must have at least 56K with CP/M 2.xx (48K with earlier CP/M versions). CBASIC-2 (2.06 or higher) is required.

PMATE

by Phoenix Software Associates, Ltd.

This text editor is reviewed in this issue, pages 3-5. It requires a cursor addressable serial terminal or memory mapped video.

Tiny C Two

by Tiny C Associates

This compiler written in Tiny C requires 32K. The disk includes the source code, faster I/O than Tiny C, and intermediate code-emitting capabilities.

New Versions

CP/M 1.42 for Altair, ICOM 3712, and ICOM 3812

This change implements the patch reported in the October Lifelines (Volume 1, Number 5, page 14). With this new version, Microsoft BASIC, Pascal M, MDBS and other software will run properly. Per-tec BASIC will not operate with this version.

CP/M 2.22 on North Star

The following bugs have been fixed in this version:

- 1-CONFIG version 4.0 corrects device selection in DSPCH routine of the Horizon User Area. In CONFIG 3.x RRC is now RLC, at location 105F.
- 2-Read after write has been repaired.
- 3-An attempt to write on a disk with a write protect tab will now yield an immediate disk error message.
- 4-The cold boot loader now returns to location E800H PROM upon error.

Microspell Version 4.1

The following changes have been made in this version:

- 1-Microspell will now check single-letter words after BUILD has been used to insert them in the dictionary.
- 2-The maximum word length has been extended to 29 characters. Oversized words will be flagged with a "? TOO BIG" message.
- 3-The interrupt manipulation has been removed. Microspell now works with computers which will cease to function with the interrupts disabled (e.g., Super-brain, Altos).
- 4-The QUIT command now requests
(continued next page)

verification (Y/N) to avoid accidental aborting of the program.

5-To save disk space, an /Q switch has been added; it suppresses the creation of a backup file.

6-One guessing routine has been expanded so that more guesses are provided.

7-Suffix stripping and verification routines are tighter.

8-The dictionary has been expanded to over 25,000 entries.

9-The line numbers of target words and suffix expansions are now printed. This feature is only functional when the file uses line feeds in the usual manner.

10-The BUILD utility will now report the total number of entries added to each LEX file, including misspelling/correction pairs. If there are no changes it will not write the current LEX file back to disk.

These bugs have been corrected:

- 1-A word with an apostrophe as its second character is no longer treated as two words.
- 2-Only those words recognized by suffix stripping will now appear as upper case in the EXC file.
- 3-LEX files now contain the version number, so that BUILD and INVERT cannot perform potentially disastrous changes on incompatible versions of LEX. SPELL, in this version, does not check the version number.

Pascal/Z Version 3.3

These changes have been made in the new version:

- 1-Separate modules are numbered, allowing separate compilation and accurate tracing across modules.
- 2-INCLUDE files are implemented. INCLUDE temporarily changes the input file to the assembler, allowing code in another file to be inserted into a program during assembly. When the INCLUDED file is exhausted, the assembler resumes reading the source lines from the original source file, recommencing with the line after the INCLUDE instruction. Nested INCLUDEs are not allowed; a file which is an argument to the instruction may not itself contain any INCLUDE instructions.
- 3-RENAME and ERASE have been

- added.
- 4-There is a new floating point/fixed point output format.
 - 5-Explicit input from the console can be implemented, using CON:.
 - 6-The compiler now checks for implementation limitation of forty fields per RECORD in RECORD declarations.
 - 7-Range-checking of SETs has been improved.
 - 8-'.' is now checked for at the end of a .PAS file.
 - 9-Duplicate field names are now checked in a RECORD.
 - 10-Ten (instead of nine) significant characters are permitted in reserved words.
 - 11-Eight-character file names with four-character extensions are corrected for.
 - 12-All SETs are 32 bytes in size.
 - 13-Value parameters passed to user routine are checked.
 - 14-New UCTRANS files have been added.
 - 15-EXTERNALS can be declared only in the main module, not in separate ones.
 - 16-A new listing file has page, statement, line and nesting level numbers.

