JULY 1981

LIFELINES®

$2.50

VOLUME TWO NO. 2

MDBS, Part I

The CP/M Users Group:
CPMUG Volume 51
A Tutorial on Volume 51

Zoso

LIFELINES®

Editor-in-Chief: Harris Landgarten
Managing Editor: Jane Mellin
Production Assistant: K. Gartner

Volume II No.2 July

CONTENTS

Lifelines,Volume II, Number 2. 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, $40.00 when destined for any other country. All checks should be made payable to Lifelines Publishing Corporation, Foreign checks must be sent 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 Corporation, 1651 Third Avenue, 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 second class postage pending at New York, N.Y.

Editorial Comments

Last month I covered the interesting software news of NCC. This month I will tell you about the hardware that caught my eye.

Based on the number introduced, I would have to say that hard disks dominated. It seems that every known peripheral manufacturer has announced some sort of small Winchester type hard disk. Most of the people I spoke with were less interested in the specifications than in which of these drives would survive until the next NCC. Of the ones I saw, the Priam, boasting 70 megabytes in an 8-inch drive, was the most alluring. For a full score card, I suggest Infoworld; they somehow knew what we all would see at NCC long before the show was to open.

On the subject of disks, Sony was showing their 3-inch floppy drive. Although it had been previously introduced, NCC was my first chance to see this drive up close, and it is a marvel of miniaturization. Sony is asking very high OEM prices ($1200) for this drive, leading one to speculate as to when Sony itself will introduce a competitively priced portable computer with these drives. Certainly, Sony will venture into the computer market before long.

Sony also demonstrated their 19 megahertz color monitor. The detail of the color graphics is indescribable. Unfortunately, these monitors are too expensive for inclusion with your graphics board and keyboard, for the moment. Any student of history will tell you that it won't be long bfore such monitors are commonplace. While on the subject of terminals, I wonder when the long-awaited LCD displays will be introduced. My eyes will have to tolerate CRTs a little while longer. Maybe next year.

An interesting add-on for CRT terminals was shown by both Lear Seigler and Ampex. The touch-sensitive face plate, which can be mounted in front of any CRT, reports the coordinates of points touched via a separate RS232 port. This is a good alternative to a light pen in many user-oriented applications. Just imagine the fully equipped terminal of the future (sooner than we think), with voice synthesis, voice recognition, and a touch-sensitive screen. I wonder if it will have a keyboard.

Speaking of keyboards, Micro Switch introduced some keyboards at NCC which were so close to typewriter feel that you could barely tell with your eyes closed. I look forward to seeing these beauties incorporated in 'better' terminals later this year. Why only 'better' terminals? Because the manufacturers of bargain terminals would rather give us blinking, flashing, underlining, highlighting, inverse video, and color, than a comfortable keyboard. Probably because such features sell terminals while a good keyboard only serves to make them usable.

On the Japanese front, computer systems were shown by Toshiba, Oki, and NEC. All exhibit wonderful features at competitive prices. The Oki for example has the ability to produce wonderful, high-resolution color graphics. Unfortunately, most of the Japanese machines I have seen have painfully slow disk access times, some taking between 30 and 40 seconds just to load Micosoft BASIC-80. It seems that their CP/M implementations could use a little American 'know-how'.

A new computer was introduced by Televideo. It is composed of a 950 terminal with a 64k, Z80-based, single board computer housed within. The buyer can choose between various floppy and 5-inch Winchester combinations in outboard boxes. Multi-user configurations are accomplished by hooking several of these computers together in a distributed processing type manner. To this end, Televideo will distribute a multi-user operating system called MMM/OST which is CP/M compatible. General Electric service contract will be offered to purchasers. Perhaps, this will be the first 'appliance' computer.

New microcomputer-oriented printers were everywhere at NCC. Slow Daisy wheels, fast Daisy wheels, dense dot matrices, even an ink jet. Among the dot matrix printers, many are attempting letter quality print by either multiple passes or in one case (Toshiba), by a single pass with a 25 wire head. To my eye, none have yet achieved the print quality shown by the Sanders printer. Sharp exhibited an ink jet printer which produced print that was very close to letter quality. Most computer veterans, however, shake with fear when this type of ink jet mechanism is mentioned, and start quoting horror stories of unreliable operation. I hope that, if and when Sharp decides to market this printer, the historic ink jet design hurdles will be overcome.

One of the most significant signs of things to come, however, did not happen at NCC. In fact it was the introduction of a new audio product by Sony, the 4 3/4-inch video disk style audio recording. The specs tell of laser technology and one hour playing times. Movement towards making this a world audio standard are already well underway. I look forward to seeing this type of technology applied to computers; perhaps in the first full fledged 'carry-along' computer.

While any single prediction of the future is risky, it is certain that computer technology will continue to race ahead at an ever-increasing rate, a fact reinforced by this year's NCC. See you next month.

p.s. Last month I quoted incorrect prices for the PDP 11 version of XENIX as available from Lifeboat Associates. The correct prices actually range from $3000 to $10,000 depending on the number of users.

Last month I named several software vendors as the only ones exhibiting at NCC. I omitted mentioning that Microfocus was also there, showing CIS COBOL. Sorry.

Harris Landgarten

Lifelines is accepting applications for software reviewers in many different categories. Please include your field of interest and any samples of past work. Mail all information to me, care of *Lifelines*.

ZOSO

Calumet City, Illinois
May 4, 1981

Note: The first part of this offering begins with my visit to NCC '81. I had hoped this column would appear in the June issue, but I was delayed due to bad weather... Sorry! If belated lowdowns upset you, then curse at the sky and leave me out of it!

How did I end up in Calumet City? Good question! Quite simple; I decided to visit NCC the day it opened, but by then, better accommodations were (more than) fully booked. Alas, in today's world, attempts to be spontaneous are often punished by banishment to shabby motel rooms, nearby smoke belching factories in high-crime districts. Anyway, NCC was quite a treat... I assume that most of you have already read the major scoops, so I'll try for a human interest story.

The somewhat clad model at a certain exhibit had won my heart by the second day. As it turned out, my hopes of meeting this lovely person went for naught. However I did try, part of which was buying (an overlong) lunch for two of the guys from that same exhibit. Now here's the picture, I only wanted a proper 'intro' to the pretty 'demo-lady', but all these guys would talk about was how they felt about whether some of the newest computers were mainframes', 'minis' or 'micros'. Give me a break! Some six years of reading hobbyist oriented publications has convinced me that that topic has been driven deeper into the earth than drill bits from an unproductive North Sea oil platform. (Most published benchmark results are in the same league). To grossly understate the case, I knew I was staring Baron Boredom (death-dealing Voodoo spirit) right in the eye.

Now usually I can cut tedious conversations short by pretending that I am violently ill. This time, it did not work. One of these guys used to work for his home state's Highway Patrol, where he taught emergency paramedical procedures. I mean really, where's a cop when you don't need one... Soon enough, I was listening to the old mainframe, mini and micro story again.

Looking back on this whole blasted episode, I am reminded of how the definition of 'full-size passenger cars' has changed to reflect the fact that they no longer make them. As relates to computers, I think a one line BASIC program can more than say it all. (In case it can't, superfluous parentheses have been added).

Class, class! Please come to order! (Do you remember that? Just what were they telling us?)

The program:
DF = (SC ^ 2)*(BS ^ 3)

The variables:
DF means 'Definition Factor'. Of course DF could stand for other things, but let's all act like grown-ups!

SC means 'System Cost' (and nothing else!).

BS is the 'Byte Size' (bit width of the accumulator. Don't even think it!!!

The formula explained:
DF is the product of SC squared and BS cubed. If DF is a relatively small value, it suggests a 'micro'; if DF is a huge value, it suggests a 'mainframe'; if DF falls somewhere between 'relatively small' and 'huge', it's probably a 'mini'. More to the point who cares?

If I had been the first to address this whole tedious topic, all computers would fall into one of two general categories:

1: The Unaffordables (what you can't afford-mnemonic U)
2: The Affordables (what you can afford-mnemonic A)

U2 MACHINES: Your buddies in the IRS and telephone company keep track of you with the U2 machines. Aren't you glad they do?

U3 MACHINES: A taste of Hell. Machines at this level keep track of our out of state parking tickets, summon us to jury duty and screw up our bills and bank statements.

U4 MACHINES: Look for U4 equipment when you travel abroad. They can complicate your life like the more familiar U3 models, but they will take much longer to to do so and are more prone to gross error. Some U4 machines misbehave due to primitive software, others fail because some adolescent third-world guerillas have clobbered a major power generator. It doesn't really matter why U4 machines flame out, just take it as a matter of faith that they do. When U4 software is at the 'trailing edge' of very bad, it's anyone's guess who'll be the loser. True story: A few years back, I spent way too much during a long and deluxe European vacation. Some two years later, it dawned on me that some umpty thousand dollars worth of overseas charges would never appear (on my bills from a major credit card company) unless of course I chose to bring it up. A major test of my character if ever I saw one.

THE 'A' MACHINES

A1 MACHINES: Here we have state of the art eight bit systems; as in the best stereo systems, separate components are often the tip-off. Also, look for multiple user options, bus compatible cards from several manufacturers (e.g. S-100 or STD), extended memory addressing and/or bank select options. No matter which CPU is involved, it is generally being run fast enough to screw up the neighbor's TV reception while also irradiating its user. Ironically, most of these eight bit beauties cannot draw detailed pictures of the Starship Enterprise (which many lesser systems draw with ease).

Buy an A1 machine. Even better, learn how to use it; you just might end up with a reliable management tool suited to your own small business. If not, don't get down on yourself, at least you showed good acumen for a [well-heeled] hobbyist.

(continued next page)

A2 MACHINES: Like the A1 machines, but here some corners have been cut. You will find these wherever hardware geniuses hang out. A2 items are functionally identical (or perhaps better than) A1 units, but are rarely seen in public. This is because everything is interconnected with unmapped wires and alligator clips. Some of America's most innovative technology exists within the A2 category, but don't count on ever seeing any of it. The people who construct these unique and delicate masterpieces have no time for new friends, especially you!

Obviously, some 'Affordable' machines would belong in even lesser categories, but I have already written threadbare computers, and in keeping with my policy of trying to write a class column, I refuse to discuss them any further.

Three weeks later, how about some old business... You may recall that in the May issue I discussed Intertec's [non] service policies for certain equipment which they used to make, and that Mr. Wells of Intertec (sort of) replied. I say 'sort of ' because I think Mr. Wells missed my point. I was questioning their policy of refusing to offer factory service on [recently] discontinued products. As far as I'm concerned, there is only one good reason for a company to turn its back on products less than five years old, that one good reason being Chapter XI.

On page 65 of the June '81 Byte is another full page, full color ad from Intertec. Oddly, they mention a "new QD model", and go on to say "QD users have been impressed by the inherent reliability of the system". I don't get it; is the QD new or are there legions of satisfied users? This ad goes on about service with a "common screwdriver" and "total commitment [sic] to product and customer support" (see my May column). The beat goes on...

On page 20 of Computerworld (May 18,'81) is a summary of Datapro's 'User Ratings of Computer Systems' (1981 part III). It sure seems to vindicate my last excursion into the land of hardware tips and caveats. (For $25.00 you can buy the whole report direct from Datapro Research Corp., 1805 Underwood Blvd., Delran, N.J., 08075). If $25.00 is too big a bite, and if the May 18 Computerworld cannot easily be found, I'll tell you this much: Seventeen desktop microcomputer systems were rated; guess which one was rated dead-last!

