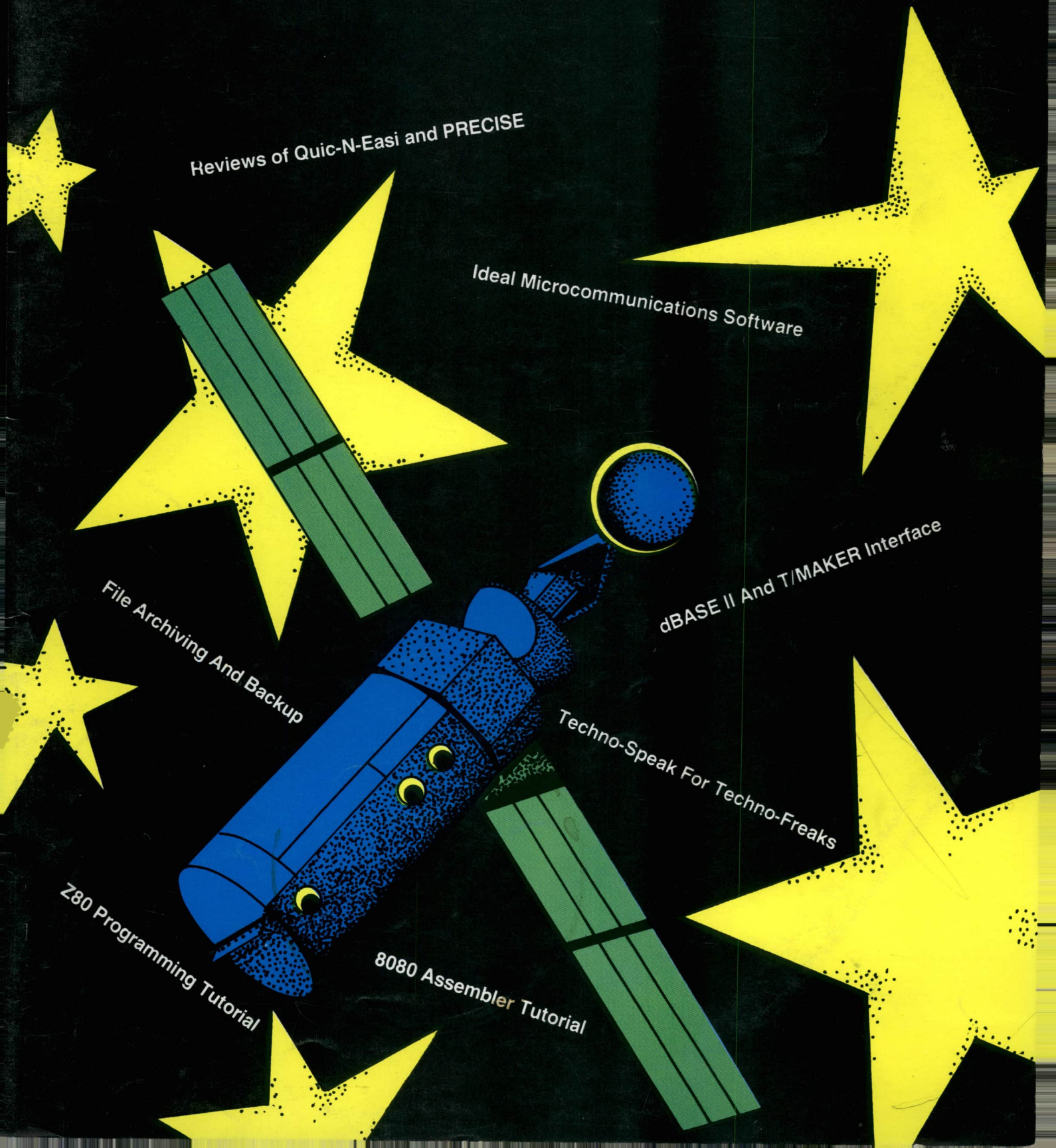
LIFELINES

The Software Magazine™

$3.00 August 1982 Volume III, No. 3 (ISSN 0279-2575, USPS 597-830)



TIM™ III

The Non-Programming Approach to Data Base Management

Data Base Management

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

What is T.I.M.?

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

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

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

The Manual

Many people believe that the manual is just as important as the software itself, a view that we at Innovative Software, Inc. tend to share. The manual for T.I.M. is divided into two sections, the Reference section and the Primer. The Reference section describes all of T.I.M.'s commands and subcommands. This is done in English, not in technical terms or in our own language. Even if you have never seen a computer before in your life, you'll be able to read and understand our manual immediately. The second section is a primer which goes through several examples for you, again in plain English. These true-to-life examples take the beginner by the hand, and instructs him what to do and when. You will be able to see for yourself that T.I.M.'s only limitation is the imagination of the user.

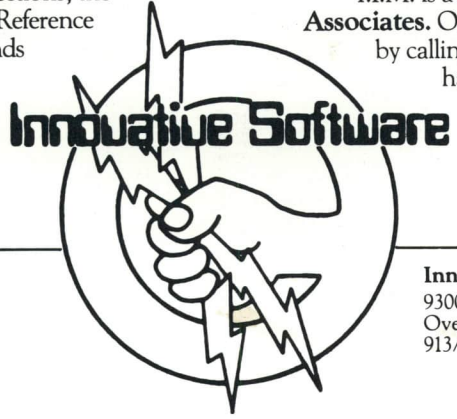
Features of T.I.M.

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

How powerful is T.I.M.? With a maximum record size of 2400 characters and the ability to keep up to forty fields sorted properly at all times, T.I.M. is powerful enough to handle just about any application. T.I.M. can handle over 32,000 records per file, and two files can be linked together for reports if your application requires a many-to-one relationship. T.I.M. also includes all of the same editing commands as your word processor, thus making data entry and editing a snap. You can also pull selected records from one file to place them into another. Files may be restructured to add or subtract fields and/or change field lengths or types. T.I.M. even has it's own utility for backing up hard disks onto floppies.

Where to Find T.I.M.

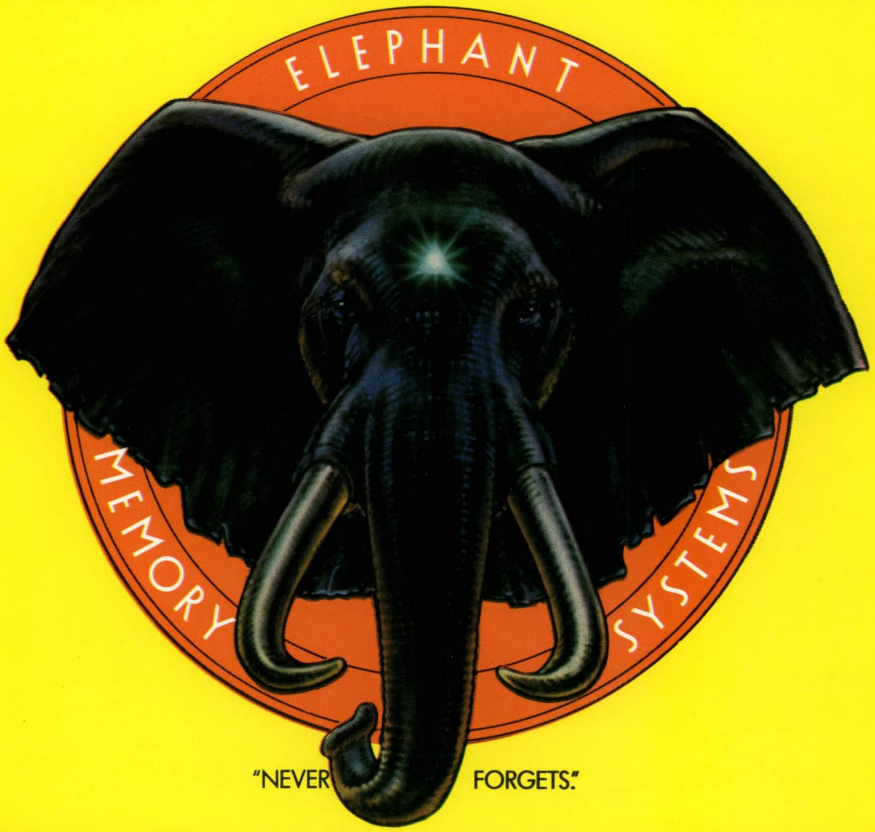
T.I.M. is available from **Lifeboat Associates.** Or you may purchase from us direct by calling 913/383-1089. Either way you will have the finest data management program available.



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

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

REMEMBER:



MORE THAN JUST ANOTHER PRETTY FACE.

Says who? Says ANSI.

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

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

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

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

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

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

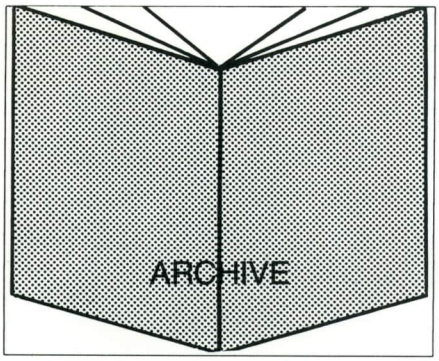
ELEPHANT.™ HEAVY DUTY DISKS.

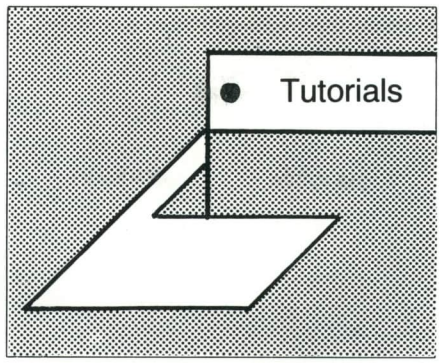
For a free poster-size portrait of our powerful pachyderm, please write us.
Distributed Exclusively by Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021
Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.

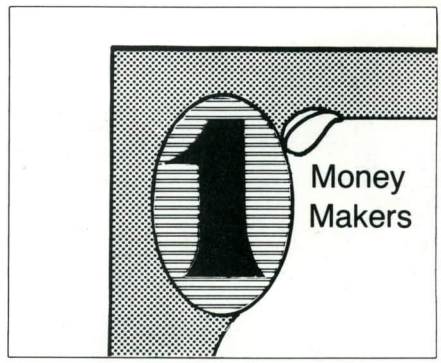
LIFELINES
The Software Magazine

August 1982 Volume III, No. 3

DEPARTMENTS







FEATURES

Display Manager™

The competitive edge in applications development.

You can significantly reduce development time and provide better application programs with Display Manager from Digital Research. Display Manager lets you interactively design displays faster than ever before, and ensures that whatever features your CRT supports can be used by the program, automatically. Since Display Manager supports most CRT attributes, including flashing, reverse video, underlining, and highlighting, your program is more dramatic and easier to use, no matter what CRT you use. You can even test a prototype application without a lot of costly and laborious coding. In short, Display Manager saves time, provides CRT independence and saves memory.

Display Manager works with Digital Research's commercial programming languages, Pascal/MT+,™ PL/I-80,™ and CB-80,™ the CBASIC® Compiler. Combined with Display Manager, they add up to the most powerful programming packages you can buy. So try Display Manager, the advanced productivity tool that makes your CP/M compatible programs better than the competition's. For more information, call Digital Research, (408) 649-5500 or (408) 649-3896, or write to 160 Central Avenue, Pacific Grove, CA 93950.

DIGITAL RESEARCH™
The creators of CP/M™

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

Far East
Microsoftware Associates
6 Floor A.Y. Building, 3-2-2 Kitaaoyama
Minato-ku, Tokyo 107, Japan, 03-403-2120
Telex: 2426875

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

Opinion
Editorial Comments

Edward H. Currie

Sometimes the Most Obvious is the Most Obscure

In previous editorials we discussed the hardware of the future and even suggested some forms it might take. The response has been gratifying; you seem to share our interest in this fascinating topic. For those who have been following this column, NCC in Houston held some interesting surprises, one of which was a small 8088 system. As we had predicted, this system uses bubble memory as a virtual floppy and features one of the finest flat screen displays we have seen. Undoubtedly this is the first of many such systems.

8088/8086 hardware is now everywhere to be seen, and increasing numbers of applications are appearing. These programs are, often as not, new software written expressly for the sixteen bit environment.

Your response to our book reviews has been excellent and we will be devoting a monthly column to discussion of new books and periodicals. We welcome any suggestions you may have for this column.

DEC has announced an interesting line of new microcomputer products, including, among other items, dual processors with both CP/M-80 and CP/M-86 co-resident in memory at the same time. The system first examines the directory entry for a .COM file and then decides whether to enter CP/M-80 or CP/M-86 for its execution.

Those of you who are aspiring authors for microcomputer applications packages and/or documentation/books should take a hard look at the IBMPC marketplace. IBM has set some high standards for offerings in this marketplace, and the demand will be substantial for years to come.

A reader pointed out that, while TIMEX will soon be providing the Sinclair Z80 system, there is little software for this machine. The entrepreneurs among you might take this hint.

One often-overlooked area for new software development is in relatively simple applications programs of limited scope, which some of us might find uninspiring, intellectually unchallenging and perhaps downright frustrating. Interest in such programs stems from the realization that first time microcomputer users represent an infinite sink for programs which, while rudimentary, are trivial to learn and use.

A considerable number of 68000 machines are now appearing, but the software available is rather uninspiring. Virtually all of these systems are available with UNIX. Perhaps this is yet another area for authors to pursue.

Good documentation is lacking for almost all programs and supplementary texts find a significant customer base. After all, there is much to be said for the kind of good documentation which renders relatively difficult application packages easier to use.

Don't fall into the obvious trap of believing that the only new products of interest are those which push forward the frontiers of the technology.

And from your point of view, what could be better than to have a machine cranking out royalty dollars while you sleep or fish or whatever ...

Graphics software is increasing in popularity and is, in most cases, available for a variety of printers; though screen graphics is relatively primitive – unless hardware graphics add-ons or true graphic terminals are employed. Currently there doesn't seem to be any comprehensive graphics software for micros operating in the scientific environment. Those of you conversant in FORTH might consider producing graphics packages for engineering and scientific applications, together with boards (such as the Microangelo hardware, etc.).

There is now an RPG package for microcomputers, and there are several vendors for ADA subsets. If you are planning to use ADA in serious applications, other readers would be most interested in articles describing your observations.

The iRMX-86 users group (iRUG) is now formally chartered and has been met with great enthusiasm. As you may know, Lifeboat Associates is cosponsoring this group together with Intel. If you have applications, tools, utilities, etc. for this environment please give serious thought to submitting them to iRUG. This will do much to stimulate the worldwide interest in this group. A newsletter is currently in production to keep those of you that are interested advised of the group's activities and software offerings. (Send your questions or contributions to iRUG, Lifeboat Associates, 1651 Third Avenue, New York, N.Y. 10028).

There is an exciting new service planned by the post office. It seems that a number of asynchronous lines for subscribers' use will be provided, permitting ASCII text to be transmitted anywhere in the U.S., via the telephone, to a post office which will print the file in a format similar to a mailgram and deliver it the next day with the mail. We will keep you posted (no pun intended) as more details become available.

It would be wonderful if the FCC would take the hint and give the microcomputers of the world ready access to the airwaves. Perhaps they will take heart and we will be able soon to transfer files of all types around the world. Imagine being able to transfer your favorite public domain utilities to a new friend in Afghanistan.

Micros continue their all-pervasive march into every area of human endeavor and we are fortunate enough not only to observe but to participate...

George Mitchell

Dual Modes Are Standard

Printek's Model 910 also tightly couples hardware and firmware features. This $1695 printer gallops along at 170 cps, with a maximum throughput of 140 lpm and bidirectional logic-seeking capability. In addition, the printer sports a graphics capability function with a 144 × 144 dots/in.² density; it can lay down 2000 dot/sec in the graphics mode. Like similar printers, the Model 910 offers either hardware or software selectable baud rates from 300 to 9600, XON/XOFF and ETX/ACK protocols, plus parallel and serial interfaces.

Falling into the under $2000 price range, at $1595, is the Zenith Model Z-25. This printer was developed to work in concert with Zenith Data Systems Model Z-89 microcomputer and the soon-to-be-announced 8086 microprocessor-based Z-100 system. It operates at 150 cps with a maximum throughput of 300 lpm, and uses either an RS-232C or 20 mA current loop for system interface. According to Carl Goy, manager of computer products engineering, the Z-25 was designed to meet the reliability needs of the unsophisticated user, plus provide software flexibility to take advantage of existing and planned products from Zenith. Be aware, however, that this printer is also available in kit form, and it's a difficult kit to build. Numerous users have reported paper skew problems and have had to rely on Heath technical personnel to resolve the problem – something they are well-equipped to do.