These bugs have been righted:

- 1-Run time range errors are better reported.
- 2-Pascal/Z permits output of functions returning a structured type which normally can be output.
- 3-String relational operations for strings 255 characters long are handled properly.
- 4-Floating point output has been corrected to handle "2 to the 23rd" and "2 to the 23rd plus one" output.
- 5-Output of a REAL expression to a non-text file is corrected.
- 6-The VAR parameter is fixed.
- 7-Now there is correct compensation for a function returning a string, when that function is the parameter of one expecting a different length string.



Bugs & Bug Fixes



Pascal/Z Version 3.3

A bug occurs in LINK/Z when the /O (offset) and /P (load) options are used. (These are the only details we have from the manufacturer.)

BUG FIX

PLINK

This fix is for users of Plink who wish to link programs meant to run on 8080/8085 target machines. PLINK 3.25 introduces a Z80 opcode into programs it links. This fix corrects the problem.

```
A>DDT B:PLINK.COM
DDT VERS 1.4
NEXT PC
4280 0100
-S2FDB
2FD8 ED 2A
2FD9 7B 06
2FDA AF
2FDB CD
2FDC D3
2FDD 32
2FDE 21
2FDF 06 00
2FE0 00 F9
2FE1 AF
A>^CSAVE 66 PLINK.COM
```

Coming

In the next few months we'll be featuring the catalogue and abstracts of a new CPMUG volume. Next month the final installment of Martin McNiff's Osborne tutorials (describing Payroll) will appear. In addition, a review of SSS FORTRAN is in the works, along with more commentary on some other packages from Phoenix Software Associates, Ltd.

We hope you'll be contributing more tips and techniques--the contest is still on. If you're a recent subscriber and haven't heard about the contest, give us a call at 212-722-1700 for details.

MICROPROCESSOR SUPPORT I.C.'S

WE GUARANTEE FACTORY PRIME PARTS

We are going to become the largest supplier of prime microprocessor support I.C.'S. We guarantee that our I.C.'S are purchased from manufacturer authorized distributors. This is the only way to deliver prime parts at the lowest possible prices. Our commitment is to offer the best price and the fastest delivery to our customer. We give many thanks to our valued customers who have helped us grow.

NEC 16Kx1 DYNAMIC RAM 200 N.S.
These are prime 4116's from one of the best MOS RAM manufacturers in the world.

4116 **200ns**
8 for \$25.00 **32 for 96.00**
Only prepaid orders on this special.

8080A CPU	4.95	2708 EPROM 1Kx8	4.95
8085A CPU	8.95	2716 EPROM 2Kx8	8.95
8086 CPU	99.95	2732 EPROM 4Kx8	21.00
8088 CPU	44.95	4118 STATIC 1Kx8	15.00
Z-80 CPU	10.50	4164 200ns 64Kx1 Call	
Z-80A CPU	12.95	Z80B CPU	21.00

NEC 1Kx4 STATIC RAM 250 N.S.
These are prime low power static ram's NEC for the finest in MOS MEMORY.

2114L **250ns**
8 for \$25.00 **32 for \$96.00**
Only prepaid orders on this special.