Enjoy the Nice Weather,
Zoso

CPMUG Volume 51

DESCRIPTION: STAGE2 MACRO PROCESSOR

By: Dick Curtiss

NO.	SIZE	NAME	COMMENTS
		CATALOG.051	CONTENTS OF CP/M VOL. 51
51.1	2K	ABSTRACT.051	Overview of volume
51.2	16K	ALX.S2M	Assembly Lang. Extension macros
51.3	3K	ALX-.DOC	Doc on above
51.4	3K	ALXTEST.ALX	Sample macros to test above,
		ALXTEST.ALX	including errors to be detected.
51.5	4K	CIO.ALX	Console I/O routines in ALX
51.6	2K	CRCK.COM	K. Petersen's program to CRC check files
51.7	1K	CRCKLIST.CRC	Contains the CRCs for all files on this disk. (except itself)
51.8	2K	DEMO.S2M	Interactive STAGE2 demo macros.
51.9	17K	DISKIO2.SRC	(*) disk I/O for STAGE2 processor
51.10	6K	DISKIO2-.DOC	(*) DOC on above
51.11	5K	FLD1.DAT	(*) Sample data for FLUB test
51.12	4K	FLD2.DAT	(*) "
51.13	13K	FLT1.FLB	(*) Sample program for FLUB test
51.14	11K	FLT2.FLB	(*) "
51.15	1K	FLUB$.SUB	(*) Submit file for FLT1,FLT2,STG2.
51.16	4K	FLUB8080.S2M	(*) Macros to turn FLUB to 8080 asm
51.17	3K	HELP.DOC	First-timer's "to do" list.
51.18	2K	IMPL.DOC	(*) Notes on implementing STAGE2
51.19	2K	INTERACT.S2M	Another macro demo.
51.20	22K	INTRO.DOC	Info to "read" STAGE2 macros, and with diligence, write them.
51.21	16K	IOOP.SRC	(*) I/O processor 8080 source
51.22	1K	IOOP$.SUB	(*) submit file for macro pass + asm of above.
51.23	1K	MEMORY.INP	Another macro demo.
51.24	2K	ST2T.DAT	(*)
51.25	12K	STAGE2.COM	The executable macro processor itself.
51.26	48K	STG2.FLB	(*) Source for STAGE2 in FLUB code
51.27	5K	STG2MATH.ASM	(*) STAGE2 support routines
51.28	5K	STG2SUP.ASM	(*) "
51.29	8K	TERM.ALX	Sample 8080 terminal program w/ALX macros.
51.30	3K	TERMSUP.ASM	Subroutines for TERM.ALX
51.31	2K	USE.DOC	How to execute STAGE2.
51.32	6K	VDB.ALX	TDL video driver in ALX, a "state machine".
51.33	5K	VOLUME51.DOC	Dick Curtiss' own excellent documentation.

(*) Files are necessary only to change STAGE2.

A Tutorial on Volume 51

by Ward Christensen

I have just completed cataloging volume 51. Richard Curtiss of Seattle, WA, implemented the STAGE2 macro processor which is described in the book: "Software Tools for Non-Numeric Applications" by Wm. M. Waite available from Prentice-Hall, Inc., Rt. 59 at Brook Hill Dr., P.O. Box 505, West Nyack, NY 10994. The price is $24.95 plus tax and handling.

Richard did an excellent job, both in the implementation, and in the documentation.

WHAT IS A MACRO PROCESSOR?

The first question might be: "What is STAGE2?". The answer: "A macro processor". Feel free to skip to the heading "What is STAGE2?" if you know what a macro processor is. Let me give the rest of you an overview.

Assembly language programmers on large mainframe computers frequently use MACROS to avoid learning and implementing detailed instructions and data formats which ask the operating system to perform a certain function. The most frequently used instructions aid input and output on files. For example, as in CP/M you must open a file before you can read from it or write to it. In order to open a file in CP/M, you might typically code the sequence below. (I have included all EQUates to make the example complete.)

```
FCB  EQU  5CH  ;SYSTEM
OPEN EQU  15   ;FNC #
BDOS EQU  5    ;ADDR
     .
     .
     .
     LXI  D,FCB
     MVI  C,OPEN
     CALL BDOS
```

These instructions will cause a directory search of the disk to find the file whose name is stored in the file control block (FCB).

I have written some simple macros for CP/M, to save writing quite so much code. For example, a generalized CPM macro takes two operands:

1-a function, like OPEN in the above example.

2-a value to be passed to BDOS. This might be the address of the FCB for the file to be opened, the DMA address to be set, etc.

By simply defining a set of equates and a CPM macro, I can do things quite "nicely", or more importantly, "readably".

(Should you be interested in the macro, I have included it at the end of the article. See footnote 1.)

How is this macro used? For example, the file open used before now becomes simply:

```
CPM  OPEN,FCB
```

If I wanted to write one sector to a new file, I'd have to erase the file, make it, set the DMA to the buffer, write, and close it. This would require quite a few 8080 instructions. With macros, this becomes:

```
CPM  ERASE,OUTFCB
CPM  MAKE,OUTFCB
INR  A    ;0FFH = ERROR
JZ   ERROR
CPM  SETDMA,OUTBUF
CPM  WRITE,OUTFCB
CPM  CLOSE,OUTFCB
CPM  SETDMA,80H
```

(The final SETDMA to 80H is a good practice, the reason being: if you are running under SUBMIT in CP/M 1.4, and have a program which returns to CP/M without resetting the DMA address to 80, the next command in the SUB file will not be processed properly.)

I've gone into a lot of detail on a specific macro application. Let's generalize. The Waite book says that a group of macros is like an abstract machine. This machine is like the real computer but has a larger set of operations. The individual macros are operations of the abstract machine. They are expressed in assembly language and the macro assembler translates.

For example, when I coded the macro:

```
CPM  OPEN,OUTFCB
```

the macro processor (Digital Research's MAC in this case) replaced the macro with the 8080 assembly instructions:

```
LXI  D,OUTFCB
MVI  C,OPEN
CALL BDOS
```

because that is what the CPM macro I wrote told it to do.

In this example, OPEN, and OUTFCB, were parameters passed to the macro. The actual macro itself didn't contain:

```
LXI  D,OUTFCB
```

because then it could only do things to OUTFCB. Instead, it contained:

```
LXI      D,?P
```

where ?P is the "symbolic" name for the second parameter passed via the CPM macro.

Macros can be much more complex than this, and can deal with much more than simply assembly language programming. Let's look at STAGE2 for some examples.

WHAT IS STAGE2?

STAGE2 is a general purpose macro processor. As such, it can handle text other than assembly language. An example, written by Richard Curtiss in STAGE2, is used in conjunction with a master catalog as maintained by my UCAT program. Dick occasionally renames a disk, so that now and then two disks appear in MAST.CAT with the same number, but with different names. I ran his program on my MAST.CAT just for the fun of it, and was surprised to find I had two disks with the same number. It was an oversight

(continued next page)

on my part, and thanks to his program, I learned about it and can correct it.

Text which is to be processed by STAGE2 is most easily handled if each "chunk" of it is on one line. This would make translation of random English text, for example, quite difficult. In the case of processing MAST.CAT this is no problem, since MAST.CAT consists of various lines. At the front of the file is the list of names not to be catalogued; the form follows:

```
(ASM.COM
PIP.COM
STAT.COM)
```

Following this is the list of filenames and disk names and numbers:

```
ALPHA.ASM,ASM-A-I.841
ALPHA.COM,SYSRES.80X
etc.
```

Thus the macro processor at least must be able to tell the difference between the list at the front, of names not to be catalogued, and the list of names which have disks associated with them. It uses "pattern recognition" to find a match. A line which contains a disk is unique in that it contains a comma. You can consider it to consist of

```
$,$
```

where "$" represents some string of text. Specifically, the first $ means the filename.filetype, and the second number the diskname.disknumber.

We could then look in more detail at the second parameter, and consider it to be of the form:

```
$.$
```

i.e. the volume name, then a period, then the volume number.

These two forms, "$,$", and "$.$, are actually STAGE2 macro "prototypes". A "prototype" is a macro language term, meaning loosely "a picture" showing the item you want to match.

The STAGE2 command line is like a Microsoft MACRO-80 command line, i.e.

```
STAGE2 destination,list=source
```

"source" typically consists of a macro library, then a comma, then the file of text to process. Thus, to scan MAST.CAT for duplicates using STAGE2, with the DUP.S2M macros, (S2M means Stage 2 Macro, but may be any name), type:

```
STAGE2 NEW.CAT,CON:=DUP.S2M,
   MAST.CAT
```

NOTE the MAST.CAT is immediately following the ...S2M, but these narrow columns couldn't hold it.

Let's look at the DUP macro itself. BEWARE: it is very "cryptic", but if you bear with me, you'll understand it. The complete macro is shown in FIGURE 1. I'll explain, hopefully, enough to help you understand it.

NOTE that Richard's INTRO.DOC file is sufficient to help you read, and eventually write, your own macros. What I am doing is adding some more details on examining a single macro, to help save you a bit more time.

The line that starts out with "#$%!0 (+-*/)" is NOT someone swearing at you, but rather tells STAGE2 what delimiters you are going to use in the macros that follow.

To see how the delimiters are used, let's look at the structure of a file of STAGE2 macros. Each macro in the file consists of:

```
a prototype line
one or more macro body lines
an end of macro line
```

In the line

```
#$%!0 (+-*/),
```

the characters have the following meanings:

```
# end of prototype line
$ match any text
% end of macro body line
! escape character for parameter conversions and functions
0 (+-*/) are the characters to be used for zero, space, left parenthesis, addition, subtraction, multiplication, division, and right parenthesis.
```

This makes STAGE2 immediately appear "cryptic", but consider it instead as "flexible"--since you can choose any characters so they don't conflict with the characters in the text you are processing.

Richard Curtiss has recently taken to using the accent grave and the "not" sign (ASCII 60H and 7EH) since he can "standardize" on them, because they would rarely appear in anything being processed. Also, since the "match any text" character, and the "parameter escape" character can never occur on the same line, nor can the two characters "end of prototype" and "end of macro body", these four characters may be represented by two.

Let's look at the first prototype line of the DUP macros:

```
$,$#
```

This matches any line which contains a comma. The # says this is the end of the prototype line, so comments may follow.

The next line:

```
_VOL !20%
```

_VOL is a call on the _VOL macro, passing it the volume name.number. The name.number is represented by the string "!20". The appearance of "!" followed by a number from 1 to 9, means you are requesting a parameter; in this case, parameter 2, the disk name.number. The "0" following the "!2" is a parameter conversion request. Conversion 0 in STAGE2 simply copies the parameter to a line which STAGE2 is building called the CURRENT LINE, or CL, which will be output, or used for further processing.

The final line in the $,$ macro is a lone % sign, which means "end of macro".

The _VOL macro then follows. Its prototype line is:

```
_VOL $.$#
```

Thus its first parameter becomes the volume name, and the second becomes the volume number. The next line:

```
_IF !21= SKIP 3%
```

is a call on yet another macro, _IF. The !2 means parameter 2, but now, !21 means parameter conversion 1, which is a very sophisticated "look the symbol up in the symbol table". STAGE2 has the immense power of allowing you to store arbitrary strings in a symbol table, using other arbitrary strings as their key. You could as easily store the character string "1" under the key "ONE", as you could store the string "Richard Curtiss" under the key "STAGE2 AUTHOR".

In this case, the !21 is looking up the disk serial number to see if it is already known. The "= SKIP 3" says if it is "nothing" (i.e. there is a space after the = sign) then skip these 3 macro instructions:

```
_IF !21=!10 SKIP 1%
_DUP !20,!21,!10%
_!F9%
```

The first line says: if parameter 2 looked up in the symbol table (!21) is equal to parameter 1 taken as is (!10) then skip one line. To see why this test was made, let's look first at the next line, the one starting with _DUP.

This line calls the _DUP macro, passing it 3 parameters: !20 which means the literal second parameter (the volume number), !21, which means the volume name stored in the symbol table under that number, and !10, the new (duplicate) name given to the same volume number.

The final line, !F9, is a new idea: There are 10 system FUNCTIONS. Each is indicated by the escape character ("!" in this case) followed by the letter F, followed by a function number, from 0 to 9. Function 9 says to "exit the macro".

Finally, the line:

```
_STO !20=!10%
```

is executed, IF we took the "SKIP 3" previously. _STO is yet another macro, which places its second operand (the one after the "=") into the symbol table, under the key of its first operand (the one before the "="). This line is using !1 and !2, i.e. the first and second parameters in the prototype, and is further doing parameter conversion 0 on them, i.e. copying them literally (no processing).

Let me explain just a few more line which do things we haven't seen before. The line:

```
!10.!30!46%
```

shows several new STAGE2 ideas. !10 and !30 are simple parameters 1 and 3 taken literally. The "." is simply a literal period character. !46 uses a new parameter conversion: 6. Conversion 6 says to take the current line (CL) (in this case, parameter 1, the period, then parameter 2), and put them back as parameter 4. The CL is then cleared to null.

The line (which I'll have to wrap around two lines since it's too wide to fit in this narrow column):

```
DUPLICATE (!10) (!20)
(!30)!F15%
```

says to build up the CL substituting parameters 1, 2, and 3.