To provide this flexibility, the Z-25 offers 95-character ASCII in upper/lower case, 33 block graphics characters, and software selectable pitches from 10 to 16.5 cpi. In addition the full-sized carriage can handle paper sizes up to 15-in. in width and print as many as 217 columns.

Anacom General has taken the approach of supplying high-performance printers to vertical markets, like firms dealing in shipping or printing mailing labels. The series 150/150Z printers fill this need by offering an integrated tractor drive, a head that's mounted at a five degree angle (a feature that the company says helps lower the noise level) and a 150 cps print speed. The Model 150 employs a 9 × 9 character font, while the 150Z uses a 9 × 12. In addition the $1895 Model 150Z has dot addressable graphics with a 70 × 70 dots/in.² resolution. The lower priced ($1495) Model 150 can have graphics as an option. To avoid paper waste, both are designed to skip over perforations to the first printable line, and both employ either parallel or RS-232C interfaces. The 150Z utilizes XON/XOFF handshaking protocol. A 4K buffer is standard on the higher priced mode, but you can beef up the 150's 256-character model by adding optional 1K or 4K buffers.

Interestingly, the Anacom printers are based on the Tritell design; but Anacom's vice president Richard Lombardi says the Tritell design was only a starting point, indicating that Anacom added the features it felt necessary for the market it serves.

Considered by many to be only second to Texas Instruments in reliability, is Anadex's Model 9500. This printer, priced at $1650, employs 6809 microprocessor control, has a print speed of 150/200 cps and a throughput ranging from 64 to 320 lpm, depending on the mode. Using a technique called 'grab the wire' the printer has a 72 × 60 dots/in.² graphics resolution and can provide full raster style graphic printouts. In addition, the series 9500 offers parallel, RS-232C, and 20 mA current loop interfaces as standard. Full Bidirectional printing with logic line seeking and a 600-character buffer, expandable to 2.6K-characters, is included.

Another manufacturer slanting its product line toward mid-range high-performance, high-duty cycle applications is Okidata Corp. The Model 2350, a 300 lpm serial dot-matrix printer, sports the ability to print subscripts, superscripts, underlines, handle two color printing and operate bidirectionally at speed of up to 350 cps. Reportedly, by using short line seeking logic and fast horizontal and vertical slew rates the overall throughput is expected to rival much more expensive printers, and ultimately take a very large nip out of the mid-range application market.

Although not in the under $2000 price category, carrying a price tag of over $4500, the Facit 4542 Flexhammer printer is notable for its unique print method. This high-priced unit can skim along at speed of 535 cps, handle gray scale and 4-color printing and output a graphics picture with a 72 × 140 dots/in.² resolution. The key to the Facit printer is the head design. The Flexhammer uses a stored energy technique whereby the print hammer is actually a spring, normally held back in a retracted position by a permanent magnet. When in operation, the magnetic field is interrupted, allowing the hammer to drive the wire into the ribbon. Restoring the field returns the wire to the rest position.

Although print head designs such as this are found more frequently in high-priced units like Facit's, the rapidity of technology and the need to provide high throughputs mean that similar techniques in lower cost printers are probably not far off.

Each stratum has character

Of course, not all printers are designed for the mid-range application. By far the most prevalent are those for the under $1000 market.

Typically, most of the under $1000 printers fall in the 80-column class with a few like the Epson MX-100 being in the 132-column class. Shasta General Systems (Sunnyvale, CA) vice president of operations, Phil Parise, sees this type of printer being primarily used for inexpensive, low duty cycle applications.

(continued next page)

Although most of the printers in this class are light weight in comparison to the higher-priced models, they also support such features as multiple character sets, correspondence quality printing, and graphics.

Some manufacturers, MPI for instance, are adding value by improving the paper handling system. In the case of the Model 99G, MPI has developed an integrated auto load cut sheet feeder as standard equipment. Coupled with correspondence quality printing, this feature makes the printer acceptable to low-volume applications.

Although it appears for all practical purposes that Epson owns the low-end market, others are aiming at capturing their fair share and in some cases are seeking dominance.

The one company that has received very little of the press, and has the biggest chance of cutting into Epson's market share is Axiom Corp.

This company has primarily built electron-sensitive printers used for data logging and other audit trailing applications. Some of their recent introductions include a $2450 Model 1650 Video printer that plugs directly into any video output port (computer, tv camera) and makes an absolute copy.

According to vice president Simon Harrison, Axiom is gearing up to garner part of the low end market and consequently is offering a number of interesting, if not exciting, products.

The most recent Model is the GP-100 unihammer printer priced at $389. This printer uses a spinning channeled platen and a voice coil-driven hammer; stepper motors are employed on the head shuttle. The timing is precise, to form the various characters which can be standard or downloaded from the host.

Be aware, however, that few of these printers have very much road time and it might be wise to subject them to rigorous testing before buying.

In general, with the under $1000 printers you can expect to find many of the same high-performance attributes that more expensive printers tout. The problem may be in the implementation. And currently, no real data exists on what to expect as far as long term reliability or performance. Epson is the exception, and that data is only a little over a year old.

Finally, be sure you understand what's being offered. Many of the low-cost printers employ line canceling – moves to the closest margin, rather than logic seeking, to improve throughput. The technique is workable but doesn't quite stack up to full logic seeking.

If you require a fully-formed printer, however, you might want to lay your hands on C. Itoh Electronics' latest introductions.

The Models F10-40 and 55 daisy wheel printers are designed to fit the needs of small business systems requiring letter-quality printing. Using an 8085 microprocessor to perform all the control functions, the printers require fewer parts, improving reliability of the electronics.

The F10-40 is a 40 cps model while the F10-55 clicks along at 55 cps. Both printers employ a static font impact system with a ballistic hammer arm mechanism. In addition, both employ automatic self-testing, automatic print pressure control (varied according to character shape), and can use both Diablo and Qume print wheels.

Fits most needs

Both versions of the printer are able to print 136 columns at Pica pitch or 163 for Elite fonts. Other features included: print spacing of 1/120-in., line spacing of 1/48-in., ability to print 96 character types (can use full range of print wheels) and conformance to EIA RS-232C (DTR, XON/XOFF, ETX/ACK protocol).

The F10-40 differs slightly from the F10-55. Specifically, the slower printer has a 900 msec carriage return while the high speed model returns at 500 msec, in addition, the 40 cps model handles an original plus two copies and the 55 cps unit an original plus five copies. While both use the industry standard Diablo-like ribbon cartridge, the F10-40 supports only one color wheel; the F10-55 is capable of ribbon shift and can use the black/red combination.

Easy to configure

The standard configuration for the printer is RS-232C and can be easily configured to a variety of baud rates fitting system throughput. The F10-40 can be configured from 300 to 2400 baud while the F10-55 sports baud rates from 110 to 9600 baud.

In addition to selecting baud rates, you can, via dip switches, select either an XON/XOFF or ETX/ACK protocol, a serial or line mode. The former causes data to be printed as soon as it is received from the host, while the line mode waits for a full line (136 characters) or line feed before printing.

As with other daisy wheel printers, you can also configure the parity (odd, even, none), character length (7- or 8-bits), stop bits (1 or 2), auto line feed, auto carriage space (necessary in serial mode so the carriage will move over one character width). The length of the form can be adjusted to fit the application and ranges from one to eight inches in length.

The printers can accept configuration data from the host using ANSI standard escape sequences. Typically, you can set the printer for Pica or Elite fonts, set proportional spacing on, command bold or shadow printing and automatic underscoring. Moreover, the printer can be placed in a graphics mode for tight character spacing of 1/60 inch horizontally, and 1/48 inch vertically.

Since the F10 series is pre-programmed for the Courier 10 font, you can download print wheel data so that the printer can use any currently available wheels.

Options for the printers include 8- and 12-bit parallel interfaces, a 2k buffer, a uni- or bidirectional tractor, a BDT 160 automatic sheet feeder, and alert signals/devices.

The F10 series of printers has a noise figure of 65 dbA measured at one meter from the platen. Both require 180-254, or 90-127V AC at 50/60 Hz. Available now, F10-40 price is under $1200 depending on where you buy – so shop wisely.

Lowering the noise level

Noise, especially noise generated by data processing equipment, can be a

nuisance in an office environment. Consequently, printer manufacturers are faced with meeting user-defined noise criteria.

No regulations govern printer noise levels, but many manufacturers shoot for that sometimes elusive and often asked for figure of 55 dB(A). But be aware that the noise level may have been measured in ideal conditions and can be higher in actual applications.

Bandaid treatment

Testing laboratories, such as H.L. Blachford, Inc. of Corona, CA, specialize in measuring noise and assisting designers in developing methods to dampen it. According to Neil Dunbar, General Manager of Blachford, manufacturers often have opted for bandaid noise cures rather than designing for low noise level in the first place.

Dunbar points to such cure-alls as overcases which fit over the printer and dull noise levels, as well as hastily-added noise absorbing material.

But this jury-rigged approach is apparently changing. Dunbar says that many manufacturers are now taking the time to understand what noise is and how to keep it in tow.

One of the new methods employs absorption material coupled with baffles or barriers to direct the sound to a location where it can be absorbed. And to ensure that the noise absorbing add-ins aren't a waste of time, designers are developing enclosures and paper handling methods that keep noise directed into the machine rather than letting it 'leak' into the environment.

Understand the terms

Laboratories like Blachford provide two noise measurements; the noise figure inside a noise-free room, and a figure measured in an environment similar to the one where the printer will be used. The latter measurement can be the most important and most likely will supply the real noise figure, asserts Dunbar.

No real hazard

If you've been worried that a printer banging away at 65dB(A) may be hazardous to you or your customers' health, there is no evidence pointing to such difficulties. Printer noise may be irritating, but it's more of a nuisance than a danger.

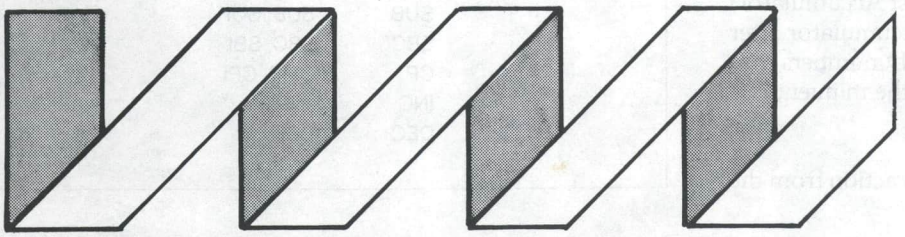
But noise specifications can be confusing. Therefore, to shed some light on the subject, Blachford's engineers offer the following definitions:

dB(A). The overall 'A'-weighted decibels or dB(A) is the scale predominately used for measuring noise. A single number, overall dB(A), is measured with a sound level meter containing a filter. The filter shapes the noise to make it roughly correspond to the way the ear hears it.

Basically, noise at low frequencies is de-emphasized relative to noise at high frequencies because the ear is less sensitive to low frequency noise.

Octave bands. For a more detailed analysis of a noise, some sound level meters have additional filters that can break up the overall noise into frequency bands called octave bands - the noise being measured in decibels. Usually 8 to 10 bands are used, and the bands are named by their center frequencies, in Hertz, or Hz (cycles per second). Thus, "parts" of the overall noise might be measured in decibels in the 31.5 Hz band, and so forth. At this point, the decibels measured in the various bands have not been adjusted to take the way the ear hears them into account.

'A'-weighted Octave Bands. The decibels read in the octave bands may be adjusted to make them correspond to the relative manner that the ear hears them, and an 'A' weighting applied. An overall 102dB(A), for example, may be considered to be the "added-up" result of the separate 'A'-weighted octave band decibels.



Feature

More Undocumented Z80 Opcodes

James R. Reinders

The Z80 chip is a powerful microprocessor indeed, but not all its power is documented. It seems that all Z80 chips will execute some instructions that very few users know about. Recently, Z80 users have found some nice additions to the documented Z80 instruction set that most Z80 chips will execute. I heard about these new instructions and tried them on my own Z80. In the course of studying these new instructions, I found that there were still more! This article will discuss all these new instructions, ponder a little on why they work, and perhaps head you on your way to your own discoveries. If you are not a Z80 user (too bad) this article should still be worth reading - maybe your favorite chip has secrets too (i.e., see Dec. '80 *Microcomputing*, "The Secret Life of the 8085").

The Z80 microprocessor by Zilog Corporation can execute all the op-codes of the 8080 microprocessor, with some minor differences in flag operations. The 8080 chip is documented to execute 244 of the 256 possible combinations of eight bits to form a byte. Zilog set about to design a microprocessor which would include the entire 8080 instruction set; then they added many new instructions. They gave the Z80 a complete second copy of the 8080 general purpose eight bit registers A, F, B, C, D, E, H and L; they named these registers A', F', B', C', D', E', H' and L'. The Z80 also got two sixteen bit registers IX and IY, a refresh register and an interrupt vector register. Contained in the Z80 instruction set are a large number of instructions not found in the 8080 set. These operations use the Z80 chip's new registers; they implement new bit manipulations, two new interrupt modes, memory move and search instructions and much more.

The Z80 does not execute the 8080A instruction set exactly as any 8080A does. As it turns out, all 8080A microprocessors do not work the same way!

NEC Microcomputers Inc. produces an 8080A chip that they consider to be an upward enhancement of the Intel 8080A. (NEC now makes an Intel 8080A equivalent chip too. The chip I call the NEC 8080A will refer to the enhanced version.) The main enhancement of the NEC 8080A is the inclusion of a SUB (Subtract) status bit, thereby allowing subtract operations on BCD (Binary Coded Decimal) numbers as you would on the Z80. On the NEC 8080A the SUB flag is bit 5 of the Program Status Word (PSW). The equivalent flag is called "N" for the Z80, and it is bit 1 of the Z80 flag register. On the Intel 8080A, the DAA (Decimal Adjust Accumulator) instruction assumes that it is adjusting the accumulator after an addition. Thus to correctly subtract BCD numbers you need to subtract from 99 hex and then add the minuend; finally a decimal adjust will give the answer.

On the NEC 8080A or the Z80, a simple subtraction from the minuend and then a decimal adjust instruction will yield the correct result.

The Carry and the Auxiliary Carry (called Half Carry for the Z80) statuses are not affected by boolean instructions in the NEC 8080A, yet the Intel 8080A and the Z80 always reset the Carry and may set or reset the Auxiliary Carry.

The Advanced Micro Devices 9080A (their version of the 8080A) clears both the Carry and Auxiliary Carry on all boolean operations, as does the Intel 8085 (Intel's upgrade for the 8080A). Yet the 8085 always sets the Auxiliary Carry for AND instructions.

A second difference between the 8080 chips and the Z80 chip deals with the 8080 parity flag. The Z80 redefines this flag (bit 2 of the flag register) as a Parity or Overflow indicator. On the Z80 this flag is used to indicate overflow with signed arithmetic; otherwise it is a parity flag as on the 8080A.

Of the differences between these chips, those affecting the

Table A

Program Status Flags.

Bit		7	6	5	4	3	2	1	0
Intel	8080A	S	Z	x	Ac	x	P	x	C
NEC	8080A	S	Z	SUB	Ac	x	P	x	C
	Z80	S	Z	x	H	x	P/V	N	C

x = indeterminate.

The NEC 8080A SUB is basically the same as the Z80 N flag. The 8080A Ac (Auxiliary Carry) is basically the same as the Z80 H (Half Carry for BCD).

Operations for which the Z80 uses bit 2 for an overflow indicator, when the 8080A would use it as a parity indicator:

Mnemonics	
Z80	8080A
ADD	ADD, ADI
ADC	ADC, ACI
SUB	SUB, SUI
SBC	SBC, SBI
CP	CMP, CPI
INC	INR
DEC	DCR

Table B

Comparison of "equivalent" instructions via their effects on the Carry and Auxiliary Carry flags, and for the Z80, the N flag and the NEC 8080A chips's SUB flag.