Z80-P10	7.75	8289	49.95	4050	.69	4531	.99	74C925	6.95	74LS107	.45	74LS244	1.95
Z80A-P10	9.75	4000	.35	4051	1.10	4532	1.25	74LS00	.35	74LS109	.45	74LS245	4.95
Z80-CTC	7.75	4001	.35	4052	1.10	4539	.99	74LS01	.28	74LS112	.49	74LS247	1.10
Z80ACTC	9.75	4002	.35	4053	1.10	4543	1.99	74LS02	.28	74LS122	.55	74LS248	1.10
Z80-DMA	22.25	4006	1.39	4055	3.95	4553	3.50	74LS03	.28	74LS123	1.19	74LS249	1.69
Z80A-DMA	27.75	4007	.29	4056	2.95	4555	.75	74LS04	.39	74LS125	1.35	74LS251	1.79
Z80-S10/0	24.95	4008	1.39	4059	9.95	4556	.75	74LS05	.28	74LS126	.89	74LS253	.98
Z80A-S10/0	29.95	4009	.49	4060	1.39	4581	1.99	74LS08	.39	74LS132	.79	74LS257	.98
Z80-S10/1	24.95	4010	.49	4066	.75	4582	1.01	74LS09	.39	74LS136	.59	74LS258	.98
Z80A-S10/1	29.95	4011	.35	4068	.35	4584	.55	74LS10	.28	74LS138	.89	74LS259	2.95
Z80-S10/2	24.95	4012	.29	4069	.35	4585	.99	74LS11	.39	74LS139	.89	74LS260	.69
Z80A-S10/2	29.95	4013	.49	4070	.49	4702	9.95	74LS12	.39	74LS145	1.25	74LS261	2.49
3205	3.45	4014	1.39	4071	.35	74C00	.39	74LS13	.47	74LS148	1.49	74LS266	.59
3242	10.00	4015	1.15	4072	.35	74C02	.39	74LS14	1.25	74LS151	.79	74LS273	1.75
8155	11.25	4016	.59	4073	.35	74C04	.39	74LS15	.39	74LS153	.79	74LS275	4.40
8185	29.95	4017	1.19	4075	.35	74C08	.49	74LS20	.26	74LS155	1.19	74LS279	.59
8185-2	39.95	4018	.99	4076	1.29	74C10	.49	74LS21	.38	74LS156	.99	74LS283	1.10
8202	45.00	4019	.49	4078	.35	74C14	1.65	74LS22	.38	74LS157	.99	74LS290	1.29
8205	3.45	4020	1.19	4081	.35	74C20	.39	74LS26	.39	74LS158	.75	74LS293	1.95
8212	2.00	4021	1.19	4082	.35	74C30	.39	74LS27	.39	74LS160	.98	74LS295	1.10
8214	3.95	4022	1.15	4085	1.95	74C32	.99	74LS28	.39	74LS161	1.15	74LS298	1.29
8216	1.85	4023	.38	4086	.79	74C42	1.85	74LS30	.26	74LS162	.98	74LS324	1.75