The !F15 says to then perform system function 1 (output the CL). The "5" is expected by system function 1, to tell it what "channel" to output to. Channel 5 is specified in the STAGE2 command, and is usually the console, or a disk file.

A digression: Do you see how parameters substitute? If so, skip down to the centered line of dashes.

Let's say the _DUP macro were called via:

```
_DUP 871,CBBSBIO,CSRC
```

The _DUP prototype looks like:

```
_DUP $,$,$#
```

which means it "recognizes" the character string "_DUP ", followed by 3 character strings, separated by commas.

Since parameter conversion 0 means "the literal parameter", then within the _DUP macro:

THIS	Means this string
!10	871
!20	CBBSBIO
!30	CSRC

Thus the line

```
DUPLICATE (!10) (!20)
(!30)!F15%
```

after parameter conversion, looks like:

```
DUPLICATE (871) (CBBSBIO) (CSRC)
```

I have not gone through the entire macro, but hope, if you read the INTRO.DOC file, and look over what I have said, you'll be able to read the DUP macros.

STAGE2 is actually written in a machine-independent language specifically designed for the purpose of transporting STAGE2 to other machines. This language is FLUB. The entire FLUB source for STAGE2 is on this disk, as well as the necessary support files (written by Dick) to bring it up under CP/M. Also supplied is a .COM file, so you can begin using STAGE2 immediately. It would be possible to put up STAGE2 on another processor, given only that you have an assembler for that other processor, and the ability to write native support packages for the 16 bit arithmetic, file I/O, etc.

CRC CHECKED FILES

This volume continues the practice started on volume 50, of having a file (CRCKLIST.CRC) of the CRC (Cyclical Redundancy Check) values for each file on

(continued next page)

the disk. This will enable you to determine if a file on your copy of the disk is valid, particularly if it has gone through many copying cycles, possibly including being MODEM transferred at some point. Type:

CRCK filename.filetype

to find the CRC for a particular file. Compare this to the CRC values stored in CRCKLIST.CRC.

When you get a disk, you might want to turn on your printer (via ©P) then type:

CRCK *.*

to get CRC values for every file on disk. Then compare it to the stored values.

Thanks to Keith Petersen for allowing us to include this program on the Users Group disks, and to RBS who added big buffering, and shortened the output a bit.

Incidentally, the file of CRCs, CRCKLIST.CRC, was created by typing:

CRCK *.* F

DON'T type that on your disk unless you have saved the original CRCKLIST.CRC (by PIPping it, RENaming it, etc).

CPMUG-AUTHORS, CONTRIBUTORS, AND REVIEWERS

STAGE2 was an excellent contribution. But we need more. Contributions need not be a complete volume, the way STAGE2 was. Anything from a single program on, is welcome. Just send it to CPMUG, 1651 Third Ave., N.Y., N.Y. 10028.

AUTHORS: CONTRIBUTE!

We specifically encourage AUTHORS to contribute their own programs. Do you have something you have shared with friends? Please share it with us. Just put a little .DOC file with it, and send it off.

USERS: CONTRIBUTE!

So, a friend wrote a program, which you really enjoy using, eh? Why not ask your friend if you may share it with CPMUG? You might even write up a paragraph or so telling everyone what you think of it. It becomes a ready-made abstract, which will help us get it out sooner.

REVIEWERS: CONTRIBUTE!

We can also use REVIEWERS. If you would like to offer to review and abstract some programs, write me (or Jim Mills) in care of CPMUG, at the above address. Do as little as one program, or a whole disk. It is appreciated.

FOOTNOTE 1: The CPM Macro for Digital Research's MAC:

```
;
;DEFINE CP/M MACRO -  CPM fnc,parm[,NOSAVE]
;       All index registers are automatically saved
;
;       fnc     May be any BDOS function
;       parm    is what is to be loaded into DE
;               before calling BDOS
;       NOSAVE  any non-blank third operand causes the
;               register saves to NOT be generated.
;
;Ex:    CPM     OPEN,FCB,NOSAVE
;
;the double ";;" means don't store these comments in
;memory as part of the macro definition.
;
CPM     MACRO   ?F,?P,?N        ;;3 OPERANDS
        IF      NUL ?N          ;;IF "NOSAVE" IS NUL,
        PUSH    B               ;;SAVE
        PUSH    D               ;;      ALL
        PUSH    H               ;;      INDEX REGS
        ENDIF                   ;;END OF "IF NUL ?N"
%!      IF      NOT NUL ?F      ;;WAS THERE A FNC?
        MVI     C,?F            ;;      YES, MOVE TO C
        ENDIF                   ;;END OF "IF NOT NUL ?C"
        IF      NOT NUL ?P      ;;WAS THERE A PARM?
        LXI     D,?P            ;;      YES LXI TO DE
        ENDIF                   ;;END OF "IF NOT NUL ?P"
        CALL    BDOS            ;;EXECUTE THE FNC
        IF      NUL ?N          ;;RESTORE THE REGS
        POP     H               ;;      IF THEY
        POP     D               ;;      WERE
        POP     B               ;;      SAVED
        ENDIF                   ;;END OF "IF NUL ?N"
        ENDM                    ;;END OF MACRO
```

```
;
;BDOS EQUATES FOR USE WITH CPM
;  MACRO AND FOR GENERAL USE
;
RDCON   EQU     1       ;READ A BYTE FROM CONSOLE
WRCON   EQU     2       ;WRITE CHAR IN E TO CONSOLE
PRINT   EQU     9       ;PRINT "$" TERMINATED STRING
RDCONBF EQU     10      ;READ CONSOLE BUFFERED
CONST   EQU     11      ;CHECK CONSOLE STATUS
OPEN    EQU     15      ;OPEN A FILE
CLOSE   EQU     16      ;CLOSE A FILE
SRCHF   EQU     17      ;FIRST DIRECTORY SEARCH
SRCHN   EQU     18      ;NEXT DIRECTORY SEARCH
ERASE   EQU     19      ;ERASE A FILE
READ    EQU     20      ;READ A SECTOR FROM DISK
WRITE   EQU     21      ;WRITE A SECTOR TO DISK
MAKE    EQU     22      ;MAKE A NEW FILE
REN     EQU     23      ;RENAME A FILE
SETDMA  EQU     26      ;SET THE DMA ADDRESS
;
BDOS    EQU     5       ;BDOS ENTRY JMP ADDR
FCB     EQU     5CH     ;DEFAULT SYSTEM FCB
FCB2    EQU     6CH     ;SECOND NAME IN FCB
FCBEXT  EQU     FCB+12  ;THE EXTENT BYTE
FCBRNO  EQU     FCB+32  ;THE RECORD NUMBER TO READ
```

F I G U R E 1.

STAGE2 MACRO TO DISPLAY DUPLICATE VOLUME NUMBERS IN MAST.CAT

NOTE this file is NOT on volume 51 because the volume was 100% full

```
#$%!0 (+-*/)    ;DUP: STAGE2 macros for deleting duplicate MAST.CAT names.
$,$#                          ;match "filename,diskname"
 VOL !20%                     ;pass parm 2 on to .VOL for testing
%  ================
 VOL $.$#                     ;VOL tests for duplicate volume #'s
 IF !21= SKIP 3%              ;Look up the volume #. If " ", skip 3
 IF !21=!10 SKIP 1%                  ;IF name prev. stored = current, skip 1
 DUP !20,!21,!10%                    ;Pass the # and 2 names to DUP macro.
 !F9%                                ;Terminate macro expansion
 STO !20=!10%                 ;Store the volume name, under the vol #.
%  ================
 DUP $,$,$#                   ;DUP parms are: #, first name, second name
 !10.!30!46%                  ;Store "number.name" back into parm 4
 IF !41-= SKIP 2%             ;Replace "number.name" with its looked-up val.
 DUPLICATE (!10) (!20) (!30)!F15% ;output DUPLICATE number name name to console.
 STO !40=^%                   ;store under "number.name" a "^".
%  ================ Following are "convenience" macros, usable in any prog.
 STO $=$#                     ;store second parameter under key of first.
 !F3%
%  ================
 IF $=$ SKIP $#               ;skip on parm 1 = parm 2.
 !F50%
%  ================
 IF $-=$ SKIP $#              ;skip on parm 1 not = parm 2.
 !F51%
%  ================
 $#                           ;this matches everything else and passes it
%  ================
 ($#                          ;this matches unpaired paren at start of file
%  ================
 $)#                          ;this matches unpaired paren at end of file
%%                            ;this ends the macros.  MAST.CAT comes next.
```

F I G U R E 2.
SAMPLE EXECUTION OF "DUP.S2M" STAGE2 MACROS

```
A>b:stage2 con:,nul:=b:dup.s2m,mast.cat
-116-D24-S31-M32-F32-  STAGE2  --  8080 VERSION
DUPLICATE (3-4) (CBBSASM) (CBBSMIS)
DUPLICATE (REL) (CBBSASM) (CBBSMIS)
DUPLICATE (871) (CBBSBIO) (CSRC)
```

I executed STAGE2, with output coming to the console (CON:), list output being thrown away (NUL:), and with input coming from b:dup.s2m followed by mast.cat itself.

I knew in advance that CBBS versions 3.4, and the released versions (REL) would be duplicates - that was intentional. However, finding that I had two disks with serial number 871 (CBBSBIO.871 and CSRC.871) was a total surprise. (No, I don't have 871 disks. I start the volume numbers of all my 8" disks at 800).

MDBS, Part 1: The Database Manager

by Harris Landgarten

MDBS refers to a family of database management products which follow closely in the footsteps of traditional large computer counterparts. The family consists of several generic products, each available for use with a growing assortment of host languages in either Z80 or 8080 versions. (The concept of host languages will be fully explained later in this review.) The MDBS product line is also selectively available in non-CP/M operating environments. This type of data base system is often referred to as being CODASYL-like but a more fitting classification would describe MDBS as a NETWORK type database manager. The emphasis of this review will be on the Z80 versions of the full database manager (MDBS), the query language-report writer (QRS), the dynamic restructuring utility (DRS), and the logging system (RTL), all running under CP/M using PL-I/80 as the host language and PLINKII as the system linker. Many consider this to be the optimum MDBS configuration. In conjunction with this review, I will explain how you would develop a simple order entry/accounts receivable/inventory system with this software.

Basic Concepts

The heart of the MDBS system is the data base manager itself which consists of two modules: the DDL (data description language), a stand-alone interactive program which is used for the initial definition and initialization of the application's database; the DMS (data manipulation system), a function library of MDBS commands in a form which can be called from the host language. In this case the DMS is supplied as a REL library which must be linked together with a compiled PL/I-80 application program. DMS functions are accessed from within an application by calling a common DMS entry point with the appropriate command codes as parameters. This scheme makes it seem as if the MDBS data management commands were part of the application language, hence the term host language. As I mentioned previously, MDBS can be supplied for interface with many different host languages. It is advantageous to use a host language which has internal data structure definition and overlay capabilities. The former attribute will simplify the data communication interface between the program and the database and the latter will allow the development of substantial applications in severely constrained memory space. Languages such as PL/I, COBOL, Pascal, and C are ideal, especially when used with an overlay linking loader such as PLINKII (see Vol II, no.1). In particular, I warn against the use of BASIC as a host language. The combination of no data structures, no literal parameter passing, and restricted overlay capabilities makes most applications either unnecessarily cumbersome or impossible.

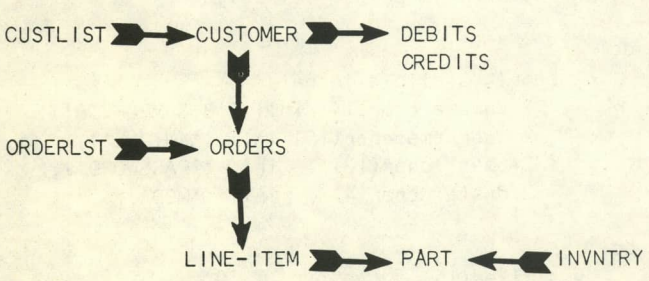
As mentioned earlier, the DDL program is a stand-alone, interactive program whose use mimics that of an interpreter. It is delivered configured for a 48k machine that uses a 64 character wide display. While this default is usable for all but the largest applications, it is suggested that you reconfigure the program for your memory size and terminal width. Unfortunately, the only way of accomplishing this is by patching such information as the address of the highest memory available to the program in hex; the patching into appropriate addresses should be effected with DDT. This procedure pre-supposes that the user has the ability to figure out such facts on his own with no help from the documentation; an assumption which I feel is unreasonable since many competent application programmers who could use MDBS are unfamiliar with such low level system constants. Furthermore, all of the configuration constants could either be set automatically or by user query with a trivial installation program.