Mnemonic Z80	8080	Zilog Z80 C	H	N	Intel 8080A C	Ac	NEC 8080A C	Ac	SUB	AMD 9080A C	Ac	Intel 8085 C	Ac
AND	ANA, ANI	0	1	0	0	↕	*	*	*	0	0	0	1
OR	ORA,ORI	0	1	0	0	↕	*	*	*	0	0	0	0
XOR	XRA,XRI	0	1	0	0	↕	*	*	*	0	0	0	0
INC	INR	*	↕	0	*	↕	*	*	0	*	↕	*	↕
DEC	DCR	*	↕	1	*	↕	*	*	1	*	↕	*	↕
DAA	DAA	↕	↕	*	↕	↕	↕	*	*	↕	↕	↕	↕
CPL	CMA	*	1	1	*	*	*	*	*	*	*	*	*
SCF	STC	1	0	1	1	*	1	*	*	1	*	1	*
CCF	CMC	↕	?	0	↕	*	↕	*	*	↕	*	↕	*
RLCA	RLC	↕	0	0	↕	*	↕	*	*	↕	*	↕	*
RRCA	RRC	↕	0	0	↕	*	↕	*	*	↕	*	↕	*
RLA	RAL	↕	0	0	↕	*	↕	*	*	↕	*	↕	*
RRA	RAR	↕	0	0	↕	*	↕	*	*	↕	*	↕	*
ADD HL,	DAD	↕	?	0	↕	*	↕	*	0	↕	*	↕	*

Flag Notation: ↕ = flag affected according to result.
0 = flag reset (zero).
1 = flag set (one).
* = flag is not affected.
? = flag is indeterminate.

Carry flag are the most notable. Consider the problem of an 8080A program for an Intel chip with a "Jump if no carry (JNC)" instruction that won't work on a NEC chip because the NEC chip won't affect the carry on certain instructions! I routinely use XOR A and OR A instructions to clear the carry on my Z80 (such usage is very common). However, such instructions will not clear the NEC 8080A carry flag.

Tables A, B, and C show the differences between the 8080 chips and the Z80 chip. Enough information is presented to create a program that could determine which particular chip was executing it... NEC 8080A, AMD 9080A, Intel 8080A, Intel 8085 or Zilog Z80.

In the 8080, all op-codes (computer jargon for: OPeration CODES, the instructions a computer actually executes) were defined in the first byte of a machine language instruction and then followed by two, one or no additional bytes of data. If Zilog had limited the Z80 to one-byte op-codes like the 8080 had, while retaining the 8080 instruction set, they would only be able to add twelve instructions until they ran out of the 256 possible bytes.

Zilog found a way out, using what is termed a "lead-in" byte. Eight of the unused 8080 op-codes were defined as one-byte op-codes. (See Table 1.) The other four bytes must be followed immediately by a second byte to determine what the operation will be. Thus the hex bytes CB, DD, ED and FD are called lead-in bytes, effectively giving the Z80 1276 op-codes to work with: 252 without lead-ins and 1024 with lead-ins. Of course these counts disregard the fact that a lead-in can be followed by another lead-in byte with an instruction! So for CB preceded by DD or FD they work as more op-codes, giving the Z80 1786 possible op-codes. Many of these op-codes are part of the documented Z80 instruction set, but some are not.

Since all the one-byte op-codes are defined in the Z80 instruction set, to find new op-codes we must examine the operations that start with a lead-in byte. Starting with the least interesting operations, we'll look at the instructions with an ED lead-in. (When I say "ED lead-in", I am referring to the use of the hexadecimal number ED as a lead-in byte.) These are boring because there are no new op-codes yet uncovered that start with an ED lead-in. There are 56 miscellaneous defined instructions in this category, leaving another two hundred undefined. The diversity of instructions with an ED lead-in makes it next to impossible to say what the undefined op-codes might cause the Z80 to do. The *defined* op-codes include operations with I/O ports; so one can know what the undefined op-codes might result in! (See Table 2.)

The next lead-in byte is CB. Of the 256 bytes that could follow a CB, 248 are defined to do various bit manipulations, rotates and shifts. Only eight op-codes here are undocumented, yet they can now be defined. These op-codes perform operations on any register identical to SLA (Shift Left Arithmetic) instructions; however, instead of moving all bits one position to the left and filling with a zero, these new operations move all bits one position to the left and fill with a one. We will call this new instruction "SLAP", (named by the authors of my first reference) meaning Shift Left Arithmetic Plus one. (See Table 3.)

Table C

Timing differences: (i.e. who's faster ?)

Mnemonics Z80	8080	Clock periods Zilog Z80	Intel 8080A	NEC 8080A	Intel 8085
CALL cc	CALL,CC,CM, CNC,CNZ,CP, CPO,CPE,CZ	10/17	11/17	11/17	9/18
ADD HL,rp	DAD rp	11	10	11	10
INC/DEC r	INR/DCR r	4	5	5	4
INC/DEC (HL)	INR/DCR M	11	10	10	10
INC/DEC rp	INX/DCX rp	6	5	5	6
EX (SP),HL	XTHL	19	18	17	16
JP (HL)	PCHL	4	5	5	6
IN/OUT x	IN/OUT x	11	10	10	10
LD r,r'	MOV r,r	4	5	4	4
RET	RET	10	10	11	10
LD SP,HL	SPHL	6	5	4	6
OR r	ORA r	4	5	5	4

Notation:
cc = condition code: C, NC, M, P, Z, NZ, PE or PO.
r = register: A, B, C, D, E, H or L.
rp = register pair: BC, DE, HL or SP.
x = port address: 0 to FF hex.

Note: Zilog intended that their IN/OUT instructions give one more clock period to the I/O devices than the 8080 did, so in one case a longer instruction duration was desired.

(continued next page)

Table 1

The twelve bytes unused by the 8080 chip are defined for use by the Z80 chip as follows:

Hex Codes	Mnemonic
08	EX AF,AF'
10	DJNZ
18	JR
20	JR NZ
28	JR Z
30	JR NC
38	JR C
CB	Lead-in byte for bit manipulations *
D9	EXX
DD	Lead-in byte for using the IX register *
ED	Lead-in byte for miscellaneous operations *
FD	Lead-in byte for using the IY register *

* See text for description of these lead-in bytes.
Zilog Z80 uses for the unused 8080 codes.

Table 2

Hex Codes	Mnemonic	Hex Codes	Mnemonics
40	IN B,(C)	67	RRD
41	OUT (C),B	68	IN L,(C)
42	SBC HL,BC	69	OUT (C),L
43	LD (nn),BC	6A	ADC HL,HL
44	NEG	6F	RLD
45	RETN	70	IN F,(C) *
46	IM 0	72	SBC HL,SP
47	LD I,A	73	LD (nn),SP
48	IN C,(C)	78	IN A,(C)
49	OUT (C),C	7A	ADC HL,SP
4A	ADC HL,BC	7B	LD SP,(nn)
4B	LD BC,(nn)	A0	LDI
4D	RETI	A1	CPI
4F	LD R,A	A2	INI
50	IN D,(C)	A3	OUTI
51	OUT (C),D	A8	LDD
52	SBC HL,DE	A9	CPD
53	LD (nn),DE	AA	IND
56	IM 1	AB	OUTD
57	LD A,I	B0	LDIR
58	IN E,(C)	B1	CPIR
59	OUT (C),E	B2	INIR
5A	ADC HL,DE	B3	OTIR
5B	LD DE,(nn)	B8	LDDR
5E	IM 2	B9	CPDR
5F	LD A,R	BA	INDR
60	IN H, (C)	BB	OTDR
62	SBC HL,HL		

* It seems that Zilog forgot to designate a mnemonic for the 70 op-code. Zilog documented it as an input that only affects the flags. I would logically name it IN F,(C) – perhaps a slight misnomer since the flags are affected and not loaded. I have not yet seen an assembler or a disassembler that has a mnemonic for this documented op-code!

Op-codes with ED lead-in.

The remaining two lead-ins, DD and FD, reveal a lot about the inner workings of the Z80 chip. After comparing the operations of instructions when preceded by DD or FD with their operation without the lead-in, the effect of the lead-in bytes becomes apparent. (See Table 4.) The lead-in byte switches the HL register pair off and uses either IX (lead-in DD), or IY (lead-in FD), instead of the HL register pair. This means an instruction like ADD HL,BC (90 hex) becomes ADD IX,BC (DD 90 hex) or ADD IY,BC (FD 90 hex) when preceded by a lead-in. If HL was to point to operand as in INC (HL), then it becomes INC (IX+d) or INC (IY+d). In these cases, the displacement byte for the index always follows the byte after the DD or FD. This fact becomes important later on when we see instructions with two lead-in bytes.

As Table 4 shows, 40 of the 256 bytes that could follow DD or FD lead-ins are defined (counting CB as one). Since these instructions use IX or IY instead of HL, you might correctly suppose that all operations following a DD or FD lead-in and not using registers H or L at all, would operate normally. This appears to be the case and the lead-in is more or less disregarded. I have tested these instructions (such as RET, JP xxxx and LD A,D) when preceded by DD or FD. The instructions with a lead-in of ED are unaffected also, even if they use HL (i.e. ADC HL,BC). Apparently the ED masks the effect of a DD or FD lead-in. A single byte instruction, EX DE,HL, is unaffected by DD or FD lead-ins. This leads me to believe that for this instruction the Z80 does not move data, rather it switches which register the Z80 calls HL and DE, probably similar in action to the EX AF,AF' and EXX instructions. I have determined that following a DD or FD lead-in with a DD or FD will nullify the first, and only the second will take effect. Thus far we have accounted for 219 of the 256 bytes which could follow a DD or FD lead-in. The other 37 are shown in Table 5. These were undocumented, yet they follow the pattern we saw in Table 4. They do have a new twist; they use IX and IY as register pairs, thereby allowing manipulations of each half of these sixteen bit indices as eight bit registers.

I have mentioned that CB can follow DD or FD lead-ins with meaningful results. Yet there is a catch: the lead-in forces (IX+d) or (IY+d) to be operated on. We must note a rule I gave before: the displacement byte for the indices is always the third byte; thus it always follows the byte which follows the DD or FD lead-in. So when CB follows DD or FD, the next byte must be the displacement byte, then the fourth byte is the rest of the normal CB xx instruction. (See Table 6.) Note that the new SLAP (IX+d) and SLAP (IY+d) instructions logically fit this pattern.

Let's examine these four-byte instructions, and discover how they work. The first byte is DD or FD and tells the Z80 to use the IX or IY register pairs instead of the HL register pair. The second byte is CB, another lead-in byte. Now because of our displacement value position rule, in order for the documented instructions to work properly, the Z80 must decide whether the third byte is a displacement, during the sensing of the CB byte. The third byte is the displacement value; the

fourth byte tells the Z80 what the operation is. As Table 7 shows us, the first two bits determine whether the operation is in the shift/rotate group or the bit-test/set/ reset group. The next three bits designate either the type of shift/rotate to perform or the bit to operate on, depending on the first two bits. The Z80 uses the first five bits to determine what operation to perform on the operand, performs the operation, and then automatically puts the result back to (IX+d) or (IY+d). The last three bits are 110 binary in all the documented operations, indicating that the operation is on the byte pointed to by HL. Of course since we are using a DD or FD lead-in, we are operating on (IX+d) or (IY+d) instead. Since we assumed all this when the second byte came in, these bits turn out to be rather redundant. If these bits specify a register (see Table 3) instead of 110 binary, another part of the Z80 (aside from the part saving the answer in memory) puts the answer in the specified register if a memory write is done. (See Table 8.) Changing these bits on the BIT instruction seems to have no effect, evidently because no memory write is done. However, SET and RES perform memory writes, and changing the bits here as a side effect alters a register too.

Table 3

Mnemonic	Symbolic Operation	Flags C Z P/V S N H	Op-code 76 543 210
SLAP r			11 001 011 00 110 r
SLAP (HL)			11 001 011 00 110 110
SLAP (IX+d)	CY <-7<--0 <-1	↕ ↕ P ↕ 0 0	11 011 101 11 001 011 < --- d --- > 00 110 110
SLAP (IY+d)			11 111 101 11 001 011 < --- d --- > 00 110 110

Flag Notation: 0 = flag reset
↕ = flag affected according to the result of the operation.
P = flag reset for parity odd, set for even.

r	Register
000	B
001	C
010	D
011	E
100	H
101	L
111	A

Note that 110 binary is a pseudo-register code, called "M" in 8080 mnemonics, and "(HL)" in Z80 mnemonics.

The undocumented Shift Left Arithmetic Plus one – fully documented.

I wrote to Zilog to find out what they had to say about all these undocumented op-codes. My letter went in the mail in Michigan on Monday and to my delight I received a phone call from California on Thursday of the same week! I spoke with Mr. Richard McMurry, Tech-support manager for the components division of Zilog. He described the op-codes as a common phenomenon among microprocessors. Zilog did not test the undefined op-codes on their finished product, as the sheer number makes this impractical. He said that none of these undocumented op-codes were designed into the chip

Table 4

Hex Code	Mnemonics: No lead-in.	DD lead-in.	FD lead-in.
09	ADD HL,BC	ADD IX,BC	ADD IY,BC
19	ADD HL,DE	ADD IX,DE	ADD IY,DE
21	LD HL,nn	LD IX,nn	LD IY,nn
22	LD (nn),HL	LD (nn),IX	LD (nn),IY
23	INC HL	INC IX	INC IY
29	ADD HL,HL	ADD IX,IX	ADD IY,IY
2A	LD HL,(nn)	LD IX,(nn)	LD IY,(nn)
2B	DEC HL	DEC IX	DEC IY
34	INC (HL)	INC (IX+d)	INC (IY+d)
35	DEC (HL)	DEC (IX+d)	DEC (IY+d)
36	LD (HL),n	LD (IX+d),n	LD (IY+d),n
39	ADD HL,SP	ADD IX,SP	ADD IY,SP
46	LD B,(HL)	LD B,(IX+d)	LD B,(IY+d)
4E	LD C,(HL)	LD C,(IX+d)	LD C,(IY+d)
56	LD D,(HL)	LD D,(IX+d)	LD D,(IY+d)
5E	LD E,(HL)	LD E,(IX+d)	LD E,(IY+d)
66	LD H,(HL)	LD H,(IX+d)	LD H,(IY+d)
6E	LD L,(HL)	LD L,(IX+d)	LD L,(IY+d)
70	LD (HL),B	LD (IX+d),B	LD (IY+d),B
71	LD (HL),C	LD (IX+d),C	LD (IY+d),C
72	LD (HL),D	LD (IX+d),D	LD (IY+d),D
73	LD (HL),E	LD (IX+d),E	LD (IY+d),E
74	LD (HL),H	LD (IX+d),H	LD (IY+d),H
75	LD (HL),L	LD (IX+d),L	LD (IY+d),L
77	LD (HL),A	LD (IX+d),A	LD (IY+d),A
7E	LD A,(HL)	LD A,(IX+d)	LD A,(IY+d)
86	ADD A,(HL)	ADD A,(IX+d)	ADD A,(IY+d)
8E	ADC A,(HL)	ADC A,(IX+d)	ADC A,(IY+d)
96	SUB A,(HL)	SUB A,(IX+d)	SUB A,(IY+d)
9E	SBC A,(HL)	SBC A,(IX+d)	SBC A,(IY+d)
A6	AND (HL)	AND (IX+d)	AND (IY+d)
AE	XOR (HL)	XOR (IX+d)	XOR (IY+d)
B6	OR (HL)	OR (IX+d)	OR (IY+d)
BE	CP (HL)	CP (IX+d)	CP (IY+d)
CB	*** The document op-codes & SLAP follow pattern.		
E1	POP HL	POP IX	POP IY
E3	EX (SP),HL	EX (SP),IX	EX (SP),IY
E5	PUSH HL	PUSH IX	PUSH IY
E9	JP (HL)	JP (IX)	JP (IY)
F9	LD SP,HL	LD SP,IX	LD SP,IY

Effects of the lead-in bytes DD and FD.

(continued next page)

purposely and that the Z80 only executes these as a side effect of the chip's overall design. Zilog does not have knowledge of any problems in the operation of these op-codes.

(*Note*: A warning to pass along: since these are not official op-codes of the Z80 chip, new Z80 chips do not have to execute them. I have not seen nor heard of a Z80 that would not execute them as I have described. Nevertheless it would not be wise to use them in any software intended for sale or distribution... One never knows what kind of Z80 the software might end up running on, or trying to run on!)

I want to acknowledge and thank the authors whose article in the first *Lifelines* got me interested in these new op-codes. They started me digging for new op-codes, and look at what I found! I hope this article will inspire others to dig into their own processors in search of undocumented op-codes too. I would like to hear of such discoveries if and when you make them, and I am sure other readers would too. Happy Hunting!

References

1. "Undocumented Z80 Instructions", Robert H.Halsall and Hubert S. Howe Jr., *Lifelines*, June 1980, pp 10-11.

2. *The Z80 Microcomputer Handbook*, William Barden Jr., Howard W. Sam Co. Inc., Indianapolis, Indiana, 1978.