8224	2.65	4024	.79	4093	.99	74C48	2.39	74LS32	.39	74LS163	.98	74LS347	1.95
8226	1.85	4025	.38	4099	2.25	74C73	.85	74LS37	.79	74LS164	1.19	74LS348	1.95
8228	5.00	4026	2.50	4104	1.99	74C74	.85	74LS38	.39	74LS165	.89	74LS352	1.65
8238	5.45	4027	.65	4501	.39	74C85	2.49	74LS42	.79	74LS166	2.49	74LS353	1.65
8243	4.65	4028	.85	4502	1.65	74C89	4.95	74LS47	.79	74LS170	1.99	74LS363	1.49
8251A	5.55	4029	1.29	4503	.69	74C90	1.85	74LS48	.79	74LS173	.89	74LS365	.99
8253	9.85	4030	.45	4505	8.95	74C93	1.85	74LS51	.26	74LS174	.99	74LS366	.99
8255A	5.40	4031	3.25	4506	.75	74C95	1.85	74LS54	.35	74LS175	.99	74LS367	.73
8255A-5	5.40	4032	2.15	4507	.95	74C107	1.19	74LS55	.35	74LS181	2.20	74LS368	.73
8257	9.25	4033	2.15	4508	3.95	74C151	2.49	74LS73	.45	74LS190	1.15	74LS373	2.75
8257-5	9.25	4034	3.25	4510	1.39	74C154	3.50	74LS74	.59	74LS191	1.15	74LS374	2.75
8259A	7.30	4035	.95	4511	1.39	74C157	2.10	74LS75	.68	74LS192	.98	74LS375	.69
8271	60.00	4037	1.95	4512	1.39	74C160	2.39	74LS76	.45	74LS193	.98	74LS377	1.95
8275	32.95	4040	1.29	4514	3.95	74C161	2.30	74LS78	.65	74LS194	1.15	74LS385	1.95
8279	10.80	4041	1.25	4515	3.95	74163	2.39	74LS83	.99	74LS195	.95	74LS386	.65
8279-5	10.80	4042	.95	4516	1.69	74164	2.39	74LS85	1.19	74LS196	.89	74LS390	1.95
8282	6.70	4043	.85	4519	.99	74173	2.59	74LS86	.45	74LS197	.89	74LS393	1.95
8283	6.70	4044	.85	4520	1.39	74174	2.75	74LS90	.75	74LS221	1.49	74LS395	1.70
8284	5.85	4046	1.75	4522	.99	74C175	2.75	74LS92	.75	74LS240	1.95	74LS399	2.95
8286	6.70	4047	1.25	4526	1.15	74C192	2.39	74LS93	.75	74LS241	1.90	74LS424	2.95
8287	6.70	4048	.99	4527	1.75	74C193	2.39	74LS95	.88	74LS242	1.95	74LS668	1.75
8288	25.40	4049	.69	4528	.99	74C195	2.39	74LS96	.98	74LS243	1.95	74LS670	2.29