The database consists of three types of units: the RECORD, the DATA-ITEM, and the SET. The DATA-ITEM is the smallest named unit of this system and can be any of the following DDL types (corresponding PL/I-80 types are in parentheses): BIN 1 (fixed binary 1-7 or Bit 1-8), BIN 2 (fixed binary 8-15 or Bit 9-16), CHAR n (Char(n) or Char(n-1) varying), IDEC n (fixed decimal n.x), REAL (floating point binary). Records are the basic units of the database and are composed of zero or more DATA-ITEMS. RECORDS are analogous to records in a file with DATA-ITEMS corresponding to fields within the record. In our sample application, the CUSTOMER is an example of a RECORD with DATA-ITEMS of NAME, ADDRESS, CITY, etc. A SET is a named relationship between RECORD types and is the basic structural unit of the database. Each set must have one or more owners and one or more members. For example the SET ORDERS links a CUSTOMER with his open ORDERS. These concepts will be explained more fully in our example.

In our sample database, as diagrammed in figure 1, the RECORD types are CUSTOMER, ORDER, LITEM, PART, DEBIT, and CREDIT, which stand for customer, order header, line item, parts, and AR debits and credits respectively. SETs define the relationships between these RECORDs. Each CUSTOMER may own many ORDERs; each ORDER may own many LITEMs; each LITEM may own a PART; each CUSTOMER owns his collection of DEBITs and CREDITs. Furthermore, I defined three SYSTEM owned SETs, those which are not owned by other RECORDS, to facilitate keyed access to CUSTOMERs, ORDERs, and PARTs by account number, order date, and part number respectively. These SYSTEM SETs also cause the system to maintain a customer list, order list, and inventory list in the specified sorted order. As you can see from listing 1, SETs can be kept in a variety of orders. SORTED by key, FIFO (first in first out), LIFO (last in first out), PRIOR (each new entry is inserted before the

currently selected RECORD), NEXT (each new entry is inserted after the currently selected RECORD), and IMMAT (immaterial) orders can optionally be specified. SETs which are SORTED can be quickly accessed by the sort key. Several different SETs with the same owner and member but different sort keys can be used for multi-keyed access. Other DDL statements allow for storage specifications. A database may span all available drives with the system automatically allocating and de-allocating space as needed. An elaborate security system is also provided. A user must supply his previously assigned password when opening the database. In addition each user is assigned read and write access levels of 1 to 255. SETs, RECORDs, and DATA-ITEMs can each have individually assigned minimum read access and write access levels. MDBS will not allow anyone to violate a data area with a higher minimum access level than his. This system allows different users to selectively access different areas of the database on a need to know basis for read only or update purposes. These facilities allow very complex databases to be set up with a minimum of effort once the logical data design has been worked through.

Figure 1 Diagram of Sample DB



Listing 1 Data description for Sample DB

```
0100 FILES   B:SAMPLE.DB       1   512
0110 DRIVE   1   1000

0120 PASSWORDS
0130         ANYONE                255 255   PASSWORD

0140 RECORD CUSTOMER
0150 ITEM    NAME       CHAR 30
0160 ITEM    ACCTNO     BIN  2
0170 ITEM    ADDRESS    CHAR 25
0180 ITEM    CITY       CHAR 15
0190 ITEM    STATE      CHAR 2
0200 ITEM    ZIP        CHAR 9
0210 ITEM    PHONE      CHAR 12
0220 ITEM    BALANCE    IDEC 10

0230 RECORD DEBIT
0240 ITEM    DATE       IDEC 6
0250 ITEM    DESC       CHAR 12
0260 ITEM    AMT        IDEC 10
0270 ITEM    INVNUM     IDEC 5
0280 ITEM    PDATE      IDEC 6

0290 RECORD CREDIT
0300 ITEM    DATE       IDEC 6
0310 ITEM    AMT        IDEC 10
0320 ITEM    AMTLEFT    IDEC 10

0340 RECORD ORDER
0350 ITEM    PONUM      CHAR 12
0360 ITEM    ACCTNO     BIN  2
0370 ITEM    ORDDATE    IDEC 6
0380 ITEM    SHIPNAME   CHAR 30
0390 ITEM    SHIPADD1   CHAR 30
0400 ITEM    SHIPADD2   CHAR 30
0410 ITEM    SHIPADD3   CHAR 30
0420 ITEM    SHIPDATE   IDEC 6
0430 ITEM    STATUS     CHAR 1
0440 ITEM    VIA        CHAR 10
0450 ITEM    INVNUM     CHAR 5
0460 ITEM    TOTAL      IDEC 10
0470 ITEM    COST       IDEC 10
0480 ITEM    NLINES     BIN  1

0490 RECORD LITEM
0500 ITEM    QUANT      BIN  2
0510 ITEM    PRICE      IDEC 6

0520 RECORD PART
0530 ITEM    PARTNO     CHAR 10
0540 ITEM    DESC       CHAR 40
0550 ITEM    PRICE1     IDEC 6
0560 ITEM    PRICE2     IDEC 6
0570 ITEM    PRICE3     IDEC 6
0580 ITEM    INSTOCK    BIN  2
0590 ITEM    INPROC     BIN  2
0600 ITEM    ONORD      BIN  2
0610 ITEM    REORDER    BIN  2
0620 ITEM    COST       IDEC 6

0630 SET     CUSTLIST AUTO 1:N
0640                                SORTED   ACCTNO
0650 OWNER   SYSTEM
0660 MEMBER  CUSTOMER

0670 SET     ARLEGDER MAN  1:N
0680                                SORTED   DATE
0690 OWNER   CUSTOMER
0700 MEMBER  DEBIT
0710 MEMBER  CREDIT

0720 SET     ORDERS   MAN  1:N
0730                                FIFO
0740 OWNER   CUSTOMER
0750 MEMBER  ORDER

0760 SET     LINE     MAN  1:N
0770                                FIFO
0780 OWNER   ORDER
0790 MEMBER  LITEM

0800 SET     SORDER   AUTO 1:N
0810                                SORTED   ORDDATE
0820 OWNER   SYSTEM
0830 MEMBER  ORDER
```

(continued next page)

```
0840 SET     INVNTY    AUTO 1:N
0850                                SORTED    PARTNO
0860 OWNER   SYSTEM
0870 MEMBER  PART

0880 SET     PLINK     MAN  1:1
0890                                IMMAT
0900 OWNER   LITEM
0910 MEMBER  PART

0920 END
```

Before running the DDL program, it is necessary to create empty files (using the appropriate filenames) for the textual description of the database and for the database itself. Under CP/M, this can be done via SAVE 0 SAMPLE.DDL and SAVE 0 B:SAMPLE.DB for our example. Unless this is done before running DDL, you will find that you cannot save your database description after you have typed it in. Furthermore, your work cannot be recovered without the help of a monitor or front panel.

The reason for this seemingly foolish procedure dates back to the origin of MDBS, which was initially developed for North Star BASIC under North Star DOS. This operating system requires that files be declared prior to use, a procedure which is unnecessary under CP/M. MDBS claims that they have clung to this awkward procedure in order to maintain compatibility between the two operating systems, a weak excuse for poor human engineering in my opinion. After DDL is invoked, you can either read in a previously created file or type in a new one. The text entry system acts as if you were dealing with a keypunch machine; text lines are called cards, and strict column alignment is required.

To facilitate observance of the strict syntax requirements, DDL will draw various card masks on screen. For example, typing RE will cause the detailed format of a RECORD card to be displayed, under which you can easily align the proper information. This method of displaying helpful information makes a potentially awkward system easy to use. I recommend that the author consider added tab stops in the appropriate columns as a further aid to sloppy typists. One can't help but wonder if this DDL was adapted from a keypunch version. DDL also provides the usual array of line-oriented editor commands for inserting, deleting, editing and renumbering lines. The 'DDL' command causes the database description to be scanned, analyzed for errors; if no errors are found, the database is initialized.

At this point data may be entered into the database by application programs. It should be noted that database initialization destroys any live data that may have been in the database thereby making it impossible to change the database description of a live database without first unloading it. Database design alteration is necessary for any changes that are specified in the DDL including changing passwords. The DRS add-on deals with this problem of database maintenance.

The use of the DMS commands within an application program is straight forward. The complexity of the program varies with the difficulty of the application, but in all but the most trivial cases, MDBS eases the programming task considerably. The database manager users a page buffer algorithm for interface with the physical database on disk. The larger the page buffer, the faster the manager will operate. It is the responsibility of the application programmer to allocate sufficient buffer space for the database manager in his program. Listing 2 shows how the database is opened and space is allocated for buffers. This initialization routine allocates half of the available space to buffers.

Listing 2 - Opening the Database

```
init:
   proc;

   dcl
       dms entry (char(30) var) returns (fixed),
       dmsd entry (char(30) var, pointer) returns (fixed),
       setpbf entry (pointer,fixed),

   dcl
       1 opnblk static external,
          2 dbname char(14) init ('B:SAMPLE.DB'),
          2 username char(16) init ('ANYONE'),
          2 pass  char(12)    init ('PASSWORD'),
          2 rwstat char(4)    init ('MOD');

   dcl
       wds fixed(15),
       err fixed(7) external,
       maxwds returns(fixed(15)),
       allwds entry (fixed(15)) returns (pointer);

   wds=maxwds()/2;
   call setpbf(allwds(wds), wds*2);

   put list ('Allocating',wds*2,'bytes to DMS');

   err = dmsd('open', addr(opnblk));
   if err<>0 then
       do;
          put list ('can''t open - error ',err);
          stop;
       end;

end init;
```

Suppose that we wished to place an order for a customer consisting of 5 of part number A172 and 10 of part number B1000. The logic is outlined in listing 3. All commands are executed by calls to DMS using the command string as a parameter. Commands that transfer data to or from program data

structures call the entry point DMSD and give the ADDR of the appropriate data structure as an additional parameter. There are commands for adding, deleting, finding, transversing, transferring different amounts of data between program and database, and returning run statistics. Each command returns an error code to the calling program upon completion allowing easy error trapping.
Even though the DMS provides powerful tools, the real power stems from the fact that the DMS is embedded in a host language over which you have total control. This leaves you free to manipulate the screen in any way you wish, to control the flow of information, and to format reports to your specifications using the normal PL/I-80 language facilities while using DMS to manage the data itself.

I have only two complaints with the logical PL/I-80 - MDBS implementation. The first stems from the fact that no separate provision is made for character varying strings; they are treated like normal character strings that are one byte shorter. However, that additional byte of a varying string is a string header byte specifying length. Therefore, if a character varying DATA-ITEM is specified as a sort key, the records will be sorted according the length of the key string rather than according to its value. This means that character varying strings cannot be used as key fields.

The second shortcoming refers to the handling of fixed decimal numbers. PL/I, like COBOL COMP 3, stores these numbers in ASCII form without decimal point and the compiler keeps track of an implied decimal point in accordance with the data declaration. FIXED (8.2) refers to an 8 digit number with an implied decimal point two digits to the right. MDBS does not in any way keep track of the implied decimal point, which, while not effecting application programs, confuses the QRS module; it thinks 2.02 is 202.

Listing 3 - Logic to enter an order

1. Ask for customer's account number.

2. Find customer using the command 'FMSK' (Find Member Sort Key).

3. Make the found customer the owner of the ORDERS set using the 'SOM' command (Set Owner based on current Member).

4. Ask for order header information such as PO number.

5. Create a new order header using the command 'CRS' (Create Record and Store). This causes the order to automatically be inserted into the order list (SORDER) in sorted order.

6. Add the order header just created to the ORDERS set that is owned by the current customer by using 'AMS' (Add Member to Set).

7. Make the order header owner of the LINE set using 'SOM'.

8. Find part number A172 using 'FMSK' on the INVNTRY set.

9. Create an LITEM record using the price retrieved from A172 and the quantity of 5 using 'CRS'.
10. Add this LITEM to the LINE set using 'AMS'.