3. *OSBORNE 4 & 8-Bit Microprocessor Handbook*, Adam Osborne and Gerry Kane, OSBORNE/McGraw-Hill, Berkeley, California, 1981.

Table 5

Hex Code	No lead-in.		DD lead-in.		FD lead-in.	
24	INC	H	INC	IXH	INC	IYH
25	DEC	H	DEC	IXH	DEC	IYH
26	LD	H,n	LD	IXH,n	LD	IYH,n
2C	INC	L	INC	IXL	INC	IYL
2D	DEC	L	DEC	IXL	DEC	IYL
2E	LD	L,n	LD	IXL,n	LD	IYL,n
44	LD	B,H	LD	B,IXH	LD	B,IYH
45	LD	B,L	LD	B,IXL	LD	B,IYL
4C	LD	C,H	LD	C,IXH	LD	C,IYH
4D	LD	C,L	LD	C,IXL	LD	C,IYL
54	LD	D,H	LD	D,IXH	LD	D,IYH
55	LD	D,L	LD	D,IXL	LD	D,IYL
5C	LD	E,H	LD	E,IXH	LD	E,IYH
5D	LD	E,L	LD	E,IXL	LD	E,IYL
64	LD	H,H	LD	IXH,IXH	LD	IYH,IYH
65	LD	H,L	LD	IXH,IXL	LD	IYH,IYL
6C	LD	L,H	LD	IXL,IXH	LD	IYL,IYH
6D	LD	L,L	LD	IXL,IXL	LD	IYL,IYL
84	ADD	A,H	ADD	A,IXH	ADD	A,IYH
85	ADD	A,L	ADD	A,IXL	ADD	A,IYL
8C	ADC	A,H	ADC	A,IXH	ADC	A,IYH
8D	ADC	A,L	ADC	A,IXL	ADC	A,IYL
94	SUB	H	SUB	IXH	SUB	IYH
95	SUB	L	SUB	IXL	SUB	IYL
9C	SBC	A,H	SBC	A,IXH	SBC	A,IYH
9D	SBC	A,L	SBC	A,IXL	SBC	A,IYL
A4	AND	H	AND	IXH	AND	IYH
A5	AND	L	AND	IXL	AND	IYL
AC	XOR	H	XOR	IXH	XOR	IYH
AD	XOR	L	XOR	IXL	XOR	IYL
B4	OR	H	OR	IXH	OR	IYH
B5	OR	L	OR	IXL	OR	IYL
BC	CP	H	CP	IXH	CP	IYH
BD	CP	L	CP	IXL	CP	IYL

Note:
IXH, IXL are the high and low bytes of the IX register.
IYH, IYL are the high and low bytes of the IY register.

Undocumented effects of the lead-in bytes DD and FD.

Table 6

Mnemonic	.....Hex Codes.....		
	e = (HL)	e = (IX + dd)	e = (IY + dd)
RLC e	CB 06	DD CB dd 06	FD CB dd 06
RRC e	CB 0E	DD CB dd 0E	FD CB dd 0E
RL e	CB 16	DD CB dd 16	FD CB dd 16
RR e	CB 1E	DD CB dd 1E	FD CB dd 1E
SLA e	CB 26	DD CB dd 26	FD CB dd 26
SRA e	CB 2E	DD CB dd 2E	FD CB dd 2E
SLAP e	CB 36	DD CB dd 36	FD CB dd 36
SRL e	CB 3E	DD CB dd 3E	FD CB dd 3E
BIT 0,e	CB 46	DD CB dd 46	FD CB dd 46
BIT 1,e	CB 4E	DD CB dd 4E	FD CB dd 4E
BIT 2,e	CB 56	DD CB dd 56	FD CB dd 56
BIT 3,e	CB 5E	DD CB dd 5E	FD CB dd 5E
BIT 4,e	CB 66	DD CB dd 66	FD CB dd 66
BIT 5,e	CB 6E	DD CB dd 6E	FD CB dd 6E
BIT 6,e	CB 76	DD CB dd 76	FD CB dd 76
BIT 7,e	CB 7E	DD CB dd 7E	FD CB dd 7E
RES 0,e	CB 86	DD CB dd 86	FD CB dd 86
RES 1,e	CB 8E	DD CB dd 8E	FD CB dd 8E
RES 2,e	CB 96	DD CB dd 96	FD CB dd 96
RES 3,e	CB 9E	DD CB dd 9E	FD CB dd 9E
RES 4,e	CB A6	DD CB dd A6	FD CB dd A6
RES 5,e	CB AE	DD CB dd AE	FD CB dd AE
RES 6,e	CB B6	DD CB dd B6	FD CB dd B6
RES 7,e	CB BE	DD CB dd BE	FD CB dd BE
SET 0,e	CB C6	DD CB dd C6	FD CB dd C6
SET 1,e	CB CE	DD CB dd CE	FD CB dd CE
SET 2,e	CB D6	DD CB dd D6	FD CB dd D6
SET 3,e	CB DE	DD CB dd DE	FD CB dd DE
SET 4,e	CB E6	DD CB dd E6	FD CB dd E6
SET 5,e	CB EE	DD CB dd EE	FD CB dd EE
SET 6,e	CB F6	DD CB dd F6	FD CB dd F6
SET 7,e	CB FE	DD CB dd FE	FD CB dd FE

Note:
dd is the signed displacement byte for (IX + dd) or (IY + dd)

Comparison of the shift/rotate and bit manipulation operation codes.

Table 7

The fourth byte of all instructions in Table 6 for e = (IX + dd) or e = (IY + dd) are of the form: aabbb110 binary.

aa = 00 for the shift/rotate group (RLC etc.)
= 01 for bit testing (BIT).
= 10 for bit setting (SET).
= 11 for bit resetting (RES).

If aa = 00 then bbb defines which shift or rotate to perform.

If aa = 01, 10 or 11 then bbb defines which bit to operate on.

bbb	operation	bbb	bit
000	RLC	000	0
001	RRC	001	1
010	RL	010	2
011	RR	011	3
100	SLA	100	4
101	SRA	101	5
110	SLAP	110	6
111	SRL	111	7

Fourth byte of Table 6 instructions.

Table 8

We will use RES 4,(IY + dd) as an example. Normally this op-code is FD CB dd A6. It will only affect bit 4 of the byte pointed to by IY + dd, and it will force bit four to a zero (reset).

If we alter the fourth byte (A6 = 10100110 binary) by changing the last three bits from 110 to 010 (code for register D – see Table 2) we get A2 (= 10100010 binary). Thus the whole operation becomes FD CB dd A2. This op-code will operate identically to the RES 4,(IY + dd) instruction except that upon completion, the register specified (register D in this case) will contain the same number as the location pointed to by IY + dd.

So the old code:	FD CB dd A6	RES 4,(IY + dd)
	FD 56 dd	LD D,(IY + dd)
Becomes:	FD CB dd A2	RES D,4,(IY + dd)

Similar changing of such op-codes gives us:

RLC r,(ii + dd)	RRC r,(ii + dd)	RL r,(ii + dd)
RR r,(ii + dd)	SLA r,(ii + dd)	SRA r,(ii + dd)
SLAP r,(ii + dd)	SRL r,(ii + dd)	SET r,b,(ii + dd)
	RES r,b,(ii + dd)	

ii = IX or IY.
r = any register: B, C, D, E, H, L, or A as in Table 3.
b = any bit: 0, 1, 2, 3, 4, 5, 6 or 7 as in Table 7.

New shift/rotate and bit manipulation instructions with modification to do a register load too.

Software Notes
Tips & Techniques
Patching PIP For Multiple File Transfers

Kelly Smith

"We shall not cease from exploration, and the end of all our exploring will be to arrive where we started and know the place for the first time." – T.S. Eliot, Little Gidding

Digital Research's PIP (Peripheral Interchange Program) is the CP/M-80 transient command used to move a single file or multiple files from one disk to another, or to other peripheral devices. Unfortunately PIP will not allow the transfer of files to multiple disks – you must always exit PIP and re-log any new disks or warm boot (by entering your keyboard Control-C) the CP/M-80 system, then invoke PIP again to proceed, an agonizing process if you want to do multiple disk copies of specific files.

How would you like to be able to transfer files to multiple disks without exiting PIP, as well as have it repeat the last operation that you entered at PIP's command level with ONE keystroke? It's *easy*! Edit and assemble the following 'patch', then merge it into your existing CP/M-80 (Version 2.2) 'PIP.COM' utility as follows:

```
A>ddt pip.com<cr> <-- invoke DDT to load memory with 'PIP.COM'
DDT VERS 2.2      <-- DDT telling us that...
NEXT  PC
1E00 0100         <-- 'PIP.COM' uses 29 pages of memory
-ipatch.hex<cr>   <-- Input the 'patch' file
-r<cr>        <-- Read it in to memory (overlaying parts of PIP)
NEXT  PC
1E00 0000
-^C               <-- Control-C to exit DDT

A>save 29 rpip.com<cr> <-- save 29 pages of memory as 'RPIP.COM'
```

So, for a little "demo" of what you can do and expect from adding this 'patch', let's give it a whirl:

```
A>rpip<cr>        <-- invoke our new 'Repeating PIP'...

PIP 1.5 with (R)eset Disks and (Q)uick Repeat
*r<cr> <-- lets try 'R' for R)eset of the disk system
Resetting all disks to R/W <-- keeping user informed...
*b:=a:*.lib<cr> <-- change any (or all) disks here if you like,
                    and start PIP'n to your hearts content...
COPYING -
MACRO.LIB
COPYRGHT.LIB
*q<cr>          <-- change disks, and then a little "quickie"
                    with Q)uick Repeat...

Repeating:  b:=a:*.lib <-- PIP remembers us...amazing!

COPYING -
MACRO.LIB
COPYRGHT.LIB
*q<cr>          <-- change disks again for two "quickies"
                    in a row (and no Vitamin-E required!)
Repeating:  b:=a:*.lib

COPYING -
MACRO.LIB
COPYRGHT.LIB
*         <-- PIP waiting patiently for something else to do...
```

Alright, now that I have whetted your appetite to incorporate this into your PIP. . . here's the code to do it!

(continued on page 19)

Feature

8080 Assembler Programming Tutorial: Subroutines, Part 3

Ward Christensen

New Terms

The first several installments of this tutorial presented terms related primarily to 8080 programming. Since the tutorials are getting into CP/M-80 related I/O, it's time to present a few more terms which might be new to you. Skip to "CONSOLE I/O" if you know these terms.

BIOS

BIOS stands for Basic Input/Output System, and is that part of CP/M-80 which is hardware-dependent upon the particular machine on which CP/M-80 is running. The BIOS includes the routines to drive the console, disk, and if present, a printer, and perhaps a reader and punch. The BIOS has a table of JMP entries at its beginning, so other programs may refer to the appropriate entry points in BIOS without having to know exactly where they were assembled. The other programs need know only the address of the first JMP, which can be found in any running CP/M-80 system. I'll talk more about this later.

BDOS

BDOS stands for Basic Disk Operating System, and is that part of CP/M-80 which does the "logical" I/O to the console, disks, printer, reader and punch. The BIOS deals with the console as single characters, and the disks as track, sector, etc. BDOS deals with these in a higher sense – with the disk as files and a directory, and with console I/O as both characters, and strings of characters, or messages.

Console I/O

There are four distinct means of doing console character input/output in CP/M-80. They are:

(1) direct I/O through ports;
(2) BDOS calls 1, 2, and 11 (RDCON, WRCON, and CONST);
(3) direct I/O through the BIOS JMP table; and
(4) direct console I/O using the CP/M-80 2.0 (or later) or MP/M BDOS call 6 (DIRCON).

Means (1) and (2) have been covered previously, so (3) and (4) will be the primary topic of this month's tutorial.

Direct port I/O is usually performed only for non-console I/O. For example, a modem program would need to do direct I/O to the modem ports. For an external modem, it would typically execute direct I/O to a serial port.

Of the table of equates I presented in previous tutorials, this one will deal with:

```
RDCON   EQU     1   ;read console
WRCON   EQU     2   ;write console
DIRCON  EQU     6   ;direct con. I/O
CONST   EQU     11  ;console status
```

Using BDOS calls 1 (RDCON) and 2 (WRCON), is quite easy, as shown last month. However, there are some free "features" which may not be suitable for a particular application. Specifically:

RDCON waits for any character to be typed at the console. If the character is printable, or is a tab (09H), a carriage return (0DH) or a linefeed (0AH), it is echoed to the console. Further, the tab is expanded to the correct number of spaces to hit a multiple-of-8 tab stop.

WRCON sends any character to the console, but also tests for control-S to see if you want to suspend output. If you pressed control-S, BDOS doesn't return until another character is pressed. If the character pressed is a control-C, BDOS will reboot CP/M-80, and thus never return.

CONST tests if a key has been pressed, and returns an 01 in the accumulator if so. RDCON may then be used to read the character.

JMP TABLE I/O: This technique gives you greater control, but has its own drawbacks. The advantages are: (1) you have complete control over the console; (2) no testing for control-S or control-C will be done; (3) it works for both CP/M-80 1.4 and CP/M-80 2.x. The disadvantage is that it will not work under MP/M-80. This disadvantage is the primary reason that CP/M-80 2.0 came out with direct console I/O, which will be covered later in this tutorial.

You will find JMP table I/O used a lot in the various CPMUG volumes. Here is how it works:

The start of the Basic I/O System of CP/M-80 is a table of JMPs. The first few BIOS JMPS are:

```
        JMP     CBOOT   ;cold boot
WBOOTE  JMP     WBOOT   ;warm boot
        JMP     CONST   ;console status
        JMP     CONIN   ;console input
        JMP     CONOUT  ;console output
```

Since location 0 in CP/M-80 contains:

```
JMP     WBOOTE  ;to warm boot
```

we can use the address at 1 to "point to" the other entries in the JMP table. For example, if WBOOTE is at 0DA03H, then locations 0-2 contain: C3 03 DA, where C3 is the op-code for a JMP, and 03 DA is the usual low-byte-first format of the address DA03. Thus:

```
LHLD    1
```

will load HL with DA03, the address of the JMP to WBOOT. To point HL to the JMP table entry for console status, just increment HL by 3, the length of one JMP:

```
INX     H
INX     H
INX     H
```

OK, so now you have the address of the BIOS JMP to the console status routine in HL. What can you do with it? Several things. You could PCHL, which would place the value of HL into the program counter, i.e. "JMP to HL". You could also store it into a JMP or CALL statement, modifying it. Here are sample routines:

```
;
;first version of console status
;routine to use JMP table.
;
CONST   lhld    1   ;get WBOOT addr
        inx     h   ;skip to
        inx     h   ;   console
        inx     h   ;   status
        pchl        ;"jmp" to it.
```

Notice that this routine would have to be called as a subroutine, because PCHL is like a JMP, not a CALL. Thus the BIOS routine will return to the address on the top of the stack, which had better be the address of the routine that called CONST.

The routine becomes more complex if you want to save the registers:

```
;Set up local jumps to BIOS
        LHLD    1       ;WARM BOOT POINTER
        LXI     D,3     ;READY FOR ADD
        DAD     D
        SHLD    VCONST+1
        DAD     D
        SHLD    VCONIN+1
        .
        .
CONST   PUSH    B
        PUSH    D
        PUSH    H
VCONST  CALL    $-$     ;ADDR FILLED IN BY
        POP     H       ;  'INIT'
        POP     D
        POP     B
        RET
;
CONIN   PUSH    B
        PUSH    D
        PUSH    H
VCONIN  CALL    $-$     ;ADDR FILLED IN BY
        POP     H       ;  'INIT'
        POP     D
        POP     B
        RET
```

Note how the PCHL couldn't be coded in line. Instead it had to be called, so that the top of the stack would point to the POP H to which we want the routine to return. It then POPs the registers and returns to the caller.

You may not want quite such a "busy" routine, especially if you plan to use this technique for console status, console in, and console out. A way to overcome this is to use some initialization code to modify a JMP:

```
;
;alternate first version of console
;status routine to use JMP table.
;This one saves the registers.
;
CONST   push    b
        push    d
        push    h
        lhld    1   ;get WBOOT addr
        inx     h   ;skip to
        inx     h   ;   console
        inx     h   ;   status
        call    apchl
        pop     h
        pop     d
        pop     b
        ret
;
apchl   pchl        ;"jmp" to it.
```

I use "$-$" to mean something which isn't what it appears to be. For example, if I had coded "jmp 0", you might think I meant a real JMP to 0. I use $-$ as kind of an "eye catcher", implying that it will be modified before actually being used.