MAIL ORDERS SHOULD BE SENT TO:
P.O. Box 21432 Seattle, Washington 98111
Telephone Orders & Inquiries (206) 453-0792
Minimum Order \$10.00 Add \$3.00 Shipping

HANLEY ENGINEERING
RETAIL STORE
1644 116th NORTHEAST
BELLEVUE, WASHINGTON 98005

FOR THE FINEST IN MICROPROCESSOR SUPPORT I.C.'S

VERSION LIST

The below software is available from
the authors, computer stores, and
distributors.

PRODUCT NAME	S	M	OS	P	MR	\$	
A-NATURAL Assembler Package	1.1		CP/M	8080		330/15	
A3+ Development Package			CP/M	Z80		409/40	
A4 Development Package			CP/M	Z80		299/40	
Accounts Payable/Cybernetics	3.1		CP/M	Z80	64K	500	Needs RM/COBOL
Accounts Payable/Graham Dorian	1.11	1.11	CP/M	8080	48K	805/40	Needs CBASIC2
Accounts Payable/Structured Sys	1.3B		CP/M	8080	52K	840/40	W/IT WORKS run time pkg.
Accounts Payable/Peachtree	10-10-80		CP/M		48K	530/60	Needs BASIC-80 4.51
Accounts Receivable/Cybernetics	3.1		CP/M	Z80	64K	500	Needs RM/COBOL
Accounts Receivable/Graham Dorian	1.08	1.08	CP/M	8080	48K	805/40	Needs CBASIC2
Accounts Receivable/Peachtree	10-10-80		CP/M	8080	48K	530/60	Needs BASIC-80 4.51
Accounts Receivable/Structured Sys	1.4C		CP/M	8080	56K	840/40	W/IT WORKS run time pkg.
ALDS TRSDOS	3.38		TRSDOS	8080	32K	80/25	TRSDOS Macro-80
ALGOL 60 Compiler	4.8C		CP/M	8080	24K	199/20	
ANALYST	1.0		CP/M	8080	52K	250/20	Needs CBASIC2, QSORT/VSORT
APL/V80	3.2		CP/M	Z80	48K	500	Needs APL terminal
Apartment Management	1.03	1.03	CP/M	8080		805/40	Needs CBASIC2
ASM/XITAN	3.11		CP/M	Z80		69/20	
Automated Patient History	1.2		CP/M	8080	48K	175	
*BASIC-80 Compiler	5.24	5.24	CP/M	8080	48K	360/35	
BASIC-80 Compiler		5.1	TRSDOS		64K	400/25	TRS-80 Model II only
BASIC-80 Interpreter	5.2	5.2	CP/M	8080	40K	335/35	W/Vers. 4.51, 5.2
BASIC Utility Disk	2.0	2.0	CP/M	8080	48K	75	
*BSTAM Communication System	4.5	4.5	CP/M	8080	32K	200	
*BDS C Compiler	1.43	1.43T	CP/M	8080	32K	150/30	W/"C" book
Whitesmiths' C Compiler	2.0		CP/M	8080	60K	630/30	
BSTMS	1.2	1.2	CP/M	8080	24K	200	
BUG / uBUG Debuggers	2.03		CP/M	Z80		129/25	
Cash Register	2.0	2.0	CP/M	8080		805/40	
CBASIC Compiler	2.07P	2.17P	CP/M	8080	32K	125/20	2.17= modified 2.07
CBS Applications Builder	1.2		CP/M	8080	48K	395/40	Needs no support language
CIS COBOL Compiler	4.3.1		CP/M	8080	48K	850/50	
*CIS COBOL Compact	3.46	3.46	CP/M	8080	32K	650/50	
FORMS 1 COBOL Form Generator	1.06	1.06	CP/M	8080		150/20	
FORMS 2 COBOL Form Generator	1.1.6	1.1.6	CP/M	8080		200/20	
COBOL-80 Compiler	4.01	4.01	CP/M	8080	48K	710/35	
*COBOL-80 PLUS M/SORT	4.01		CP/M	8080	48K	845/55	
*CONDOR	1.10		CP/M	8080	48K	695/35	
CREAM (Real Estate Acct'ng)	2.3		CP/M	8080	64K	250	CBASIC needed