11. Make this LITEM owner of the PLINK set with 'SOM'
12. Link LITEM to PART by added Part to the PLINK set with 'AMS'.

13. Repeat steps 8 through 12 for each additional line item.

System Performance

MDBS performed flawlessly within its limits. Those limits are a function of the amount of buffer space that the programmer allocates for use by the DMS. As would be expected, the more buffer space alloted, the faster a given application will run, and the larger an application can be before slowing down. What is not expected, however, is the way in which the system reacts as buffer space becomes scarce. You would expect a smooth roll off which was hopefully a function of N LOG N but at worst a linear function of N where N is the number of records in the database. Unfortunately, what happens is a very rapid breakdown in response time, as if the system were doing a binary search on disk. This means that the application programmer must be very careful in allocating buffer space with an eye towards the eventual size of the database. 4k of buffer is adequate for small applications, but large databases might need 10k or more for adequate response. Adding buffer space requirements to the 18k taken by DMS itself results in a total overhead of between 20k and 30k. This overhead must fit into the memory space of the computer along with your application program. If you're using MDBS with BASIC-80, you must also include the 26k taken by the interpreter, leaving very little for the application program. In the PL/I-80 environment, the programmer can use overlays to solve the problem.

PLINKII has been a great help in this regard. A skillful programmer should be able to allocate enough buffer space for any normal application with the use of overlays. How much is enough can only be found through prior experimentation with a particular application. The critical buffer break down point is a function of the database design as well as the number of records in the database. Solutions to the memory space problem are being worked on, and the next major release of MDBS, scheduled for September, will address these failings. In the meantime, most complex applications can be elegantly accomplished with care and skill.

(continued next page)

All of the other shortcomings of MDBS are dealt with by the additional add-on programs which will be reviewed in part two of this review. For those of you considering buying MDBS by itself, I will briefly outline those shortcomings.

Like most large computer DBMS's, MDBS maintains the entire database in one CP/M file. If anything happens to that file (and that includes such trivial things as not closing the database) the entire database becomes unusable and unrecoverable. The only user recourse is to replace the damaged database with his latest backup, and reenter the interim information. Needless to say, frequent backups are recommended. The RTL (real time logging system) addresses this problem by maintaining a interim disk log of transactions since the last backup; the log can be used to bring a backup up to date if necessary.

Another shortcoming of the system is the fact that you cannot change the logical design of the database without destroying live data. This mandates the downloading of all live data to ordinary disk files, and then restoring the data after the alterations have been made. Those who have performed this procedure know it is a job measured in hours. The DRS add-on (dynamic restructuring system) allows you to alter a live database without destroying data. The QRS add-on provides many worthwhile and time saving features. I feel that anyone seriously working with this system should invest in the additional utilities. Those who feel differently should be aware of the inherent limitations of the DBMS itself.

Conclusion

MDBS has made it possible for a skilled programmer to generate a very complex application in a matter of weeks instead of months. The forte of MDBS is the flexibility it provides the programmer. In fact, I can think of no application which could not be generated using this system within the limits of current hardware. I must warn you however, that the price for this flexibility is complexity that is probably beyond the novice user. This system will not generate any stand-alone applications without a program being written. Priced at $900, MDBS is clearly meant as an application development tool for programmers, and as that it clearly succeeds. Those writing custom business packages should definitely consider MDBS as an important tool.

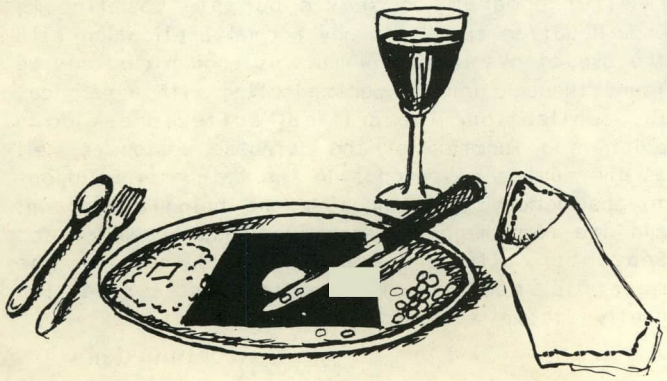


Table I

Package or version name:	MDBS Version 1.04 for PL/I-80 with Z80
Price:	$900.00
Systems available for:	CP/M, North Star DOS, Oasis, TRSDOS, Apple DOS (not all modules are available for each operating system).
Required supporting software:	Host language. Either BASCOM, FORTRAN-80, COBOL-80, PL/I-80, CBASIC, Pascal/M, Pascal/MT+, Pascal/Z, BASIC-80, Macro-80. (more are being developed)
Memory Requirements :	52K minimum.
Disk capacity required:	160k minimum.
Utility programs provided:	Relocation utility for making non 100H based versions of DDL. Sample programs.
Record size & type limits:	Types correspond to host language. Records may be of any size up to 65k. Practical limit 4k. Database size is limited by your hardware and operating system.
Portability:	Good portability between CP/M systems. Database can be unloaded under program control and written to disk in any format supported by the host language. QRS has addition output features. Databases are not generally portable between different implementations of MDBS or different host languages.
User skill level required:	Application programmer with some background in data structures. Not for the novice.

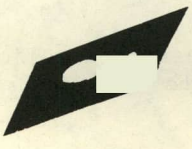


Table II
Qualitative Factors

Documentation	
organization for learning	6
organization for reference	6
readability	5
includes all needed information	7
Ease of use	
initial start up	3
conversion of external data	5
application implementation	7
operator use	*
Error recovery	
from input error	*
restart from interruption	1
from data media damage	1
Support	
for initial start up	4
for system improvement	4

* Depends upon the skill of the application programmer.

(Note that all error recovery depends upon proper backup procedures. The RTL module is designed to aid in error recovery.)

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

Table III
Data Management Capabilities

A. Underlying Data Model:

1. Data Types-alphanumeric, numeric
2. Relationships-one to many, many to one, one to one, many to many

B. Functions Provided:

1.a. Data dictionary maintenance-automatically maintained internally by system.

b. Data reorganization and conversion-must be accomplished by unloading the database, reinitializing with DDL, and the reloading. Data conversion is done by user with application program. DRS allows automatic reorganization.

2.a. Data entry and editing- user provides these functions in application program. Some functions in QRS.

b. Report generation- user provides this function in application program. Also done in QRS.

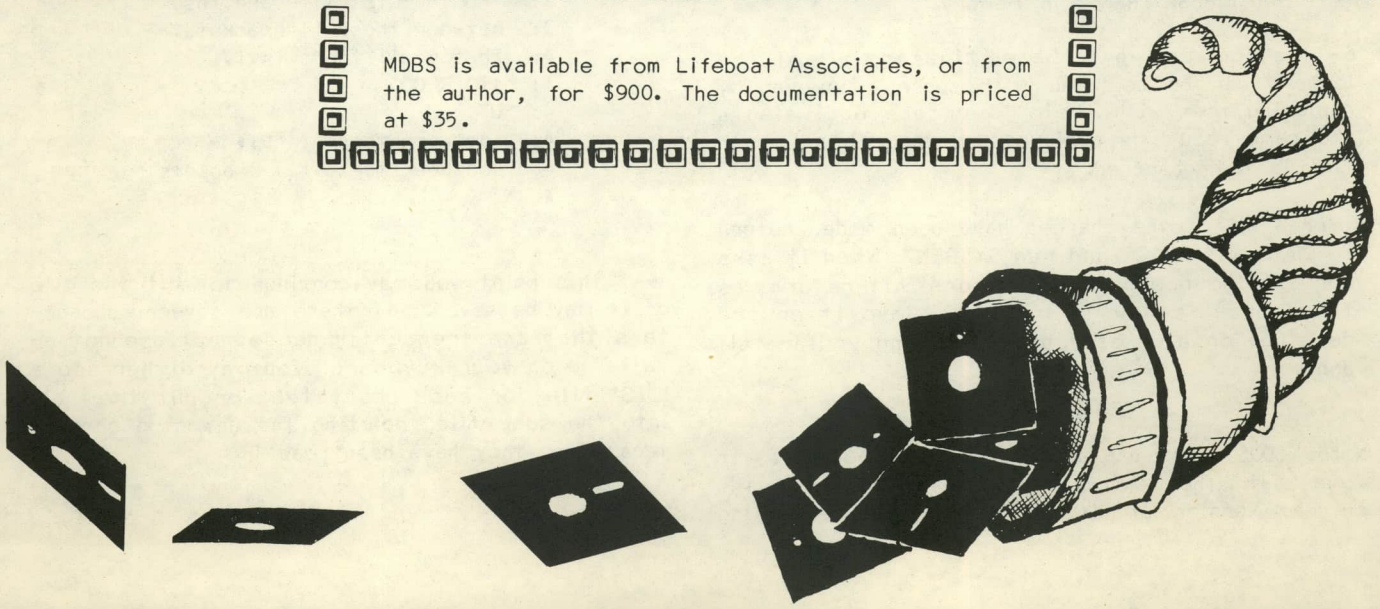
3.a. Data selection by predicate- Excellent support of complex predicates. Requires user skill without QRS.

b. Data joining & relating multiple data sets-powerful facilities provided.

c. Calculation on data- either by user program or QRS.

4.a. Data independent application interface- Database is fully independent.

MDBS is available from Lifeboat Associates, or from the author, for $900. The documentation is priced at $35.



Tips&Techniques

This month our winning tip comes from Bill Norris, of New York. Here it is:

"CP/M can be easily modified so that the intrinsic commands are renamed, eliminated or replaced with your own COM files. Some reasons for doing so are as follows:

1-Renaming the CP/M TYPE command to LIST, or the ERA command to KILL can help the beginner out by making the CP/M commands look more like those available in some higher level languages such as BASIC.

2-Faire exhibitors and computer stores have less to worry about if their demo systems no longer have the ability to ERAse or REName files.

3-You can make your 1.4 CP/M more closely resemble version 2.2. Example: One of the extended directory programs (from The CP/M Users Group or elsewhere...) can be invoked whenever the normal DIR command is given to CP/M.

To make the modifications proceed as follows:

1-Use SYSGEN or MOVCPM to place a system in memory, and then store it on disk with the save command.

2-Load it back into memory with DDT.

3-Examine memory with the D command until you see 'DIR ERA TYPESAVEREN' on the right side of the screen. This will be somewhere above 900 hex.

4-Notice that each command takes up four bytes. The DIR command is stored in the system as D,I,R,space.

5-Use DDT's S command to change the command, e.g. if you wanted to rename the DIR command to the single letter D, the IR bytes should be changed to spaces, (20 hex). If done properly, you will see 'D ERA TYPESAVEREN ' in memory.

6-If a command is to be eliminated, all of the letters can be replaced with spaces, however by replacing them with their lowercase equivalents the same result is achieved and the original location is not obscured.

7-When all of the changes have been made, return to CP/M with a ©C and run SYSGEN. When it asks for the 'source drive or return' hit return, as the source is now in memory. Save it on the destination disk of your choice and you are all done.

Note: CDOS users can use DEBUG instead of DDT. It comes with the Cromemco Macro Assembler. In the current version of CDOS (2.36) you will be looking for 'ATRIB.ATTR.BYE.DIR.ERA.REN.SAVE.TYPE.REN.'. The commands are variable in length and must be terminated with a null. The commands can be made up to 8 bytes in length subject to the following conditions:

1-Don't change their order.

2-The available space for commands is fixed, so while all commands may be readily shortened, if any are to be lengthened there must be a corresponding reduction in the length of another command.

3-The last command should be terminated with an FF (hex) after the null."

Bill also contributed a tip on getting untypeable characters into a file using 'ED.COM'. (**Bold** characters are keyboard input).

1-Run DDT and insert the hex equivalent of the required character(s), making certain that ^Z (1A hex) is the last character.

```
A>DDT
DDT  VERS 2.2
-f100,1ff,1a          ; Last char. is ^Z.
-s100                 ; Start to insert.
0100 1A 7B            ; '{' character.
0101 1A .             ; Finished.
-^C                   ; Return to System.
```

2-Save this as a file with extension 'LIB'

```
A>save 1 foo.lib
```

3-Edit your own file, and when the above character is needed, the 'READ' command will insert it for you as illustrated below:

```
*b4t
1:  This file will fetch a character
2:  from a 'LIB' file and insert it
3:  between these [] brackets.
4:  That's all there is...
1: *f[^ZRFOO
4: *OT
4:   between these [{*OTT
4:   between these [{] brackets.
4: *
```

4-At this point you may continue to edit the file or it may be saved. If there are several characters that are frequently needed but are not available on your keyboard, you may either use a 'LIB' file for each character, or put them all into the same file, deleting the unwanted characters after they have been read in.