These same techniques may be used to set up JMPs to console status, input, and output, as this fragment of code from my disk utility, DU, shows. I don't call an "init" subroutine, instead coding the initialization in line:

```
;
;second version of console status
;routine to use JMP table.
;
;Call INIT once at the start of
;the program:
;
INIT    lhld    1       ;get WBOOT addr
        inx     h       ;skip to
        inx     h       ;       console
        inx     h       ;       status
        shld    const+1 ;modify jmp instr
        ret


;
;This is the actual console status routine:
```

(continued next page)

```
;
CONST   jmp     $-$     ;addr set by "INIT"
```

Note how I load DE with 3 (LXI D,3) and use this 3 to bump HL by 3 with a simple DAD D instruction. DU, with its mission of direct disk I/O, goes on with DAD Ds through the JMP table, thus handling the addresses for many of the BIOS JMP table entries: select drive, track, sector, read, write, etc.

Enough on JMP table I/O. Let's move on to the new direct console I/O which came about with CP/M-80 2.0:

DIRCON is a single BDOS call function (6), which supplies console status, console in, and console out. It does so by looking at the contents of the E register. A value of FF requests a combined console status and console in. If no key has been pressed, a 00 is returned in the accumulator. If a key has been pressed, the character itself is returned.

To use direct console I/O for output, put the 6 in C as above, but place the character to be output into register E, then call BDOS.

This technique has the advantage of being supported by both CP/M-80 2.x, and MP/M-80. It should probably be the standard for all future program development, so the software will be as portable as possible.

Assembler programming has great flexibility. There is seldom a single "right" way to do something. Although I have mentioned quite a few ways of performing console I/O, there are subtle variations on these which are used. For example, Microsoft MBASIC uses a technique similar to DU in that it modifies JMP instructions. However, DU changes the JMP by storing the address of the JMP table entry, which then JMPs to the actual routine. Microsoft, with its ever-watchful eye on efficiency, chose to not go through the JMP table, but instead to pick up the address from the instruction in the JMP table, thus saving the "leap frog" of JMPs. This is done via:

```
        lhld    1       ;get wboot addr
        lxi     d,4
        dad     d       ;point HL to ADDR
;                        of console status
        mov     e,m     ;get low addr
        inx     h       ;to high addr
        mov     d,m     ;get high addr
;
;DE now contains address of console status
;NOT address of the JMP in the JMP table.
;
        xchg            ;move addr to hl
        shld    const+1 ;modify the CALL
        .
        .
const   call    $-$
        .
        .
```

Yet another often-used technique is to make a "copy" of the JMP table in your program. Thus your program might contain:

```
        wboot   jmp     $-$
        const   jmp     $-$
        conin   jmp     $-$
        conout  jmp     $-$
        list    jmp     $-$
```

You initialize this JMP table by moving a copy of the BIOS JMP table into it:

```
        lhld    1       ;get wboot jmp addr
        lxi     d,wboot ;get local
;                        jmp table addr
        mvi     c,15    ;# of bytes to move
movjmp  mov     a,m     ;get byte from BIOS
        stax    d       ;modify local jmps
        inx     h       ;to next
        inx     d       ;to next
        dcr     c       ;more?
        jnz     movjmp  ;       yes, loop
        .
        .
```

Still another technique is to load DE with the "offset" to the desired BIOS entry point, and call a common routine:

```
        lxi     d,3     ;offset to const
        call    bios
        .
        .
 bios   lhld    1       ;get wboot addr
        dad     d       ;add offset
        pchl            ;go to it.
```

My point in presenting so many examples: it typically does not pay to try to memorize how a particular routine should look. If you learned a single means of doing console I/O, you might not "recognize" other techniques. You'll have to admit that with the broad range of expression presented to the assembler programmer, you really do need to analyze the code for the function it performs.

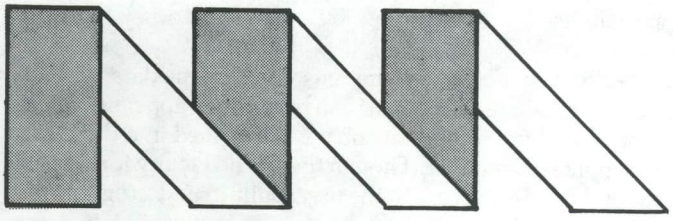
Perhaps it's now time for you to get out some of those old dusty CPMUG volumes, and look for some ASM files to browse through. See if you can spot any of these techniques, or discover yet others.

Next month, basic disk I/O, including directory searches, and making, opening, reading, writing, closing, and erasing files.

As usual, I solicit your comments, suggestions, criticisms, or questions to me *% Lifelines/The Software Magazine.*▲

(Tips & Techniques continued from page 15)

```
; PIP Patch to add '(R)eset Disks' and '(Q)uick Repeat' function
;
bdos            equ     05h     ; bdos entry address
fcb             equ     5ch     ; default file control block address
dfcb            equ     080h    ; default disk/command buffer
;
pmessg          equ     9       ; print message function
rdcbuf          equ     10      ; read console buffer function
rsetdsk         equ     13      ; reset disk system function
;
start$pip       equ     04ceh   ; normal start of pip
con$buff        equ     1ecbh   ; pip's internal console buffer
pip$cr$lf       equ     082eh   ; pip's internal cr/lf output routine
pip$prompt      equ     053ch   ; pip's command parser entry address
pip$patch       equ     096fh   ; pip gets patched at this address
;
lf              equ     0ah     ; line feed character
cr              equ     0dh     ; carriage return character
;

        org     100h
;
        jmp     begin           ; jump over INP:/OUT: vectors and EOF
;
        org     10ah
;
begin:  lda     dbuf            ; filename specified?
        ora     a               ; zero, if no filename
        lxi     d,msg1          ; and sign in please...
        cz      prnt$messg      ; print message
        jmp     start$pip       ; and off to pip
;
added:  lxi     h,con$buff      ; point to pip's console buffer
        mvi     m,128           ; set-up for 128 character command string
        xchg                    ; pointer swapped to [DE] for CP/M
        mvi     c,rdcbuf        ; read console buffer function
        call    bdos            ; let CP/M do the work
        lda     con$buff+1      ; check how many characters typed
        cpi     1               ; just one?
        jnz     save$char$cnt   ; if not, save character count and return
        lda     con$buff+2      ; get single character command
        ani     05fh            ; force to uppercase character
        cpi     'Q'             ; repeat pip function last specified?
        jnz     reset$dsk$sys   ; if not, check for reset disk system
        lhld    char$cnt        ; get character count
        shld    con$buff+1      ; stuff back to console buffer
        lxi     d,msg3          ; tell'em repeating last process
        call    prnt$messg      ; print message
        lxi     h,con$buff+1    ; point to last command entry in console buffer
        mov     c,m             ; get command length to calculate offset
        mvi     b,0             ; clean high byte bias
        inx     h               ; bump to start of command string address
        dad     b               ; add bias to locate end of string
        mvi     m,'$'           ; tag end of command string for message
        lxi     d,con$buff+2    ; point to command string for message output
        call    prnt$messg      ; print message
        mvi     c,rsetdsk       ; reset disk system function
        call    bdos            ; let CP/M do the work
        ret                     ; "Vulcan Princes...How I love You..."
                                ; - Stanley Clark, 'Return to Forever'
;
reset$dsk$sys:
;
        cpi     'R'             ; reset disk system?
        jnz     save$char$cnt   ; if not, restore character count and return
        lxi     d,msg2          ; tell'em all disk set R/W
        call    prnt$messg      ; print message
        mvi     c,rsetdsk       ; reset disk system function
        call    bdos            ; let CP/M do the work
        call    pip$cr$lf       ; do carriage return/line feed
        pop     h               ; clean the stack for pip restart
        jmp     pip$prompt      ; do pip '*' prompt and wait for command
;
save$char$cnt:
;
        lhld    con$buff+1      ; get character count + character
        shld    char$cnt        ; save character count
        ret                     ; "Welcome Back My Friends,
                                ;       To The Show That NEVER Ends..."
                                ; - Emerson, Lake & Palmer, Brain Salad Surgery
;
prnt$messg:
;
        mvi     c,pmessg        ; print message function
        call    bdos            ; let CP/M do the work
        ret                     ; "We May Never Pass This Way Again..."
                                ; - Seals and Crofts, Diamond Girl -
;
msg1:           db      cr,lf,'PIP 1.5 with (R)eset Disks and (Q)uick Repeat',cr,lf,'$'
;
msg2:           db      cr,lf,'Resetting all disks to R/W$'
;
msg3:           db      cr,lf,'Repeating:  $'
;
char$cnt        dw      0       ; console buffer character count
;
        org     pip$patch       ; patch to get to added code goes here
;
        jmp     added           ; check for reset or repeat command
;
;
;
        end
```



Notice

The July issue was placed into the mail on June 25th. If you had any problem with the timeliness of this issue, please call our Subscription Department at (212) 722-1700, or write to *Lifelines/ The Software Magazine* Subscription Department, 1651 Third Ave., New York, N.Y. 10028. We expect to place this issue, dated August 1982, into the mail around July 26th. We will print each month the date of the previous issue's mailing and would appreciate your help in tracking the deliveries.

Change of Address

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

New Address:

NAME

COMPANY

STREET ADDRESS

CITY STATE

ZIP CODE

Old Address:

NAME

COMPANY

STREET ADDRESS

CITY STATE

ZIP CODE

Feature

Applications Development Utilities: Quic-N-Easi

Joseph B. Rothstein

Quic-N-Easi (from Standard Microsystems, Inc., 136 Granite Hill Court, Langhorne, PA 19047) is an applications development language designed for screen-oriented file creation and management. It contains facilities for the design of input screens, verification of input data, quick access to particular records, and printing of reports.

A typical Quic-N-Easi application might involve the computerization of paper-based forms with subsequent quick recall to the screen of data previously entered, or Quic-N-Easi might serve as a data-entry front end to a BASIC program. For this evaluation I used a homebrew S-100 system running CP/M-80, and while still learning the system I was able to develop applications ranging from a bibliography of computer music to archiving a photograph collection.

Documentation

The manual is an impressive, loose-leaf bound volume close to an inch thick. While it generally deserves high marks for its complete and thorough coverage of the system, the physical organization of the binder leaves much to be desired. After flipping back and forth through the manual trying to locate various topics, I found it easier to rearrange many of the pages to conform to what I consider a more consistent and accessible system of organization.

The manual consists of four sections, separated by index tabs. These are:

1) Self Teaching Guide,
2) Programmer's Manual,
3) System Commands, and
4) Appendices & Supplement.

In addition, a quick reference card is supplied, with terminal-dependent commands listed on separate cards for each of the terminals supported. The user can then stick the appropriate terminal descriptor into a space provided for it on the reference card, creating a useful alternative to the manual for the experienced Quic-N-Easi programmer.

The manual's authors have wisely begun the Self Teaching Guide by describing how to make a working copy of the disk and how to configure it for the appropriate terminal. Most of the facilities of Quic-N-Easi are then covered in five lessons. Sample applications, included on the distribution disk, serve as examples to which the tutorial refers throughout.

Most of the same topics – though not the sample programs – are covered in the following section, the Programmer's Guide. Because of the level of redundancy, the user might consider reading either this section or the previous one, depending on his or her level of sophistication. The drawback to this approach is that each of the two sections contains just enough exclusive material so that reading both is required if the user is to get a complete understanding of the system.

The section on System Commands contains an alphabetized listing of each Quic-N-Easi command, with the syntax and an example of each. Once I understood the basics of the system, I found myself referring to this section more than any other. This was largely due to the unique nature of the system commands, a problem which mandated regular reference to the command syntax, and one which will be discussed in detail subsequently.

Finally, the Appendices and Supplements contain a mixed assortment of error message descriptions, reserved words, file handling considerations, configuration steps, and a discussion of the Report Writer feature. This last seems to be a recent addition to the system package, and since it is such a useful utility, it deserves a more prominent place in the manual – and will likely occupy one in subsequent printings.

References to the Appendices from within the body of the documentation can be annoying, since they do not indicate the appropriate page number and require the reader to flip pages looking for the required material. However, by rearranging some pages, I was able to overcome most of these problems.

Ease of Use

Bringing the system up using the directions in the Self Teaching Guide couldn't have been easier, particularly since the terminal I use is one of the more than a dozen which the system supports directly. Judging by the section on custom configuration, it does not appear that it would be much more difficult to configure the system for other terminals, either.

To build an application using Quic-N-Easi, the programmer first designs a form which will appear to the user on the terminal screen. This process includes designing the constant screen background, specifying the fields for user data input and their attributes, and writing programmer-defined procedures which will execute as the form is being filled in by the user. Certain procedures are invoked to display the form and store the user input, while others are invoked when a user fills in a particular field. The half-dozen applications I experimented with required anywhere from six to fifteen procedures.

Once the form design is complete, entering the data becomes almost a pleasure. The forms can be made to appear quite impressive-looking, and data entry is performed in a direct and uncomplicated manner. Though there is no facility for creating online "Help Messages" to the user, built-in and programmer-

written procedures make error trapping and data verification practically foolproof, and custom-designed error messages can help the user correct data-entry slip-ups. It is also a pleasure to find an index file recovery utility included with the package, for it alone may save hours of programming time which would otherwise be spent manually rebuilding files lost due to power failure or other unforeseen emergencies.

The restriction of record size to multiples of 128 bytes adds significant disk space overhead and is a regrettable limitation. Still, the disk access, both in retrieving records and adding new ones, is fast and efficient. Whether the programmer specifies random access by record number, or indexed access by alphanumeric key, Quic-N-Easi can retrieve or insert one record of 1000 in just a few seconds. Direct record access is considerably faster than indexed, but both appear to be within reasonable limits – though the I/O speed of any particular system will depend on such factors as clock speed and disk density.

The interactive Report Writer utility (included with the package) enables the programmer or experienced user to design and generate custom reports with minimum effort, aided by menus and prompts. This is one area in which Quic-N-Easi demonstrates a clear superiority to some of its similarly-priced competitors.

My main concern about Quic-N-Easi is the procedure definition language itself. Although entry of the screen design and procedure code is facilitated by a built-in full screen editor, the language is unique to this software product. While aspects of its syntax and form variously resemble BASIC, COBOL and some of the newer structured languages, the bottom line is that the programmer must learn yet another programming language – one which may be used only with this particular development tool. Even after designing several applications, I still found myself referring almost constantly to the System Commands section and Quick Reference Card.

Support

The folks at Standard Microsystems seem genuinely happy to offer technical advice and support to their users. Their telephone number is published in the manual; the first time I called with a question, no technical person was on hand, so someone offered to call me back at the company's expense. They seem anxious for their customers to use the product successfully, and seem willing to go to great lengths to insure it.

Conclusions

Despite the drawbacks mentioned, Quic-N-Easi has much to recommend it. For relatively straightforward applications – and particularly for computerizing paper-based forms – it is exactly what its name implies. Even a relative newcomer to programming should be able to develop impressive applications in considerably less time than it would take to write the equivalent BASIC code.

The professional programmer, though, is more likely to balk at learning yet another language, especially one which can be used only with a single software product. And while Quic-N-Easi represents a valuable addition to the growing pool of applications-development programs, the designer of complex or highly volatile applications will probably want to consider spending yet a bit more time and money on a complete relational database system. While Quic-N-Easi does an admirable job of filling out forms, its power to group fields, branch on input, or communicate between records is limited.

Still, it is a pleasure to see the emergence of software systems such as Quic-N-Easi. It does what it claims to, at a reasonable price, and occupies a distinct place along the price/performance continuum of applications-development tools. Perhaps this new generation of software products will raise the design and implementation of routine systems from the level of a black art to that of a comfortable and logical pursuit.