DATASAR Information Manager	1.1		CP/M	8080	48K	350/60	
Datebook	1.05		CP/M	8080	48K	295/30	Needs 80x24 terminal
DESPool Print Spooler	1.1A		CP/M	8080		80	
DISILOG Z80 Disassembler	4.0	4.0	CP/M	Z80		110	Zilog mnemonics
DISTEL Z80/8080 Disassembler	4.0		CP/M	8080/Z80		110	Intel mnemonics, TDL extensions
EDIT Text Editor	2.06		CP/M	Z80		129/25	
EDIT-80 Text Editor	2.0		CP/M	8080		99/15	
ESQ-I	1.0		CP/M	8080		1495/50	Needs CBASIC2
ESQ-I DEMO	1.0		CP/M	8080		75	
FILETRAN	1.20		CP/M		32K	99	1-way TRS-80 Mod I, TRSDOS to Mod I CP/M
FILETRAN		1.4	TRSDOS		32K	149	2-way TRS-80 Mod I, TRSDOS & Stnd. CP/M
FILETRAN	1.4		CP/M		32K	119/20	2-way TRS-80 Mod I, TRSDOS & Mod I CP/M
FILETRAN	1.5		CP/M		32K	99	1-way TRS-80 Mod II, TRSDOS to Mod II CP/M
Financial Modeling System	2.0		CP/M		48K	300	
FORTTRAN-80 Compiler	3.4	3.36A	CP/M	8080	36K	435/35	
FORTTRAN Package	3.38		TRSDOS			80/25	
*FPL	1.52	1.52	CP/M	8080	56K	695/30	
General Ledger/Cybernetics	1.3C		CP/M	Z80	48K	500	Needs RM/COBOL
General Ledger/Graham Dorian	1.09	1.09	CP/M	8080	48K	805/40	Needs CBASIC2
General Ledger/Peachtree	10-10-80		CP/M	8080	48K	530/60	Needs BASIC-80 4.51
General Ledger/Structured Sys	1.4C		CP/M	8080	52K	840/40	No need for CBASIC
General Ledger II/CPAids	1.1		CP/M	8080	48K	450/30	Needs BASIC-80 4.51
GLECTOR Accounting System	2.0		CP/M	8080	56K	350/25	Use w/CBASIC2, Selector III
HDBS	1.04		CP/M	+	52K	300/35	
HDBS	1.04D		TRSDOS		56K	300/35	Runs only w/Radio Shack BASIC
HDBS.SRS	1.03		TRSDOS		56K	150/5	Runs only w/Radio Shack BASIC
HDBS	1.04D		APPLE		56K	300/35	
IBM/CPM	1.1		CP/M	8080		175	
Inventory/Graham Dorian	1.0	1.0	CP/M	8080		555/40	Needs CBASIC2
Inventory/Peachtree	10-10-80		CP/M	8080	48K	530/60	Needs BASIC-80 4.51
Inventory/Structured Sys	1.0C		CP/M	8080	52K	840/40	No need for CBASIC
Job Costing	2.02	2.02	CP/M	8080	48K	805/40	Needs CBASIC2
KBASIC Interpreter	2.03		CP/M	8080	48K	585/45	
KISS File Management System	2.03		CP/M	8080	40K	335/23	
*LETTERIGHT Text Editor	1JB		CP/M	8080	52K	200/25	
LEVEL 3 BASIC / G2			TRSDOS			45	Cassettes
LINKER			CP/M	Z80		69	
MAC	2.0		CP/M	8080	20K	120/25	
MACRO-80 Macro Assembler Package	3.40	3.40	CP/M	8080/Z80		149/15	
Magic Wand	1.1		CP/M	8080	32K	395/40	
MAGSAM III	4.1		CP/M	8080	32K	145/25	
MAGSAM IV	1.0		CP/M	8080	32K	295/25	
MAILING ADDRESS Mail List System	8-13-80		CP/M	8080	48K	530/60	
*Mail-Merge	2.26		CP/M	8080		150/25	
Master Tax	1.0-80		CP/M	8080	48K	995/30	Has 1980 tax forms
MDBS	1.04		CP/M	+	48K	900/35	
MDBS-DRS	1.02		CP/M	+	52K	300	
MDBS-QRS	1.0		CP/M	+	52K	300	
MDBS-RTL	1.0		CP/M	+	52K	300	
MDBS-PKG			CP/M	+	52K	1500/60	W/all above MDBS products
MDBS	1.04D		Apple		56K	900/35	