ZSID Application Note

Digital Research has given us permission to print these instructions for changing the restart number ZSID uses for setting breakpoints and tracing program execution. The restart instruction must be inserted, followed by its vector location, in the ZSID.COM file at location 01EBH.

```
A>zsid zsid.com
ZSID VERS 1.4
NEXT  PC  END
2900 0100 nnnn
#sleb
01EB nn ef
01EC nn 28
01ED nn 0
01EE .
#^c

A>save 40 zsid.com
```

Users with serial numbers from ZS-00001 to ZS-00750 must insert the following code in the relocator module of ZSID in addition to making the changes described above.

```
A>zsid zsid.com
ZSID VERS 1.4
NEXT  PC  END
2900 0100 nnnn
#a103
0103 jp 1a0
0106 .
#a1a0
01A0 ld hl ,(01ec)
01A3 ld (1100) ,hl
01A6 inc hl
01A7 ld (1106) ,hl
01AA ld a, (01eb)
01AD ld (13b1) ,a
01B0 ld (13fe) ,a
01B3 ld (1f8d) ,a
01B6 ld (21a8) ,a
01B9 jp 013d
01BC .
#^c

A>save 40 zsid.com.
```

Licensed users may make these changes in ZSID software.

COBOL-80 Application Note

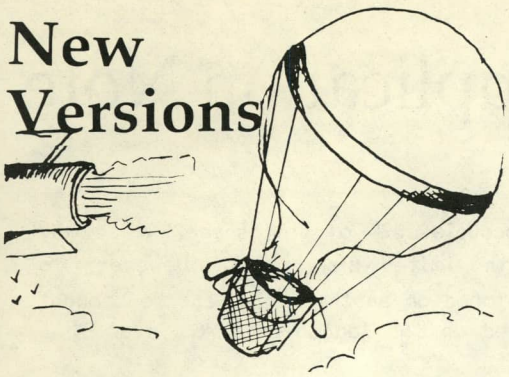
The Microsoft COBOL-80 Compiler occupies 82K of disk space, which may produce a hardship for users with limited disk space. However, to save space, the overlays can be stored on another drive and be brought in by the COBOL.COM program invoked on the logged-in disk.

In COBOL.COM , location 67CFh contains 00, which causes the logged-in drive to be searched for the overlays. If you are using drive A for your overlays, change 00 to 01. If you are using drive B, change it to 02. In relocated versions, the location is a8cfh.

Remember...

If your subscription began last August with Volume I, Number 3, you'd better renew right away--so you don't miss our next issue. The cut-off date for renewals is just around the corner. If you don't renew as soon as you read this notice we won't be able to get you into our August mailing. We hope you've held on to the reminder we sent. If not, enclose a mailing label from a copy of Lifelines. Let us know about any corrections in your address, but send that label! It will really help us process your order quickly.

New Versions



PAS-3 Medical Package
Version 1.75

This version effects improvements in the presentation of data on screen. Date range checking for the date entered in the Start program has been added, to catch operator errors.

In addition, the following bugs have been fixed:

1-The Query program no longer permits the system to enter a loop.
2-The Start program no longer echoes the Password. (This modification is only effective if the terminal recognizes the backspace character 08 Hex.
3-The Posting program has been fixed to recognize more operator errors.
4-A data/Mail merge typo bug has been remedied.

PLINKII
Version 1.08

This version contains two new features. Random disk I/O calls are now used when PLINKII is run under operating systems which support them. This normally results in a large increase in speed.

Whitesmiths' C Compiler REL file may now be linked and overlaid. The run time support libraries CLIB.A and MLIB.A, which must be present on disk, are searched automatically. The C program header CHDR.R is also included automatically in the program. The free memory symbol, _MEMORY is set to the .END. address. Not specifying the .REL file type with PLINKII is more convenient, as the program assumes that type by default.

These bugs have been eliminated with the new version:

1-OVERLAY LOAD DISK ERROR no longer appears when more than 35 overlays are used in a program; up to 255 overlays may be used now, as stated in the manual.
2-Files of the same name on different disks are now signalled by an error message, rather than being treated as separate files.
3-The FILE, SEGMENT, and MODULE statements now put segments of the program into sections as specified. Segments put into a .DATA section are now input order.
4-CONCATENATEd segments are no longer placed by default into a .DATA. section.

T/MAKER II
Version 2.2.1

This version corrects a bug in the file SORT.TMK.

ULTRASORT-II
Version 4.1A

This version fixes these bugs:

1-CBASIC2 numeric files would occasionally be missorted when USORTB2 or USORTM2 was used.
2-The Microsoft sort modules USI and USC.REL would not permit the construction of a parameter file if a select key was specified for an integer field. It gave an error #13.
3-When ULTRASORT was used on Microsoft files it did not sort correctly on files requiring any temporary workfiles to be created.

XASM09
Version 1.05

Version 1.04 contains a bug preventing short branch instructions from being flagged when out of range.

Version 1.05 corrects the problem. However, owners of Version 1.04 may patch a single location to correct the bug:

1-Load XASM09 under DDT.
2-Change location 1ACF from DC hex to C4 hex.
3-Exit from DDT.
4-Save patched copy with "SAVE 49 XASM09.COM"

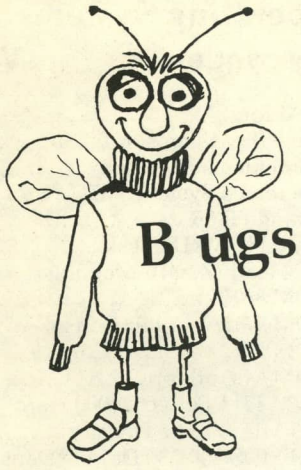
Z80 Development Package
Version 3.35

This version fixes some minor bugs in the editor, and cures a major assembler problem which caused the disk directory to be corrupted when an output file was written to a full disk.

Tips Contest

We're eager for more entries in the Tip Contest, so stop and write down some of those clever ideas you have. And the better the documentation you send with your entry, the better its chance of being published and winning you $50. Don't leave anything to our imagination. Explain your tip, its applications, its limitations.

Don't forget that in February 1982 we'll be offering a Grand Prize for the Tip of the Year. We guarantee you'll wish you entered this contest when you find out what that big prize will be.



PL/I-80
Version 1.3

Concatenation does not work properly when applied to subscripted character variables. This problem can be avoided in some circumstances by assigning temporary variables.

Pickles and Trout CP/M for TRS-80 Model II With Wordstar Version 2.26

Wordstar does not work properly if installed to access the screen memory directly. Install WordStar to work with the normal console driver supplied with Pickles and Trout CP/M.

PIP
Version 2.2

PIPing to the PRN device causes output to be routed to the LST device with default tab expansion and pagination. A side-effect of using the PRN device is that the I/O byte will be set to 0 by PIP. This will cause problems in any system which uses the I/O byte to re-direct I/O.

ZSID

ZSID will not properly assemble certain 16-bit loads if the operand is in the top page of memory. This seems to be a holdover from earlier versions of DDT. DDT version 2 and SID have fixed this bug.

New Products

The products mentioned in this column are available from their authors, computer stores, software distributors, and software publishers.

PRISM/LMS
by Micro Applications Group

This list manager is designed to store various types of information, as defined by the individual user. A predefined file format is not part of the package. The user may also define multiple keys.

A forms generator allows the creation of mailing labels, envelopes, preprinted forms, Rolodex cards, personalized form letters and other forms.

Reports can be produced to show listings of selected fields and data records. Column totals may also be printed for numeric fields.

Series 8000 Medical and Dental Management Systems
by Univair

These two systems are designed for single physicians/dentists or for small clinical practices; they handle accounts receivable and billing. They utilize a balance forward accounting method on charges and payments, with monthly aging. Patients are referenced by account number and last name. Error checking signals duplicate account numbers, validity of dates, etc.

Patient master files comprise account numbers, names, responsible parties, full addresses, phone numbers, insurance company numbers, policy numbers, year-to-date personal payments, billing status, aged balances, and payments, as well as date of last visit, and date of last payment (automatically updated). Separate files handle charges and payments; these are cross-indexed by patient, doctor, insurance code, and procedure code. In addition, doctor files, insurance files, and procedure code files with standard charges are provided. About 3000 patient and charge records can be stored on a 500K disk. A patient scheduling register is included in the packages.

Reports are as follows: clinic data, detail and summary doctor/dentist reports, insurance company report, billing messages, procedure code descriptions and charges, detail and summary patient master, patient numeric and alphabetical account number listings, patient sorting routines, detailed ticket and payment registers, aged trial balances, and production analysis reports by doctor/dentist or procedure codes.

At the end of the month these packages generate billing statements, produce mailing labels, fill out portions of standard forms.

These products require a Z80, 8080, or 8085 processor; 48K RAM, CBASIC2 (version 6 or 7), and a minimum of two disks (241K capacity each) are also needed.

Pascal MT+ SpeedProgramming Package
by MT Microsystems

Not actually an independent product, this package is comprised of development tools for the Pascal MT+ programmer, concentrating on improving productivity and efficiency.

SPP contains a supervisory program in source code, a screen-oriented text editor, an interactive syntax scanner (on demand by operator command). A source text reformat utility and variable spelling check utility are included, also only activated upon user command. Source code modifications are logged on disk, and a hard disk backup utility is part of the package. The package

can be adapted to any terminal and users can add their own tools.

A syntax scanner and screen editor work together, signalling errors by cursor placement and a screen message. The package is intended to thus detect "dumb" errors which may be tedious and frustrating to search for. A misspelled variable check renders a frequency count of the identifiers in the user's text, helping to spot those identifiers used only once, which may be misspellings.

As mentioned before, an on-line log of source code modifications is maintained. In addition, version number comments can be automatically updated at the beginning of source program files. A reformatting utility can readjust margins and indentations for easier reading.

SpellStar
by Micropro

This spelling package is designed to accompany Micropro's WordStar and works interactively with it, permitting a user to correct errors as soon as SpellStar has completed flagging them in a document. SpellStar comes with a 20,000-word dictionary and permits additions, depending on a user's disk space.

After checking a document it reports the number of misspelled words, of different words, of words currently in the main and supplementary dictionaries. Errors are flagged in the document and must be corrected by the user. They are high-lighted and tagged with a flashing character. Words can be added to the main dictionary, added to a supplementary dictionary, corrected, or ignored.

SpellStar sells for $250. Lifelines will have more details on system requirements in next month's issue.

New Publications

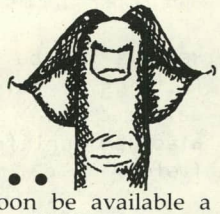
6809 Microcomputer Programming and Interfacing, with Experiments
by Andrew C. Staugard, Jr.

This book is intended to provide instructions on programming and interfacing the 6809 8-bit microprocessor. It explains addressing modes, registers, data movement instructions, arithmetic, logic, branching, I/O signals, applications, and test instructions for the 6809. The chapters are structured with objectives at the beginning of each, and review questions and answers at the end.

The Logic Design of Computers--An Introduction
by M. Paul Chinitz

This is an instruction book, not requiring knowledge of electronics or higher mathematics. Computer organization, number representation, and machine-language programming are introduced first, followed by a basic set of logic elements used to construct the common logic modules. The modules are assembled to create a simple computer, which is exhaustively analyzed.

Rumor Has It...



...that there will soon be available a hardware addition for all PET microcomputers, called The SoftBox™, which will provide PET users with the ability to run CP/M software. It is suspected that The SoftBox will be the functional equivalent of the Z80 SoftCard™ for the Apple®, except that the unit will simply daisychain down the GPIB cable. It is rumored that it will be compatible with all CBM units in the field from 2000 through 8032, even booting off cassette if necessary, and with all revs of PET disk from 2040 through 8050. No price or availability information yet, but we guess $1,000 with operating systems, for shipment in 45 days.

...that a new software product called CP/Emulator™, which allows PDP-11 and VAX users to run CP/M and many applications programs, is soon to appear. Versions for all standard DEC™ operating systems (RT-11™, RSX-11™, RSTS/E™ and VMS™) should be ready in the near future. Price wil be around $500 for RT-11 and $1500 for the multi-user systems. A $75 demo without disk writing capability and without DEC to CP/M file exchange will also be offered. No figures yet on speed, but expect 5 times slower or worse on the small 11's.