TABLE R1
Facts & Figures

Package or Version name: Quic-N-Easi, Version 1.4
Price: $395
Systems available for: Z 80 systems running CP/M version 2 with an addressable-cursor CRT
Required supporting software: none
Memory requirements: 48K RAM
Diskette capacity required: Two or more disk drives recommended
Utility programs provided: A Report Writer, index file recovery utility, and system configuration utility are included
Record size & type limits: Record lengths must be in multiples of 128 bytes ASCII text and delimited files supported Random access by record number, or indexed access by alphanumeric key
Portability: Files created by Quic-N-Easi may be readily accessed by programs in other languages – BASIC, FORTRAN, etc. Run-time module available for $700 license fee, covers application distribution to 10 CPU's; additional licenses $35 each
User skill level required: experienced amateur or professional programmer
System upgrade policy: All purchasers will receive notification of updates, available from Standard Microsystems for an expected cost of $10 to $30 per update

(continued next page)

TABLE R2 Qualitative Factors	Rating*
Documentation	
organization for learning	5
organization for reference	4
readability	4
includes all needed information	6
Ease of use	
initial start up	6
conversion of external data	6
application implementation	4
operator use	6
Error recovery	
from input error	7
restart from interruption	6
from data media damage	6
Support	
for initial start up	7
for system improvement	6

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

TABLE R6
APPLICATION DEVELOPMENT FACILITIES

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

Software Notes
Macros of the Month
conducted by Michael Olfe

The macro this month dissects PMATE. This editor is so pliable that it exists in many forms on my disks, with different instant commands, and I got tired of making different help files for each one to remind me what was there. This macro lists the instant command table in the PMATE on which it runs. It also provides an example of using macros to access memory directly. You could modify it to list any area of memory – user-defined instant commands, your BIOS jump table, the auto-load section of the CCP, whatever. It could be employed, for example, to insert your BIOS jump table automatically into an assembly language program you are writing. The advantage of doing this with a macro is that it gives you information about the currently running system, so is accurate for any version of PMATE or system size you are using. It has not, however, been tested on any of the 16-bit versions of PMATE. Note that the "@b=2" and "@end" directives are for the auto-load macro published in the June *Lifelines/The Software Magazine*.

```
; macro to display current "instant commands"
; @b=2

; uses buffers 1,2(execution),3,4 and registers 9 and 8

; keep house

blkb3kb4k

; get the address of the keystroke dispatch table

l6qo            ; output radix is hex
l6qi            ; input radix is hex
l22v9           ; load register 9 with hi byte of address of pointer
ble             ; use buffer l to hold pointer in hex
@@\i$           ; insert hi byte hex string into buffer
-lva9           ; debump address
@@\i$           ; insert lo byte hex string into buffer

; get size of entry from icsiz
l23v9           ; address of icsiz
@@v8

; finish translation of keytab pointer

b3e             ; i can't modify myself -- switch buffers
i0000$a         ;
c0000$^A@1$
iv9$            ; construct a command string in buffer 3
b2e
.3$             ; execute it to
                ; store keystroke dispatch table address in reg. 9


; read off the table

b4e                     ; put it in buffer 4
0aqoaqi                 ; back to decimal
[@@=255%                ; end of table
 @@\i$                  ; insert the number of the command
 va9                    ; bump pointer
 @8[i $@@i$va9]         ; display icsiz number of bytes in ASCII
 13i$                   ; next line
]

a                       ; show it

; @end
```

Feature

Developing Applications For dBASE II: T/MAKER II As A Report Generator

Steve Patchen

T/MAKER is an ideal tool for creating reports which contain a lot of calculated numbers. It also has bar graph facilities to draw horizontal bar graphs on screen or using a standard printer. Since writing complex financial projections and similar reports would require a lot of programming in dBASE II, I decided to find a means of loading dBASE II data into T/MAKER data files. This makes it easy to transfer data kept in a database to a T/MAKER report table used as a mask. The example used in this article is very simple, but it shows the essentials required for the transfer of data. This method assumes that the names of the data base fields will be used as the variable names in the T/MAKER table. The data table is computed and then bar graphs are created for the resultant totals. As an alternative for simple bar graphs of dBASE II data, I also developed a simple routine to draw bar graphs directly from dBASE II data. I used the current version of the menu system discussed in the May 1982 *Lifelines/The Software Magazine* (p.23) to organize and simplify the use of these reports.

Figure 1 shows the menu display for these reports, TMAKER.CMD and the database files associated with the selections in the menu. The current version of the menu system reserves the uppercase letters A-Z for built in commands and passes all others to a user-implemented routine. Selection 1 uses the 'B' menu command to implement a dBASE II BROWSE on the PROJECT.DBF file. PROJECT has only a single record in it with fields named to correspond to the variables in the T/MAKER PROJECT.MSK table mask. Selection 2 is used to call the TMAKER.CMD function to convert the data in PROJECT.DBF into text strings in TMAKER.DBF, and then into a text file (TMAKER.TXT) in the proper format for T/MAKER mask loading with the LOAD command. Selection 3 is then employed to create and print the report using T/MAKER. Selection 3 utilizes another new menu command, 'D'; 'D' interprets the contents of the menu's FUNCTION field as the name of a help message line containing a CP/M-80 command line. In this case 'TMAKER GET PROJECT.MSK DO' is passed to CP/M-80. The batch processing facilities of T/MAKER are used to process the report and print it.

Figure 2 shows the T/MAKER PROJECT.MSK table and the successive command lines used to process it: PROJECT.BAR, PROJECT2.BAR and PROJECT.TTL. For those of you not familiar with T/MAKER, the command DO causes the top line on the screen to be loaded into the entry buffer and executed as if the operator typed it in. The INSERT command can load more lines to the top of the screen to be similarly processed. The end of each command line inserts and executes the next line of processing until the last one is reached; it does the printing and then STOPs to return to the menu system. PROJECT.BAR inserts the bar graphs into the report and PROJECT2.BAR adds a title, cleans the report and prints it. The BAR command has to continue on one line until the END is issued to terminate the bar graph drawing. Otherwise, the BAR command continues to prompt the operator at the end of each line. The BAR command uses the 'ex' line to determine which values to print in the bar graph. The percentage column example number format begins with a '0' to tell it not to produce a BAR for this column. The prompt 'NEXT PAGE (Y/N)?' is used by T/MAKER before printing and between pages to allow printer set up and page changes.

Figure 3 contains an example printout and the parts for the dBASE II bar graph routine. Menu selection 4 provides a means to manually enter values to records in the GRAPH.DBF file. Selection 5 calls GRAPH.CMD to print a bar graph from this data. GRAPH formats the chart, prints the contents of the NAME field at the left of each bar, loads the number of bar positions to print to GLENGTH and calls BARGRAPH.CMD to overprint an 'E' ,an 'H', and an 'X' for each position in the bar. A backspace is printed before each character. After all the bars are printed, a count ruler is printed at the bottom of the graph.

In its current simple form only positive integers can be used for the bar graph count. The T/MAKER bar function can handle both positive and negative numbers and can scale the numbers automatically or by specifying MAXIMUM and MINIMUM points for the bars. If your printer does not recognize the backspace character, you will have to change BARGRAPH.CMD to print only a single character at each position. The length of the string of characters sent to print can be very long with the overprinting, which might cause some problems. This graph printed well from the program in dBASE II; however, when I printed it to a file to include in this article and then printed out a draft of the article, the bars wrapped around to the beginning of the same line at the 35th position in the bar.

Without modification, these sets of routines probably would not be directly useful for a very wide range of applications. The conversion of data to a T/MAKER data file (for a numerical report) should be planned backwards from the report. That is, you should lay out and test the T/MAKER report first using hand-entered T/MAKER data. Then plan the method for forming a dBASE II file structure which will provide the fields to be substituted into the T/MAKER report. Then modify the T/MAKER.CMD routine if required to perform the conversion. If this procedure becomes too complicated then writing the report in the dBASE language might be easier.

I will be improving the dBASE II bar graph routines to handle negative numbers for a local project. I will include the improved routines in the next revision of the DATAWARE MENU SYSTEM.

(continued next page)

Figure 1

```
*************************************************************************
*     06/12/82          DATAWARE SYSTEMS                  T1            *
*-----------------------------------------------------------------------*
*              T/MAKER REPORTS AND BAR GRAPHS                           *
*                                                                       *
*     1       Enter test data                          PROJECT          *
*     2       CONVERT data to T/MAKER data format       TMAKER          *
*     3       PRINT The T/MAKER report                  L1              *
*     4       Enter bar graph data                     GRAPH            *
*     5       PRINT THE BAR GRAPH                      GRAPH            *
*     6       GO TO T/MAKER TO CREATE A REPORT          L1              *
*     7                                                                 *
*     8       GO TO THE MENU EDITOR TO MAKE ADJUSTMENT MO               *
*     9       RETURN TO THE SYSTEM MENU                AO               *
*                                                                       *
*     >       To do a CP/M command                                      *
*     .       To do a DBASE II command                                  *
*     *       To go to a MENU not listed by entering its name           *
*     ?       To see help messages for any selection                    *
*-----------------------------------------------------------------------*
*  SELECTION         DESCRIPTION                 MENU/FUNCTION NAME     *
*************************************************************************
                  ENTER A SELECTION CHARACTER!
WAITING 1
```

```
* 06/12/82
* TMAKER.CMD
* THIS ROUTINE CONVERTS A FILE TO THE TMAKER 'name=data' FORMAT
* THE PFILE IS THE NAME OF THE FILE TO GET DATA FROM
* THE SFILE IS THE NAME USED FOR THE T/MAKER FILE AND THE LINE FORMAT FILE

SELECT SECOND
USE &SFILE
COPY STRUCT TO TEMP
USE TEMP
COPY STRUCT TO &SFILE
SELECT PRIMARY
USE &PFILE
COPY STRUCTURE EXTENDED TO TEMP
DO WHILE .NOT. EOF
    STORE 1 TO FIELDNUM
    SELECT SECOND
    USE TEMP
    DO WHILE .NOT. EOF
        STORE FIELD:NAME TO DATANAME
        DO CASE
        CASE FIELD:TYPE="N"
                STORE DATANAME-"="+STR(P.&DATANAME,FIELD:LEN,FIELD:DEC) TO CLINE
        CASE FIELD:TYPE="C" .OR. FIELD:TYPE="L"
                STORE DATANAME-"="+"'"+&DATANAME+"'" TO CLINE
        ENDCASE
        USE &SFILE
        APPEND BLANK
        REPLACE LINE WITH CLINE
        USE TEMP
        GOTO FIELDNUM
        SKIP
acl                         -           +       grw
uc2                                     +       prj       +       +       +

              INCOME

+       sales         {>PSALE#S}    {>SALES}
+       service       {>PSERV#ICE}    {>SERVI#CE}
+       consulting    {PCONSU#LT}    {CONSUL#T}
        ------------------------------------------------------------------
=+        TOTAL INCOME

                EXPENSES

+       cost of goods {PCOG }    {COG    }
+       operating cost{PEXPE#NSE}    {EXPENS#E}
        ------------------------------------------------------------------
=-       TOTAL EXPENSES

 =       PROFIT
```

```
PROJECT.BAR is:
BAR 'TOTAL INCOME' VALUES 12 MAX 99999 WIDTH 1 SPACE 0 CHAR # NUMBERS INSERT
     BAR 'TOTAL EXPENSES' CHAR * INSERT BAR PROFIT CHAR X INSERT
     END INSERT B:PROJECT2.BAR DO (note: these 3 lines should appear on one line)

PROJECT2.BAR is:
1/1 INSERT B:PROJECT.TTL CLEAN PRINT 46 0 STOP

PROJECT.TTL is:
 ----- PROJECTION INCOME AND EXPENSE REPORTS -------

The printed report is:
 ----- PROJECTION INCOME AND EXPENSE REPORTS -------
          <0 PROFIT X                                          99999>

     11100    XXXXXX
     13100    XXXXXXX
     16123    XXXXXXXXX
     20802    XXXXXXXXXXX
     28138    XXXXXXXXXXXXXXX
              <0 TOTAL EXPENSES *                              99999>

     17100    **********
     21700    ***********
     27654    ***************
     35389    ********************
     45470    *************************
              <0 TOTAL INCOME #                                99999>

     28200    ################
     34800    ##################
     43777    ###########################
     56190    ###############################
     73608    ########################################
        A GROWTH PROJECTION FOR INCOME AND EXPENSES
-------------------------------------------------------------------------
            LAST YEAR  CURRENT YEAR  CHANGE  PROJECTED
-------------------------------------------------------------------------
```

```
       INCOME
       STORE # TO FIELDNUM
    ENDDO
    SELECT PRIMARY
    SKIP
ENDDO
USE &SFILE
COPY TO &SFILE SDF DELIMITED WITH ,
RETURN


STRUCTURE FOR FILE:  PROJECT.DBF
NUMBER OF RECORDS:   00001
DATE OF LAST UPDATE: 06/12/82
PRIMARY USE DATABASE
FLD       NAME       TYPE WIDTH   DEC
001       PSALES       N    006
002       SALES        N    006
003       PSERVICE     N    006
004       SERVICE      N    006
005       PCONSULT     N    006
006       CONSULT      N    006
007       PCOG         N    006
008       COG          N    006
009       PEXPENSE     N    006
010       EXPENSE      N    006
** TOTAL **             00061


STRUCTURE FOR FILE:  TMAKER.DBF
NUMBER OF RECORDS:   00010
DATE OF LAST UPDATE: 06/12/82
PRIMARY USE DATABASE
FLD       NAME       TYPE WIDTH   DEC
001       LINE         C    077
** TOTAL **             00078
```

Figure 2

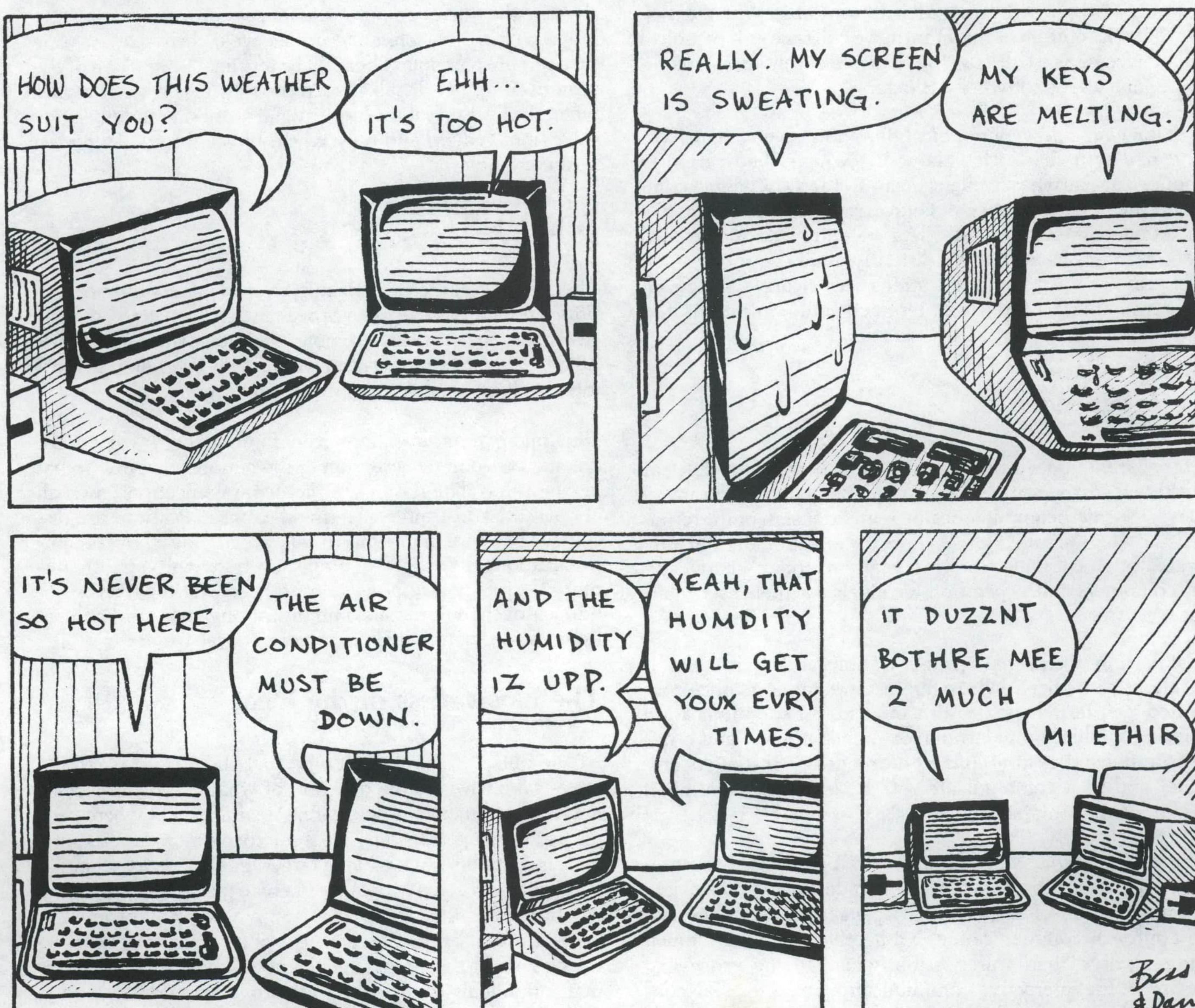
```
B:TMAKER.TXT is:

PSALES=     12000
SALES=      14000
PSERVICE=    8200
SERVICE=     8800
PCONSULT=    8000
CONSULT=    12000
PCOG=        8900
COG=        12000
PEXPENSE=    8200
EXPENSE=     9700

The T/MAKER table mask PROJECT.MSK is:

LOAD B:TMAKER.TXT COMPUTE INSERT B:PROJECT.BAR DO
              A GROWTH PROJECTION FOR INCOME AND EXPENSES
        ------------------------------------------------------------------
                    LAST YEAR  CURRENT YEAR  CHANGE  PROJECTED
        ------------------------------------------------------------------
ex                    999999      999999      0999    999999  999999  999999
zv
sales                  12000       14000        17     16333   19056   22231
service                 8200        8800         7      9444   10135   10876
consulting              8000       12000        50     18000   27000   40500
-------------------------------------------------------------------------
  TOTAL INCOME         28200       34800        23     43777   56190   73608

       EXPENSES

cost of goods           8900       12000        35     16180   21815   29414
operating cost          8200        9700        18     11474   13573   16056
-------------------------------------------------------------------------
 TOTAL EXPENSES        17100       21700        27     27654   35389   45470

 PROFIT                11100       13100        18     16123   20802   28138
```