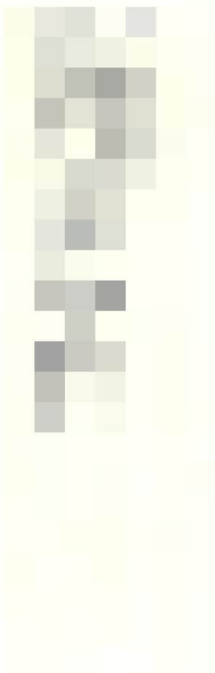
S Standard Version
M Modified Version.
OS Operating System.
P Processor
MR Memory Required
\$ Price
* Indicates a new version or new product.
+ These products are available in Z80 or 8080, in the following host languages: BASCOM, COBOL-80, FORTRAN-80, PASCAL/M, PASCAL/Z, CIS-COBOL, CBASIC, PL/I-80, BASIC-80 4.51, and BASIC-80 5.xx.

VERSION LIST

PRODUCT NAME	S	M	OS	P	MR	\$	
Microspell	4.1		CP/M	8080	48K	249	Needs 150K
Mini-Warehouse Mngmt. Sys.	5.5		CP/M	8080	48K	650	Needs CBASIC
MP/M Operating System	1.1		MP/M	8080	32K	300/50	
MSORT	4.01		CP/M	8080	48K	160/20	
Mu LISP-80 Compiler	2.03		CP/M	8080		260/30	
Mu SIMP / Mu MATH Package	2.03		CP/M	8080		250/25	muMATH 80
NAD Mail List System	3.0C		CP/M	8080	48K	115/25	
Nevada COBOL	1.403	1.403	CP/M	8080	32K	149/25	
Order Entry w/Inventory/Cybernetics			CP/M	Z80		500	Needs RM/COBOL
*PAS-3 Medical	1.71		CP/M	8080	56K	995/25	Needs 132-col. printer
*PAS-3 Dental	1.61		CP/M	8080	56K	995/25	Needs 132-col. printer
PASM Assembler	1.02		CP/M	Z80		129/25	
Pascal/M	3.2		CP/M	8080	56K	175/25	
PASCAL/MT Compiler	3.2		CP/M	8080	32K	250/30	
PASCAL/MT+	5.1		CP/M	8080	52K	250/30	Also has 32K version
*PASCAL/Z Compiler	3.3		CP/M	8080	56K	395/25	
Payroll/Cybernetics, Inc.			CP/M	Z80		500	Needs RM/COBOL
Payroll/Peachtree	11-7-80		CP/M	8080	48K	530/60	Needs BASIC-80 4.51
Payroll/Structured Sys	1.00		CP/M	8080	60K	840/40	No longer needs CBASIC
PEARL III	3.01		CP/M	8080	56K	900	W/CBASIC2,Ultrastort II
PL/I-80	1.3		CP/M	8080	48K	500	
PLINK Linking Loader	3.25		CP/M	Z80		129/25	
POSTMASTER Mail List System	3.3	3.3	CP/M	8080	48K	150/20	
Property Manager	10-10-80		CP/M	8080		925/90	Needs BASIC-80 4.51
Property Mngmt. Sys.	1.0		CP/M	8080	48K	650	Needs CBASIC
QSORT Sort Program	1.5		CP/M	8080	48K	100	
Real Estate Acquisition Programs	2.1		CP/M	8080	56K	500	Needs CBASIC
Residential Prop. Mngmt. Sys.	1.0		CP/M	Z80	48K	650	
RM/COBOL Compiler	1.3C		CP/M	8080	48K	750	w/Cybernetics CP/M 2
RAID	4.7.3	4.7.3	CP/M	8080	28K	250/25	
RECLAIM Disk Verification Program	2.1		CP/M	8080		80	
SBASIC	5.3h		CP/M	8080		295/35	
SELECTOR-III-C2 Data Manager	3.23	3.23	CP/M	8080	48K	295/20	
*SELECTOR-IV	2.11		CP/M	8080	52K	550/35	Needs CBASIC
SID Symbolic Debugger	1.4		CP/M	8080		120/25	N/A-Superbr'n
SMAL/80 Programming System	3.0		CP/M	8080		75/25	For CP/M 1.x
Standard Tax	1.0		CP/M	8080	48K	495/30	Needs BASIC-80 4.51
STATPAK	1.2	1.2	CP/M	8080		495/30	Needs BASIC-80 4.2 or above
STRING BIT FORTRAN Routines	1.02	1.02	CP/M	8080		75/25	
STRING/80 bit FORTRAN Routines	1.22		CP/M	8080		95/25	
STRING/80 bit Source	1.22		CP/M	8080		295	
SUPER SORT I Sort Package	1.5		CP/M	8080		225/40	Max. record=4096 bytes
T/MAKER Data Calculator	1.4		CP/M	8080	48K	125/25	Needs CBASIC2
TEX Text Formatter	1.1		CP/M	8080	36K	105/15	
TEXTWRITER-III Text Formatter	3.6	3.6	CP/M	8080	32K	125/20	
TINY C Compiler	800102C		CP/M	8080		105/50	
ULTRASORT II	3.1	3.1	CP/M	8080	48K	195/25	