Operating Systems

Description	Version
CP/M for:	
Apple II w/Microsoft BASIC	2.0
Cromemco System 3 8"	1.4
Cromemco System 3 8"	2.2
Durango F-85	2.23
Heath H8 w/H17 Disk	1.43
H89 Heath/Zenith-Magnolia	2.2
iCOM 3812	1.41
iCOM 3712 w/Altair Console	1.41
iCOM 3712 w/IMSAI Console	1.41
iCOM Microfloppy (2411)	1.41
iCOM 4511/Pertec D3000 Hard	2.22
Intel MDS Single Density	2.2
Intel MDS 800/230 Double Density	2.2
MITS Altair 3202 Disk	1.41
Micropolis Mod I-All Consoles	1.4x
Micropolis Mod II-All Consoles	1.4x
Micropolis Mod I	2.20A
Micropolis Mod II	2.20A
Compal Micropolis Mod II	1.4
Exidy Sorcerer Micropolis Mod I	1.4x
Exidy Sorcerer Micropolis Mod II	1.4x
Vector MZ Micropolis Mod II	1.4x
Versatile 3B Micropolis Mod I	1.4x
Versatile 4 Micropolis Mod II	1.4x
North Star SD	1.41
Mostek MDX STD Bus	2.2
Sol North Star SD	**1.41**
North Star SD IMSAI SIO Console	**1.41**
North Star SD MITS SIO Console	**1.41**
North Star DD	**1.45**
North Star SD	2.22
North Star DD/QC w/Corvus	2.22
North Star DD/QD	2.22
Ohio Scientific	**2.24**
Ohio Scientific C-3B	**2.24**
Ohio Scientific C-3C' (Prime)	**2.24**
Processor Technology Helios II	1.41
by Lifeboat TRS-80 5¼" (Mod I)	1.41
by Lifeboat TRS-80 Mod II	2.24A
by Cybernetics TRS-80 Mod II	2.25

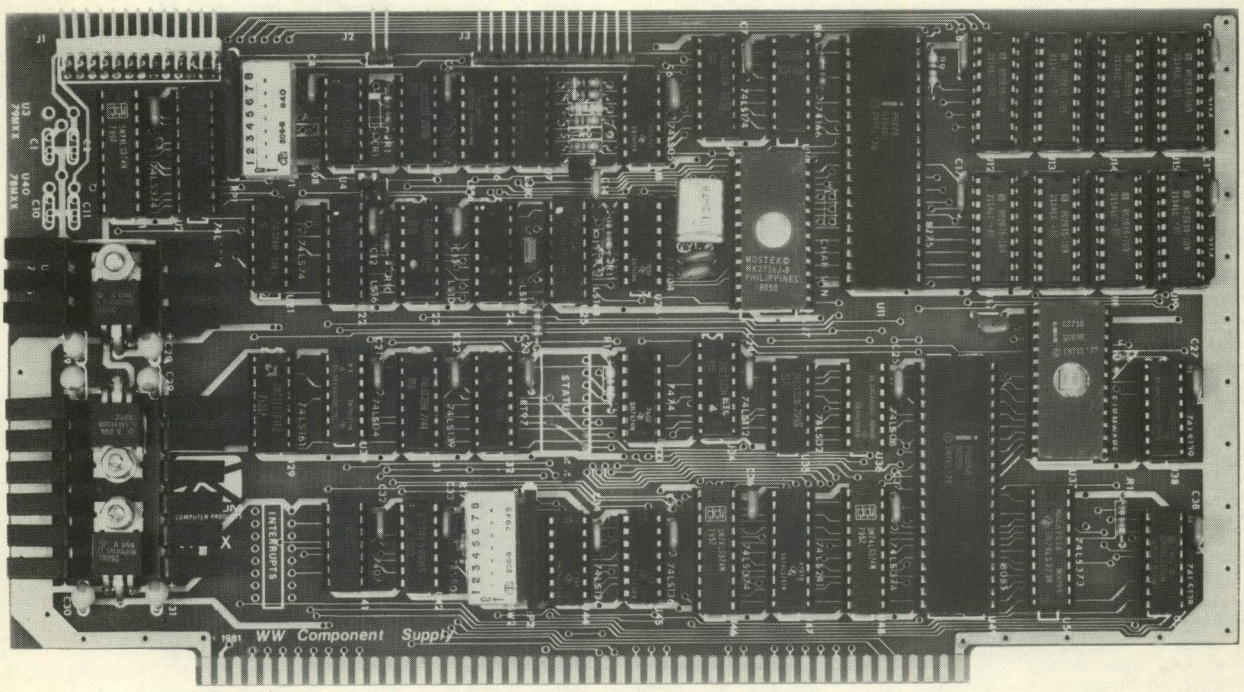
OASIS for Altos, Bell Controls, Billing, California Computer Systems, Compucorp, Cromemco, Delta, Digital Microsystems, Dynabyte, Godbout, GRI, Index Computer Systems, IBC, Intertechnique, Kontron, Media Systems Corporation, Micromation Doubler, Morrow Thinker Toys, NNC Electronics, Onyx, Quay, S.D. Systems, Teletek, TRS-80 Model II, Vector Graphics, Vorimex, Zilog.

Hard Disk Modules

Description	Version
Corvus Module	**2.0**
Apple-Corvus Module	**2.0**
KONAN Phoenix Drive	1.8
Micropolis Microdisk	**1.92**
Pertec D3000/iCOM 4511	1.6
Tarbell	1.5

New products and **new versions** appear in boldface.

INTELLIGENT VIDEO I/O FOR S-100 BUS



VIO-X

The VIO-X Video I/O Interface for the S-100 bus provides features equal to most intelligent terminals both efficiently and economically. It allows the use of standard keyboards and CRT monitors in conjunction with existing hardware and software. It will operate with no additional overhead in S-100 systems regardless of processor or system speed.

Through the use of the Intel 8275 CRT controller with an onboard 8085 processor and 4k memory, the VIO-X interface operates independently of the host system and communicates via two ports, thus eliminating the need for host memory space. The screen display rate is effectively 80,000 baud.

The VIO-X1 provides an 80 character by 25 line format (24 lines plus status line) using a 5 × 7 character set in a 7 × 10 dot matrix to display the full upper and lower case ASCII alphanumeric 96 printable character set (including true descenders) with 32 special characters for escape and control characters. An optional 2732 character generator is available which allows an alternate 7 × 10 contiguous graphics character set.

The VIO-X2 also offers an 80 character by 25 line format but uses a 7 × 7 character set in a 9 × 10 dot matrix allowing high-resolution characters to be used. This model also includes expanded firmware for block mode editing and light pen location. Contiguous graphics characters are not supported.

Both models support a full set of control characters and escape sequences, including controls for video attributes, cursor location and positioning, cursor toggle, and scroll speed. An onboard Real Time Clock (RTC) is displayed in the status line and may be read or set from the host system. A checksum test is performed on power-up on the firmware EPROM.

Video attributes provided by the 8275 in the VIO-X include:

- FLASH CHARACTER
- INVERSE CHARACTER
- UNDERLINE CHARACTER or
- ALT. CHARACTER SET
- DIM CHARACTER

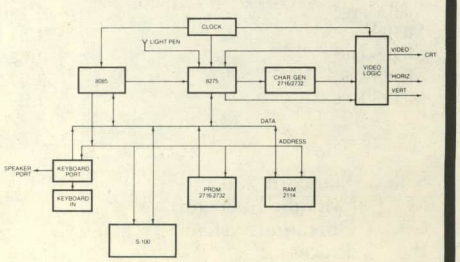
The above functions may be toggled together or separately.

The board may be addressed at any port pair in the IEEE 696 (S-100) host system. Status and data ports may be swapped if necessary. Inputs are provided for parallel keyboard and for light pen as well as an output for audio signalling. The interrupt structure is completely compatible with Digital Research's MP/M ®.

Additional features include:

- HIGH SPEED OPERATION
- PORT MAPPED IEEE S-100 INTERFACE
- FORWARD/REVERSE SCROLL or
- PROTECTED SCREEN FIELDS
- CONVERSATIONAL or BLOCK MODE (opt)
- INTERRUPT OPERATION
- CUSTOM CHARACTER SET
- CONTROL CHARACTERS
- ESCAPE CHARACTER COMMANDS
- INTELLIGENT TERMINAL EMULATION
- TWO PAGE SCREEN MEMORY

VIO-X1 - 80 × 25 5 ×7 A & T **$295.00**
Conversational Mode
VIO-X2 - 80 × 25 7 × 7 A & T **$345.00**
Conversational & Block Modes



VIO-X S-100 I/O INTERFACE

FULCRUM™ COMPUTER PRODUCTS

Distributed by:
WW COMPONENT SUPPLY INC. 1771 JUNCTION AVENUE • SAN JOSE, CA 95112 • (408) 295-7171

June 12, 1981

VERSION LIST

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

Product	S	M	OS	P	MR	$	
ACCESS-80	**1.0**		**CP/M**	**8080/Z80**	**54K**	**792**	
Accounts Payable/Cybernetics	3.1		CP/M	Z80	64K	500	Needs RM/COBOL
Accounts Payable/Structured Sys	1.3B		CP/M	8080	52K	840/40	
Accounts Payable/Peachtree	10-10-80		CP/M	8080	48K	530/60	Needs OBASIC
Accounts Receivable/Cybernetics	3.1		CP/M	Z80	64K	500	Needs RM/COBOL
Accounts Receivable/Peachtree	10-10-80		CP/M	8080	48K	530/60	Needs OBASIC
Accounts Receivable/Structured Sys	1.4C		CP/M	8080	56K	840/40	
Address Mngemt.Sys	1.0		CP/M	8080		150	Requires 2 drives
ALDS TRSDOS	3.38		TRSDOS		32K	80/35	
ALGOL/60	4.8C		CP/M	8080	24K	250	
ANALYST	2.0		CP/M	8080	52K	250/20	Needs OBASIC, QSORT or VSORT
APL/V80 Compiler	3.2		CP/M	Z80	48K	500	Needs APL terminal
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 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	16K	200	
BSTMS	1.2	1.2	CP/M	8080	24K	200	
BUG/uBUG Debuggers	2.03		CP/M	Z80		129/25	
BDS C Compiler	1.44	1.44T	CP/M	8080	32K	150/30	
Boss Finc'l Acctg Sys			**CP/M**	**8080**	**54K**	**2495**	
Whitesmith's C Compiler	2.0		CP/M	8080	60K	630/30	
CBASIC2	2.07P	2.17P	CP/M	8080	32K	125/20	Needs 2 drives w/min. 240K ea.
CBS Applications Builder	1.22		CP/M	8080	48K	395/40	CDOS version too
CIS COBOL Standard	4.3,1		CP/M	8080	48K	850/50	
CIS COBOL Compact	3.46	3.46	CP/M	8080	32K	650/50	
FORMS 1 CIS COBOL Form Generator	1.06	1.06	CP/M	8080		150/20	
FORMS 2 CIS COBOL Form Generator	1.1,6	1.16	CP/M	8080		200/20	
COBOL-80 Compiler	4.01A	4.01A	CP/M	8080	48K	710/35	
COBOL-80 PLUS M/SORT	4.01		CP/M	8080	48K	845/65	
CONDOR	1.10		CP/M	Z80	48K	695/35	
CREAM (Real Estate Acct'ng)	2.3		CP/M	8080	64K	250	Needs CBASIC
DATASTAR Information Manager	1.101		CP/M	8080	48K	350/60	
Datebook	2.03		CP/M	8080	48K	295/30	
DESPOOL Print Spooler	2.0		CP/M	8080	19K	80	
DISILOG Z80 Disassembler	4.0	4.0	CP/M	Z80	24K	110	Zilog mnemonics
DISTEL Z80/8080 Disassembler	4.0		CP/M		24K	110	
EDIT Text Editor	2.06		CP/M	Z80		129/25	
EDIT-80 Text Editor	2.0		CP/M	8080		99/25	
ESQ-1	2.1A		CP/M	8080	48K	1495/50	Needs CBASIC2
FABS	2.4A		CP/M	8080	32K	195/25	
FILETRAN		1.2	TRSDOS		32K	99/20	1-way TRS-80 Mod I, TRSDOS-CP/M
FILETRAN		1.4	CP/M		32K	149/20	2-way TRS-80 Mod 1, TRSDOS & CP/M
FILETRAN	1.5		CP/M		32K	99/20	1-way TRS-80 Mod II, TRSDOS -CP/M
Financial Modeling System	2.0		CP/M		48K	300	
FORTRAN-80 Compiler	3.42	3.42	CP/M	8080	36K	435/35	
FORTRAN TRS	3.38		TRSDOS			80/35	
FORTRAN Package	3.38		TRSDOS			150/35	
FPL	2.0	2.0	CP/M	8080	48K	695/30	
General Ledger/Cybernetics	1.3C		CP/M	Z80	48K	500	Needs RM/COBOL
General Ledger/Peachtree	10-10-80		CP/M	8080	48K	530/60	Needs OBASIC
General Ledger/Structured Sys	1.4C		CP/M	8080	52K	840/40	
GLECTOR Accounting System	2.0		CP/M	8080	56K	350/25	Use w/CBASIC2, Selector 111 C-2
HDBS	1.04		CP/M	+	52K	300	
Lifeboat IBM/CPM	1.1		CP/M	8080		175	
Inventory/Peachtree	10-10-80		CP/M	8080	48K	530/60	Needs OBASIC
Inventory/Structured Sys	1.0C		CP/M	8080	52K	640/40	
JRT Pascal			**CP/M**	**8080**	**56K**	**225/25**	
LETTERIGHT Text Editor	1.1B		CP/M	8080	52K	200/25	
MAC	2.0		CP/M	8080	20K	120/25	
MACRO-80 MACRO Assembler Package	3.40	3.40	CP/M	8080/Z80	159/25		
Magic Wand	1.11		CP/M	8080	32K	395/40	
MAGSAM III	4.2		CP/M	8080	32K	145/25	For CBASIC/MBASIC
MAGSAM IV	1.1		CP/M	8080	32K	295/25	Needs CBASIC
MAILING ADDRESS Mail List System	12-2-80		CP/M	8080	48K	530/60	Needs OBASIC
Mail Merge	2.26		CP/M	8080		150/25	Needs same version Wordstar
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	