Figure 3

```
                          BAR GRAPH CHART                    06/15/82
-------------------------------------------------------------------------
                |
RED             |XXXXXXXXXXX
                |
ORANGE          |XXXXXXXXXXXXXXXX
                |
YELLOW          |XXXXXXXXXXXXXXXXXXXX
                |
GREEN           |XXXXXXXXXXXXXXXXXXXXXXXXXXX
                |
BLUE            |XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                |
VIOLET          |XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                |
```

```
RED      |HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
         |
BLACK    |HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
         |
WHITE    |HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
         |
GREY     |HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
         |
-------------------------------------------------------------------------
COUNT    |    5    10        20        30        40        50        60        70


STRUCTURE FOR FILE:  GRAPH.DBF
NUMBER OF RECORDS:   00010
DATE OF LAST UPDATE: 06/03/82
PRIMARY USE DATABASE
FLD       NAME       TYPE WIDTH   DEC
001       NAME        C    008
002       COUNT       N    003
** TOTAL **              00012

RED          10
ORANGE       15
YELLOW       20
GREEN        25
BLUE         30
VIOLET       35
RED          40
BLACK        45
WHITE        50
GREY         55

* 06/03/82
* GRAPH.CMD
* THIS ROUTINE DRAWS HORIZONTAL LINES OF LENGTH GLENGTH CHARACTERS
* The file GRAPH has two fields, NAME is used to label each bar
* and COUNT is the number of character positions to fill in the bar

USE GRAPH
STORE T TO BACKSPACE
SET PRINT ON
? "                           BAR GRAPH CHART                         "+DATE()
? "-----------------------------------------------------------------------------------"
? "         |"
DO WHILE .NOT. EOF
        ? NAME+" |"
        STORE COUNT TO GLENGTH
        DO BARGRAPH
        SKIP
        ? "         |"
ENDDO
? "-----------------------------------------------------------------------------------"
? "COUNT    |    5    10        20        30        40        50        60        70"
RELEASE GLENGTH
EJECT
SET PRINT OFF
RETURN

* 06/03/82
* BARGRAPH.CMD
* THIS ROUTINE DRAWS A HORIZONTAL LINE OF LENGTH GLENGTH CHARACTERS
* GLENGTH is the number of positions to plot

DO WHILE GLENGTH>0
        ?? CHR(08)+'E'+CHR(08)+'H'+CHR(08)+'X'
        STORE GLENGTH-1 TO GLENGTH
ENDDO
RETURN
```

KIBITS



Feature

Ideal Microcommunications Software

Davis Foulger

In the June issue of *Lifelines/The Software Magazine* we explored microcommunications applications, the use of microcomputers for communications with terminals, remote timesharing computers and other microcomputers in "An Introduction to Microcommunications". Last month readers learned about "The Basics of Microcommunications Software", the elements that determine whether a microcommunications software package is competent. The article argued that there were two general sets of these elements. The competent microcommunications software package must, on the one hand, perform a set of operations permitting the communicating microcomputer to talk to, and be understood by, other computers. It must, however, also take advantage of the microcomputer to create a friendly interface with the user. As the series continues this month, we will take a close look at that interface as we discuss the elements of the "ideal" microcommunications software package.

Not long ago *microcommunications* wasn't even a word. No one had written articles about it. No one had seriously thought about what it might mean to the way people communicate. It had a history, of course, represented in terms like terminal communications, data communications, record communications and, more distantly, mail, but only in the last year has microcommunications been clearly distinguishable from its closest relation, terminal communications.

A Declaration Of Independence

The relative youth of the microcomputer is responsible for the recent emergence of this term and many others. For many users, the microcomputer began as an extension of the terminal and, despite solid movement in the direction of fully integrated microcomputer systems, many of today's better microcomputers still depend on terminals for their user interface.

The parting of terminal communications and microcommunications starts here. Where microcomputer users once connected their terminals to both a modem (for communication with a timesharing mainframe or minicomputer) and a microcomputer, they now connect the modem to the microcomputer and let it communicate with both the user's terminal and the remote mainframe.

This change, which brought microcomputer-based communications closer to what has been traditionally called data communications, is important; it opens the door for the user to approach communications with other computers much more flexibly than was possible when all the processing power in the interactive communications process was concentrated in a remote IBM 370. On the most elementary level, microcommunications allowed the user to reduce telephone and other mainframe access charges by performing many simple and routine tasks off-line, sending files to the mainframe only after they were finished, saving information sent by the mainframe for later examination.

These two tasks – the transmission of finished files and the saving of transmitted information – amount to a declaration of independence from the most troublesome features of mainframe computing. They are, however, only an introduction to the custom and semi-custom information processing and reprocessing capabilities that microcommunications potentially offers to users.

Because the microcomputer has its own processing capacity, it doesn't have to act like a terminal, even when treated as one by a remote computer "host". The terminal is the *slave* of the remote computer. It talks when the host tells it to. It replies when the host tells it to. It does what it does, with only a few exceptions, when, and only when, directed to do so by the host computer.

A Subtle Deception

The microcomputer breaks this tyranny by a subtle deception. Though it has its own processing capacity and could as easily play the host as the remote computer's slave, it acts like a slave when communicating with the host, fooling the host into thinking the microcomputer is a terminal.

Here microcommunications comes into its own. For once it has succeeded in its deception of the remote computer (what last month's "The Basics Of Microcommunications" was all about), the constraints of terminal communications are dissolved. Now we can begin to talk about "ideal" microcommunications software – communications software that is more flexible and responsive to the needs of the microcommunicator than terminal communications, data communications, record communications or mail ever were.

The Elusiveness of the Ideal

At the outset it should be noted that there is no fixed and universal answer to the question of what comprises "ideal" microcommunications software. As in any field where people aspire to "perfection", ideal microcommunications software is something of a moving target. Its elusiveness comes in part because no two people are likely to pick exactly the same attributes for an "ideal" system, making compromise a necessary part of the development of any mass market software. That elusiveness is heightened, however, by the fact that individuals change their ideas over time. Hence, software that seems fantastic today may be antiquated tomorrow.

Although final answers to the question of what constitutes "ideal" microcommunications software will remain elusive, there is still much to be gained from an attempt to answer the question. Microcomputer software has steadily improved over the last few years, but most microcomputer software could bear more improvement.

Software developers are not to be faulted for this. Some of the developers' achievements, within the constraints of 64K bytes of memory, have been truly astonishing. Still, 64K has put a firm upper limit on what a package can do without long delays as disks are accessed (and occasionally shuffled) for program overlays and sometimes continuous data read/write operations.

New Opportunities For Building Better Software

That limitation is disappearing, however, in the face of an increasing range of flexible hardware options that promise to change the way we think about microcomputer software over the next few years. These options include hard disk storage at prices approaching $200 per megabyte, the increasing availability of *RAM disks* and the rapidly falling prices of bubble memory, which promise faster throughput in many microcomputer applications.

The more important opportunities, however, are coming from the widespread commercial availability of microcomputers, like the IBM Personal Computer and the Radio Shack Model 16, that can address large amounts of RAM. These computers allow software developers to be far more imaginative. Word processors will be able to process better. Program languages will be more complex. Spreadsheet packages will become more talented. Relational database management systems will become more like their big brothers in the mainframe world.

These software applications will be (and in fact already are being), integrated into comprehensive business software packages. Where packages are not integrated, multitasking (concurrent) operating systems will allow users to run several programs and/or applications on the same computer simultaneously.

Few classes of microcomputer software could stand more improvement, as a group, than microcommunications software. Most visibly, much microcommunications software tends to do very little more than act like a terminal. In general, these packages fit in a small fraction of the RAM available in the user's machine. Rarely, however, do they take much advantage of the microcomputer, beyond giving the user the opportunity to save files received from a remote mainframe and send files prepared off-line.

Although capable of sending and receiving information used by other software packages (including word processors, spreadsheet programs and databases), microcommunications software is almost never able to work with (or even in parallel with), those software packages. Users are forced into playing the floppy shuffle, continually moving between word processor, spreadsheet and communications program. Only careful planning stands between the user and total frustration.

For the most part, moreover, the way microcommunications software presents information directly to users reflects the way information is passed between machines. Characters are displayed serially, a circumstance that leads to gibberish when two users type at the same time. It doesn't have to be this way.

If we carefully examine the ways in which we use microcommunications, microcomputers, and the product of microcommunications (information) in other microcomputer applications, we can go a long way toward specifying the kinds of features that would improve microcommunications software packages.

Microcommunications and Microcomputers

The only reason I would be interested in the use of a microcommunications package is my use of a microcomputer. Few users would buy a microcomputer just to allow them to talk with other computers. Although there are many advantages to using a microcomputer in data communications applications, microcomputers cost significantly more than terminals; terminals should be favored in most straight data communications applications.

The microcomputer allows me to perform data processing and information handling tasks without communicating with a remote mainframe, permitting much better control of software and applications.

The microcomputer also gets me out of the portability trap. Over the last ten years I've used six different mainframes (and three minis) at five different universities (I taught at some of them). Every time I changed schools, my software changed, too. For some programs the changes involved little more than a new set of JCL (Job Control Language; the mainframe computer equivalent of an operating system) cards. At others, whole decks of computer cards had to be repunched; whole programs had to be rewritten, whole tapes of data converted to a new format.

Finally, the microcomputer means increased (and more reliable) computer access. Microcomputers rarely "crash". I never have to worry about a line for the terminal. Run time never suffers because of a system overload. Granted, things don't always run as quickly as I might like, but I never have to wait several hours for the mainframe to clear its way through a giant morass of high priority (i.e. - small and/or administrative) programs either.

Not A Communications Machine

I bought my microcomputer for word processing, spreadsheet mathematics, personalized and secure data base management and customized statistical programming. In other words, I bought my microcomputer for the same reasons that most businessmen do. I wanted a tool that would

(continued next page)

increase my productivity in dealing with a wide range of information.

For me, data communications is an add-on, an important add-on that was planned into the machine from the day I first started planning to buy a computer, but never the central reason for purchasing the machine. Only 5% of the time I spend at a microcomputer is spent communicating with other computers. By contrast, at least 50% of that time is spent word processing, with lesser amounts of time devoted to programming, statistics, and other applications.

When I microcommunicate, I generally do so for one of three reasons. Most commonly, I am seeking information from remote information sources - news, stock prices, airline schedules, bibliographies, etc. I also frequently communicate with others through my computer in direct electronic "conversations" or asynchronous "computer conferences" and "electronic mail". Less frequently, I access remote time-sharing services to use services otherwise unavailable to me.

I would like to broaden the range of my microcommunications, particularly now that other members of my family are considering getting computers, by setting up my computer with some "bulletin board" software that would allow family and friends to leave messages on (and collect them from) my computer. I already own the necessary auto-answer modem, but I use the machine too much for word processing and other tasks to allow (with my current operating system) much time for communicating.

Integrating Microcommunications With Other Tasks

I present my use of microcommunications and my microcomputer as an example because I think that I may be typical of many microcommunicators. Owning the microcomputer has freed us from the necessity of communicating with other computers from a terminal. We now communicate by choice and, more often than not, in the context of some other microcomputer project. Microcommunications represents a small, although important, part of our computing activities, and we wish to microcommunicate quickly, preferably without interrupting the progress of other tasks.

Microcommunications software doesn't need to stand on its own (even in 64K byte microcomputers), so there really isn't any reason why a person shouldn't be able to use microcommunications in the context of those other activities. If a person uses database, spreadsheet, word processing or other programs, there is no reason (particularly on new high capacity machines like the IBM Personal Computer and the Radio Shack Model 16) why microcommunications shouldn't integrate with those programs.

Microcommunications software's most important need, if my experience is a reasonable guide, is integration with the other tasks we perform at a microcomputer. If, for instance, we make extensive use of microcommunications while word processing, our microcommunications package must be able to operate in conjunction with that word processing package. If, moreover, that use is in conjunction with collecting numbers for stock market analysis, our microcommunications package must work with our stock market analysis package.

There are three ways of achieving this kind of integration, each of which should be considered carefully by software developers. The first and simplest way to achieve this integration is through the operating system. Today's microcomputer operating systems are, for the most part, single user and single-tasking ones. Indeed, the description fits both CP/M-80 (the standard of the eight-bit microcomputer world) and SB-86 (MS-DOS, PC-DOS - the current standard of the new 16-bit microcomputer world).

A single-user, single-tasking microcomputer operating system allows only one person to use a microcomputer at a time, and permits that user to run only one software package at a time. When this kind of operating system is in use, one cannot, for instance, program in both BASIC and Pascal on the same time, or jump over to a spreadsheet program to check some figures while in the middle of a word processing task.

The Concurrent Processing Option

A new kind of single user operating system is emerging in the 16-bit microcomputer world, however. Called multiprocessing or concurrent, these systems allow the user of a machine with extended memory capacity (like the IBM PC) to run more than one piece of software at the same time. Two packages that promise this kind of concurrent operation are already available on the IBM PC (although neither is available through IBM's software central). Both (and others) will doubtless be available for other supermicrocomputers in the future.

The current offerings include Concurrent CP/M-86 and Qunix, two products that, because of their importance to the ways in which people will be using microcomputers in the future, will most likely to be reviewed in future issues of *Lifelines/The Software Magazine*. The importance of concurrent processing is in the freedom it will give users to move between tasks flexibly. Depending, of course, on the capability of the concurrent processing system, the demands of the applications software being used, and the amount of addressable memory in the user's machine, it may also be possible to run a word processing package, a database package, a time management and accounting package, a spreadsheet package and a communications package.

No user may ever want to juggle so many software balls simultaneously, but the freedom to do such juggling would represent an important leap in the productivity potential of the microcomputer. The major advantage the concurrent approach gives users, when compared with approaches we will mention below, is the user's freedom to select any software they prefer for any given application. One would not be tied to WordStar because one happened to like CalcStar.

The concurrent approach is not without problems, however. Although the user would be able to move freely between different products, those pieces of software would not be linked, and there would be no guarantee that information would be easily moved from one to another. Reprocessing would not

be implicit to the the receipt of information, destined for the spreadsheet, that was received in the microcommunications package. Reprocessing through a third program might be necessary to move information from the microcommunications interface to the preferred word processor.

Patching And Linking

A second route to the integration of microcommunications software with other applications software is a rather more direct approach that may prove useful in both single and multi-tasking operating systems. This approach, which we will call patching and linking, adds microcommunications capabilities to whatever applications package a user prefers (or, alternately, adds an application to a microcommunications package).

There is nothing new about patching new capabilities into old software. Such patching may be conceived of and installed entirely by the user, implemented and sold as an enhancement by a third party, or issued as an upgrade by the original developer.

The software protection that has become a way of life in the commercial software world (even if you can make a backup copy, you cannot easily get at the code) reduces the practicality of user-conceived and installed patches for most people. The possibility of enhancement and upgrade remains an important option, however, as we look to integrating microcommunications with other software applications.

There is no reason why a microcommunications option couldn't be patched into many of today's software packages. Such patches would give users of many packages the ability to quickly switch in and out of a microcommunications mode without losing track of their work. These patches are particularly important in the 8-bit microcomputer world, where the addressing space needed for invoking concurrent processing or developing fully-integrated software packages is limited. Indeed, these patches may be the best shot many 8-bit microcomputer users have at integrating microcommunications with other functions.