Unlock	1.3		CP/M	8080		95	Use w/ BASIC-80
VisiCalc	1.37		Apple	8080	32K	150	
VSORT Sort Program	1.8	1.8	CP/M	8080	48K	175/20	
WHATSIT? Data Manager	2.04		CP/M	8080	44K	175/25	
*WORDINDEX			CP/M	8080	48K	195	Needs Wordstar
WORDMASTER Text Editor	1.07A		CP/M	8080	40K	145/40	
WORDSTAR Word Processor	2.6	2.20A	CP/M	8080	48K	445/60	
*MAIL MERGE Printer Overlay	2.26		CP/M	8080	48K	575/40	
*WORDSTAR Customization Notes	2.2X		CP/M			95	
XASM-18 Cross Assembler	1.30		CP/M	8080		200/25	
XASM-48 Cross Assembler	1.30		CP/M	8080		200/25	
XASM-65 Cross Assembler	1.95		CP/M	8080		200/25	
XASM-68 Cross Assembler	1.96		CP/M	8080		200/25	
*XMACRO-86 Cross Assembler	3.40		CP/M	8080		275/25	
XYBASIC Interpreter Extended	2.11		CP/M	8080		450/25	
XYBASIC Interpreter Extended CP/M	2.11		CP/M	8080		550/25	
XYBASIC Interpreter Extended COMP	2.0		CP/M	8080		450/25	
XYBASIC Interpreter Extended ROM	2.1		CP/M	8080		450/25	
XYBASIC Interpreter Integer	1.7		CP/M	8080		350/25	
XYBASIC Interpreter Integer COMP	2.0		CP/M	8080		350/25	
XYBASIC Interpreter Integer ROM	1.7		CP/M	8080		350/25	
Z80 Development Package	3.3		CP/M	Z80		130	N/A-Magnolia, Superbr'n, mod. CP/M
ZDT Z80 Debugger	1.41		CP/M	Z80		50	N/A-Superbr'n, mod. CP/M
ZSID Z80 Debugger	1.4		CP/M	Z80		130	N/A-Superbr'n, mod. CP/M
Operating Systems			Intel	MDS	Single Density	1.4	Sol North Star SD 1.41
Description	Version		Intel	MDS	Single Density	2.2	North Star SD IMSAI SIO Console 1.41
			Intel	MDS	800/230 Double Density	2.2	North Star SD MITS SIO Console 1.41
					MITS Altair 3202 Disk	1.41	North Star DD 1.45
					Micropolis Mod I - All Consoles	1.411	* North Star DD/QD 2.22
					Micropolis Mod II - All Consoles	1.42	Processor Technology Helios II 1.41
					* Micropolis Mod I	2.20A	by Lifeboat/TRS-80 5 1/4"(Mod I) 1.41
					* Micropolis Mod II	2.20A	by Lifeboat/TRS-80 Mod II 2.24A
					Compal Micropolis Mod II	1.4	by Cybernetics/TRS-80 Mod II 2.25
					Black Hawk Micropolis Mod II	1.4	by Lifeboat/TRS-80 Mod II+Corvus 2.24-C
					Exidy Sorcerer Micropolis Mod I	1.42	
					Exidy Sorcerer Micropolis Mod II	1.42	
					NYLAC/REX Micropolis Mod II	1.4	OASIS for Altos, Bell Ctrls., Billings, CA
					Vector MZ Micropolis Mod II	1.411	Computer Systems, Compucorp, Cromemco,
					Versatile 3B Micropolis Mod I	1.411	Delta, Digital Microsys., Dynabyte, God-
					Versatile 4 Micropolis Mod II	1.411	bout, GRI, Index Comp. Sys., IBC, In-
					Horizon North Star SD	1.41	tertechnique, Kontron, Media Sys. Corp.,
					Mostek MDX STD Bus	2.2	MicromationDoubler, Morrow Thinker Toys,
					* Ohio Scientific C3	2.23	NNC Elect., Onyx, Quay, S.D.Sys., Teletek,
							TRS-80 Mod.II, Vector Gr., Vorimex, Zilog

Hard Disk Modules Description	Version
Corvus Module	1.6
KONAN Phoenix Drive	1.7
*Micropolis Microdisk	1.9
Pertec	1.6

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



FIRST CLASS MAIL
U. S. POSTAGE
PAID
Permit No. 416

FIRST CLASS MAIL