KEY

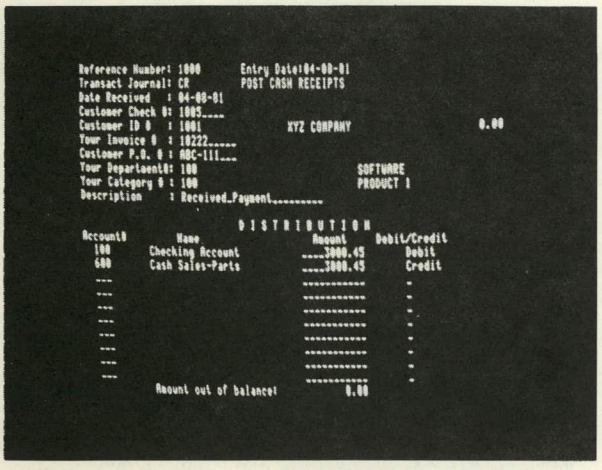
S Standard Version
M Modified Version
OS Operating System
P Processor
MR Memory Required
$ Price. These prices are given as approximations. Actual prices for these products will vary from vendor to vendor.
New products and **new versions** are listed in boldface.
+ These products are available for Z80 or 8080, in the following host languages: BASCOM, COBOL-80, FORTRAN-80, PASCAL/M, PASCAL/Z, CIS COBOL, CBASIC, PL/1, and BASIC-80 5.xx.

VERSION LIST

Product	S	M	OS	P	MR	$	Notes
Microspell	4.1		CP/M	8080	48K	249	
Mini-Wrehse Mngmt Sys	5.5		CP/M	8080	48K	650	
MP/M Operating System	1.1		MP/M	8080	32K	300/50	For Intel MDS 3000
MSORT	4.01		CP/M	8080	48K	160/20	Needs COBOL-80
Mu LISP-80 Compiler	2.03		CP/M	8080	24K	210/25	
Mu SIMP/Mu MATH Package	2.03		CP/M	8080	48K	260/30	muMATH 80
NAD Mail List System	3.0D		CP/M	8080	48K	115/25	
Nevada COBOL	1.404	1.404	CP/M	8080	32K	149/255	
Order Entry w/Inv			CP/M	Z80		500	Needs RM/COBOL
PAS-3 Medical	**1.75**		**CP/M**	**8080**	**56K**	**995/25**	**Needs CBASIC2**
PAS-3 Dental	**1.62**		**CP/M**	**8080**	**56K**	**995/25**	
ssembler	**1.02**		**CP/M**	**Z80**		**129/25**	
Pascal/M	**3.2**		**CP/M**	**8080**	**56K**	**175/25**	**Also for CDOS**
Pascal/MT Compiler	**3.2**		**CP/M**	**8080**	**32K**	**250/30**	
Pascal/MT +	5.2		CP/M	8080	52K	500/30	For Z80 too
Pascal/Z Compiler	3.3		CP/M	Z80	56K	395/25	
Payroll/Cybernetics			CP/M	Z80		500	Need RM/COBOL
Payroll/Peachtree	11-7-80		CP/M	8080	48K	530/60	Needs OBASIC
Payroll/Structured Sys	1.0E		CP/M	8080	56K	840/40	
PL/1-80	1.3		CP/M	8080	48K	500	
PLINKII	**1.08**		**CP/M**	**Z80**		**350**	
PLINK Linking Loader	3.25P		CP/M	Z80		129/25	
PMATE	2.06		CP/M	8080	32K	195	
POSTMASTER Mail List System	3.3	3.3	CP/M	8080	48K	150/20	Needs CBASIC
Professional Time Acctg	**3.11**		**CP/M**	**8080**	**48K**	**595/30**	**Needs CBASIC2**
Property Manager	10-10-80		CP/M	8080	48K	925/60	Needs OBASIC
PSORT	**1.1**		**CP/M**	**8080**		**100**	
QSORT Sort Program	2.0		CP/M	8080	48K	100	
RAID	**4.7.3A**	**4.7.3**	**CP/M**	**8080**	**28K**	**250/25**	
Real Est Aquisition Prog	2.1		CP/M	8080	56K	500	Needs CBASIC
Residential Prop Mngemt Sys	1.0		CP/M	Z80	48K	650	
Lifeboat RECLAIM Verification Prog	2.1		CP/M	8080		80	
SBASIC	**5.4**		**CP/M**	**8080**		**295/35**	
SELECTOR III-C2 Data Manager	3.24		CP/M	8080	48K	295/25	Needs CBASIC
SELECTOR IV	2.13A		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 Sys	3.0		CP/M	8080		75/25	For CP/M 1.x
STATPAK	1.2	1.2	CP/M	8080		495/30	Needs BASIC-80 4.2 or above
STRING BIT	1.02	1.02	CP/M	8080		75/25	
STRING/80 bit	1.22		CP/M	8080		95/25	
STRING/80 bit Source	1.22		CP/M	8080		295	
Supersort I Sort Package	1.5		CP/M	8080		225/40	Max.record = 4096 bytes
T/MAKER II Data Calculator	**2.2.1**		**CP/M**	**8080**	**48K**	**275/25**	
TEX Text Formatter	1.1		CP/M	8080	36K	105/50	
TEXTWRITER-III	3.6	3.6	CP/M	8080	32K	125/20	
TINY C Interpreter	800102C		CP/M	8080		105/50	
TINY C II Compiler	800201		CP/M	8080		250/50	
TRS-80 Customization Disk	1.2		CP/M	8080		75	
ULTRASORT II	**4.1A**		**CP/M**	**8080**	**48K**	**195/25**	
Lifeboat Unlock	1.3		CP/M	8080		95	Use w/BASIC-80 5.24 or above
Visicalc	1.37		Apple	8080	32K	150	
Wordindex	3.0		CP/M	8080	48K	195	Needs WordStar
WordMaster	1.07A		CP/M	8080	40K	145/40	
WordStar	2.26	2.20A	CP/M	8080	48K	445/60	
WordStar w/MailMerge	2.26		CP/M	8080	48K	575/85	
XASM-05 Cross Assembler	1.04		CP/M	8080	48K	195/25	
XASM-09 Cross Assembler	**1.05**		**CP/M**	**8080**	**48K**	**200/25**	
XASM-51 Cross Assembler	1.07		CP/M	8080	48K	200/25	
XASM-F8 Cross Assembler	1.02		CP/M	8080	48K	200/25	
XASM-400 Cross Assembler	1.02		CP/M	8080	48K	200/25	
XASM-18 Cross Assembler	1.40		CP/M	8080	48K	200/25	
XASM-48 Cross Assembler	1.60		CP/M	8080	48K	200/25	
XASM-65 Cross Assembler	1.96		CP/M	8080	48K	200/25	
XASM-68 Cross Assembler	1.99		CP/M	8080	48K	200/25	
XMACRO-86 Cross Assembler	3.40		CP/M	8080	48K	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	**3.35**		**CP/M**	**Z80**		**130**	**N/A Suprbrn,Magnolia,mod CP/M**
ZDM Debugger	1.2		CP/M	Z80		45	For Micropolis,N'Star,Apple,IBM 8"
ZDMZ Debugger	2.0		CP/M	Z80		45	See note above
ZDT Z80 Debugger	1.41	1.41	CP/M	Z80		50	N/A Superbr'n,mod CP/M
ZSID Z80 Debugger	1.4		CP/M	Z80		130	See note above.

Computerize your bookkeeping without terrifying your bookkeeper.

Introducing The Boss:™ the most advanced, yet most understandable, financial accounting system. Designed to automate your bookkeeping without confusion or mistakes.



Typical screen format - actual photograph.

If "fear of the unknown" is standing between you and computerization, you should find out about The Boss.

The Boss system is immediately comprehensible to bookkeepers because it utilizes virtually the same format they're used to.

So even someone with no previous computer experience can easily learn and operate The Boss system.

Features seldom found in packaged software.

The Boss system is fully interactive, fully departmentalized and exceptionally fast. It can generate an astounding number of complex reports at the touch of a button.

General Ledger and Accounts Receivable and Payable transactions can be entered in any order in a single program.

Up-to-the-minute financial reports can be obtained without batch processing.

The Boss system computes financial ratio analysis.

It protects data from unauthorized personnel as well as computer malfunction.

And it has the largest programming and storage capacity of any micro system.

The Boss runs on most small business computers with CP/M® or similar operating systems. Its cost is only $2,495.

Get full support from Lifeboat.

The Boss is brought to you exclusively and supported completely by Lifeboat Associates, world's largest computer software publisher. For more information about how you can profit from this extraordinary financial accounting system, send us the coupon below. Or call (212) 860-0300.

For more information on The Boss, mail coupon to Lifeboat Associates, 1651 Third Avenue, New York, NY 10028.

Name_____

Title_____

Company_____

Street_____

City_____

State_____

Zip_____

0502

Boss is a trademark of Balcones Computer Corp.
CP/M is a trademark of Digital Research, Inc.

LIFEBOAT WORLDWIDE offers you the world's largest library of software. Contact your nearest dealer or Lifeboat:

Lifeboat Associates
1651 Third Ave.
New York, N.Y. 10028
Tel: (212) 860-0300
Telex: 640693 (LBSOFT NYK)
TWX: 710-581-2524

Lifeboat Inc.
OK Bldg., 5F
1-2-8, Shiba-Daimon
Minato-ku, Tokyo, 105 Japan
Tel: 03-437-3901
Telex: 2422723 (ASRTYOJ)

Lifeboat Associates, Ltd.
PO Box 125
London WC2H 9LU, England
Tel: 01-836-9028
Telex: 893709 (LBSOFTG)

Lifeboat Associates GmbH
PO Box 168, Aegeristrasse 35
CH 6340 Baar, Switzerland
Tel: 042-31-2931
Telex: 865265 (MICO CH)

Intersoft GmbH
Schlossgartenweg 5
D-8045 Ismaning, W. Germany
Tel: 089-966-444
Telex: 5213643 (ISOFD)

Lifeboat Associates, SARL
10, Grande Rue Charles de Gaulle
92600 Asnieres, France
Tel: 1-733-08-04
Telex: 250303 (PUBLIC X PARIS)

Lifeboat Associates

Software with full support

IFELINES®
1651 Third Avenue / New York, N.Y. 10028

NEW YORK N. Y.

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

FIRST CLASS MAIL