In the 16-bit world of concurrent processing, such patches are best implemented as *links* that give each of several applications packages a path to a central microcommunications package. This link would provide for the movement (and, if necessary, the reprocessing) of data between the microcommunications package and each of the applications that the user was working with, taking advantage of concurrence while providing the link between applications otherwise lacking in the multi-tasking system. An added benefit of such a link is that it would ease the flow of data between applications (sending data to the microcommunications interface does not require that it exit the RS232 interface).

But a patch is, after all, a patch. It may fix a problem or add functionality, but it will generally have to stretch the underlying concept of the existing software to do so. The application may be slowed. The patch may never work quite right. The reprocessing of data may be slower than might be preferred. Even though microcommunications might be integrated with a person's most often used application, that application may remain entirely distinct from others.

Multi-Tasking Applications Software

The third solution is to write software that integrates these functions from the start. In my opinion, the best example of such software on the market today is that which runs the Xerox Star professional workstation. Star's software integrates word processing, time management and accounting, database, calculator, com munications and many other functions into a single integrated package. The display of Star can be separated into windows, with each window devoted to a different task, and other tasks operating in background without even being seen on the screen.

Star costs in excess of $18,000, more than many would pay for a microcomputer system, but the fact is that Star *is* a microcomputer. It is a special microcomputer with advanced functions, but almost all of those functions are in the software. The fact is that the hardware configuration of the Star can be duplicated almost to the letter in the IBM Personal Computer (even the resolution of the screen can be duplicated if someone takes to designing the hardware into assigned memory space) for a price that would probably fall below $10,000.

In the microcomputer market, capability almost universally translates into software, and software that integrates applications on the IBM Personal Computer is already coming to market. Context Management Systems (Torrance, CA), for instance, has introduced an integrated software package for the IBM PC called MBA that integrates word processing, spreadsheet processing, database processing and dot image graphics in a single package (Context asserts that microcommunications will be added in the fall). Other packages, some of which will be examined in next month's overview of existing microcommunications software, are also coming to market.

We hope to review Context MBA in a coming issue of *Lifelines/The Software Magazine*, as it represents an important new direction in software development. Xerox is a current leader in this area; with other companies entering the fray, however, the multi-tasking microcomputer applications software market promises to heat up fast.

The advantage of the integrated software package is twofold. First, and most important, it appeals to the buyer because the purchase of a second piece of software is not required. This may seem a small point, but as more and more non-programmers enter the microcomputer market looking for productivity tools, the "all-in-one" approach will be a very appealing one, particularly when the purchaser learns about the second advantage of integrated software.

That second advantage is in the ease with which data will move from one application to another. To work well, an integrated software package will have to use an integrated data model. In practice, the entire package will probably be built on top of a relational database (as appears to be the case in Context MBA and Xerox Star software) that keeps track of data types and keeps data consistent from one application to

(continued next page)

another. There is no need, in such packages, to print a VisiCalc file to disk in order to read it into WordStar. There is no need to write a report from a database in DIF format in order to use databased data in VisiCalc.

People who don't care how their system works (so long as it does what they want it to) will be easily sold by the pitch for integrated software, and such software will doubtless sell a lot of computers (just as WordStar and VisiCalc have). Multitasking applications software will not be a panacea, however. There will doubtless be trade offs in the integrated software world. The user who buys an integrated package with a poor word processor because he likes the other features will be stuck with that word processor.

Despite its disadvantages in the area of linking information between different applications, the concurrent approach does give the user the option of picking the strongest software in each application area. This is an advantage that experienced users will note and, in most cases, take advantage of. Although I cannot see using an integrated software package in the near future, I suspect that it could be used within a concurrent environment that allows me to do things that I can't do in the integrated software package (like run the software I'm reviewing while I write it up on a favorite word processor).

The Functions That Should Be Integrated

Regardless of the approach to the integration of other microcomputer applications with microcommunications software, such software integration should improve overall productivity by speeding the move between one task and another. The range of tasks to be integrated will depend largely on the user's needs and interests, but the following represent a partial list of functions that will be regularly integrated into microcommunications software packages, along with the reasons why.

Time Management And Accounting: Most professionals will use microcommunications for business functions, and will need to keep track of what microcommunication time was spent on what tasks and against whom the time should be billed. Those who don't have to do it will find that the option will help them to better understand how they use their time, and how they can use their time better.

Database and Spreadsheet Management: Large numbers of users employ microcommunications to keep on top of the stock market and other number-oriented systems and data. Many of these users will want data directly reprocessed into a database or spreadsheet and may even want a routine analysis of that information.

Word Processing: One of the most valuable functions of microcommunications is in the ability it gives users to quickly send and receive drafts of documents. The ability to microcommunicate directly from a word processor would speed this process, enabling users to edit information almost as fast as it is received; to send information almost as fast as it was written or edited.

Telephone Directory/Automatic Communications Control: Use of an autodial modem enables the use of microcomputer as a superpowerful auto-dialer that could hold thousands of names and telephone numbers for instant access. Dialing out through the computer might automatically invoke call management software that keeps track of calls, and time accounting software to automatically take care of billing. Used in conjunction with a clock, such software would also mean that certain numbers could be automatically called when rates are low (when the user is asleep, for instance), allowing the user to cut communications costs.

Local Electronic Mailbox Software: Like the Bulletin Board software that was discussed in the June *Lifelines*' "An Introduction to Microcommunications", such software would allow friends and family to call a user's mailbox to send and receive messages. It will also allow users with portable terminals or computers to call the microcomputer when away from home or office.

Microcommunications Games: Tying microcommunications to microcomputer games opens the door to a whole new way of playing games against remotely-located opponents, while using the microcomputer as an assistant. In such games, strategy would become more important than reflexes, particularly when users programmed the microcomputer for assistance themselves.

Improving Microcommunications As Microcommunications

While integration of Microcommunications Software with other applications will be an important step in producing the ideal microcommunications software package, much can be done to improve the way we build microcommunications software as well. These improvements, many necessary in a competent microcommunications package, are intended to make microcommunications easier for the user while opening the door to wholly new ways of communicating.

We have a tradition of developing microcommunications software that delivers each character to the screen as it is input by the user or received by the remote computer. This kind of *serial* display is very practical from the developer's perspective, because it closely models the ways in which a microcommunications package must communicate with a remote computer. It is not always in the best interest of the user, however, who must, if gibberish is to be avoided, studiously orient his data input with the other user, avoiding typing at the same time that the other user is.

There is, of course, little reason for this, as there is enough flexibility in the commonly-used full duplex asynchronous communications protocols to allow the microcomputer to keep track of who says what when. So, we are able to escape the bounds of the way the microcomputer must talk to the remote computer. There is no rule that says that the way a microcommunications software package presents information to a user must reflect how it talks to a remote computer.

As we noted at the end of the July *Lifelines/The Software Magazine*'s "The Basics Of Microcommunications Software", screens can be split into two or more windows (windows on a computer display are like windows on a

(continued on page 47)

Feature

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

Ward Christensen

It's time for a change of pace in these editor reviews – but perhaps I should say a change in "price." PRECISE is priced at $35, from Berkeley Cottage Software, 1434 Parker St., Berkeley, CA, 94702. I gave the price to get your attention, and the address, because I have only seen an ad for it in the "Computer Shopper." I have been very pleased with the authors' response to my requests for help, down to what must have been a major re-write to support my current terminal, which is ADM-31 compatible.

When you read this review, keep the $35 in mind. If I had paid $150 for PRECISE, I might be painting a much less favorable picture. It is not in the same league as the "big time" editors, but there are some ideas they might get from PRECISE, and a few PRECISE could take from them.

The major drawback about PRECISE that set it apart from all the others reviewed so far, is that the file *must fit in memory*.

Evaluation

DOCUMENTATION: The manual is 22 pages long. It is sufficient. It includes a nice one-page command summary.

SPEED: PRECISE is quite slow if you are used to better editors. For example, a simple "replace all" in this article (replace all period-space with period-space-space) took an unbearably long time. I would say it replaced about two per second. There were perhaps 800 to replace. WordMaster, for example, would have taken about ten seconds.

Like VEDIT, preemptive scrolling (interrupted scrolling when you start another) is not implemented. For $35 I guess I could be convinced to tolerate this lack.

Hardware line-insert and line-delete are supported, and make small scrolls very nice, handy when you use keyboard controls for inserting or deleting lines.

ERGONOMICS: The default keyboard layout is "geometric." The control characters A, S, W, and Z, are cursor left, right, up, and down. PRECISE customizability allows you to make it "anything you want."

CONFIGURABILITY: PRECISE arrives with only one file on the disk: CPRECISE.COM. It supports the following terminal types: Heathkit H19, Soroc IQ-120, IBM 3101, ADM-3A, ADM-31, ADM-42, Hazeltine 1500, Hazeltine 1510, Hazeltine 1520, DEC VT-52, Televideo 912, Televideo 920, Visual 200, Microbee, and IMSAI VIO.

In addition, if you don't have one of these terminals, you can key in the hex values for the various terminal functions: initialize terminal, clear screen, cursor positioning, offset for each cursor positioning character, whether column is output before line, erase to end of line, erase character on screen, erase to end of screen, insert line, delete character, insert character, turn on insert mode, turn off insert mode, delete line.

You may also configure PRECISE to: (1) verify output file write; (2) upshift before comparision; (3) insert tabs between single quotes; (4) insert tabs between double quotes.

In configuring your keyboard, PRECISE is the most flexible of the configurable editors I have tried. You may specify the characters to use in any of the following ways:

a) hex value
b) ↑ followed by letter
c) control-character

The specific single-key full screen controls supported are: cursor left, cursor right, cursor up, cursor down, scroll up one line, scroll down one line, scroll up one screen, scroll down one screen, insert line above, insert line below, delete line, restore current line, delete character under cursor, insert character(s), forward word tab, backward word tab, tab, delete from cursor to end of line, move cursor to start of file, move cursor to end of file, search for string, search for next occurrence, home cursor on this page, move block to cursor position, copy block to cursor position, repeat, access to utility commands, automatic scrolling, character to substitute for tab in TN mode, and escape to utility commands.

Apparently with an eye to supplying "total configurability", the two-letter commands used in command mode may also be changed. They are: verify all disk writes, display directory, expand tabs in file, don't expand tabs, change display mode, use default tabs, use user-set tabs, set user's tab stops, make cursor position start of file, make cursor position end of file, change block delimiter character, delete block delimiters, kill block, complement 'literal match', delete all blank lines in the file, insert file at current cursor location, replace all occurrences of string, replace strings selectively, output block or whole file, upshift all characters in block, count number of lines in file, reset disk system, exit editor writing file to disk, exit editor without writing file to disk.

You cannot customize to accommodate the multi-character sequences that most function keys generate.

EASY TO LEARN: The video-related commands are pretty straightforward, although the way in which many of the video commands work differs significantly from the fashion in which the other reviewed editors function. See "FULL SCREEN" below.

B.C.S. says that future versions of PRECISE will implement a HELP function. I don't know if this will be done with a built-in table or with a file on disk, but it will help you work without the manual.

(continued next page)

There is no relation between CP/M-80's ED and PRECISE. PRECISE has no real command mode in the sense of having command macros. However, the minimal requirements: search and change, and a few "niceties" like being able to uppercase a block of text, are present.

Objective Evaluation

A. Video-Related Criteria

FULL SCREEN: PRECISE has two distinct modes: a full screen mode, and a command mode. In full screen mode, there are keys to move the cursor one line up or down. The keys to go ahead or back one character or word behave very strangely, in my opinion; they do not leave the line you're currently on. This is frustrating.

To go into that in a bit more detail: PRECISE acts like a full screen *line* editor. For example, you cannot split or join lines of text. Also, if you press "character delete" on a blank line, most editors delete the "character" consisting of the CR/LF, i.e. delete a line. PRECISE doesn't do this. Depending upon whether you typed "delete character left" or "delete character right", PRECISE either ignores your command, or considers the line to be suddenly filled with spaces, and deletes the last one, positioning the cursor in column 79.

Sadly lacking are cursor keys to go to the front or to the end of the line. I use "word backtab" from the front of the line to go to the beginning of the last word on the line, and then cursor over to the end of the line.

Typing too long a line, or inserting text at the beginning of an already long line, causes data at the end of the line to be lost. Not good.

SCROLLING: There are keys to scroll up or down one line, or a screen full. The scrolling doesn't preempt itself, so repeated scrolls burden you with seeing all the intermediate lines or screens.

The "automatic scrolling", mentioned above under configurability, is interesting. You press the auto-scroll key, then press either the scroll up, or the scroll down (line) key. The display keeps scrolling until interrupted. The manual says ESC or ↑@ work, but I found ESC didn't. Sorry to say, that even with the hardware line-insert used when you manually scroll a line at a time, the auto-scroll completely redraws the screen. I thus consider this feature useful only on a memory-mapped terminal. I must presume this is one feature that didn't get changed when PRECISE was converted from its "memory-mapped heritage".

A good feature: terminal hardware line-insert and line-delete are supported, making small scroll amounts very nice (on supporting hardware).

INSERT: PRECISE has an insert mode.

OVERTYPE: PRECISE has an overtype mode.

UNDO-KEY: If you screw up a line, PRECISE has automatically saved it for you, and puts the original back upon your command.

REPEAT: Fairly nicely implemented, and general-purpose. When you press the repeat key, you *must* follow it with a number. PRECISE then remembers this number, and repeats the following keystroke for you that many times. Thus to draw a line of 60 dashes in your file, press "repeat/6/0/-". Only MINCE, which doesn't require the entry of a number, has a nicer repeat key.

TEXT EDITING ABILITIES: PRECISE can't handle word-wrap, nor some pretty basic text-editing needs such as splitting lines apart, combining lines, repeating lines, etc. If you had any text work to do, you would need to have some sort of text processor such as TEX, or the TSC formatter, to put things back in order.

PRECISE also has the ability to print, using a "marked block" technique, but the total lack of other formatting limits the usefulness of this.

B. Command-Related Criteria

MOVE: You may move to the top or bottom of the file only by full-screen commands.

DELETE: Characters may be deleted in full screen mode. Also, using a user-selected block delimiter character (default is back-slash), you may delimit a block and issue the "KB" (kill block) command to delete it.

INSERT: Insertion must be performed in full-screen mode. If you should accidentally type any control character for which you have not configured a function, PRECISE places the specified control character in the file, and moves the cursor right. The control character remains invisible - this is not good.

TYPE: N/A.

CHANGE: Both a "replace all" and a "query replace all" are supported. As I noted under speed, the "replace all" is quite slow. "Query replace all" clears the screen, and shows you the applicable lines, one at a time, with an up-arrow under the location at which the change would be made.

MOVE and COPY: These are implemented by using the block delimiter character and control keys to either move or copy the delimited block.

COMMAND STRINGS: PRECISE is not able to perform generalized tasks in non-full screen mode, like searching for the next blank, and changing it to a carriage return, etc. However, some commands are supported in non-video mode, so I'll address them later on.

C. File-Related Criteria

BACKUP: PRECISE does not create backups. It loads the file into memory, and writes it back on top of the original when it ends. You could use the "OF" (Output File) command to write your own .BAK whenever you begin an edit.

SAVE: Not explicitly supported. Again, the OF command could be used to write the file to disk.

QUIT: EU quits PRECISE. There is no "quit but stay in PRECISE", à la the "O" command of ED or WordMaster.

READ: To insert a file from disk into the one you are editing, you may use "IFfilename".

WRITE: "OFfilename" writes the delimited block, or if none, the entire file, to "filename".

DIRECTORY: Getting a directory list is supported very nicely. PRECISE prints it 4-columns up, and properly calculates the number of free spaces left on the disk, for any disk.

OTHER: PRECISE has a few features that don't fit the above topics. It can reset the disk system, allowing you to change disks. It can erase files on disk. It can verify disk writes. It can handle various tab expansions – either within single or double quotes or both or neither. It can report on file size, reporting size, bytes free (very important, since the file must fit in memory), and output file name. It also supports user tab stops.

Statistics

See opening paragraph.

Room For Improvements

At $35, PRECISE is unquestionably a good bargain. Criticizing PRECISE only invites the author to put a lot more time into it, and probably raise the price. However, I will still cover the shortcomings, to help you decide whether PRECISE, with its current abilities and its current price, can fill your needs.

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

(2) Auto-scroll should support line insert/delete.

(3) Lines should be able to be combined or split. More generally, carriage returns should be handled as "just another character".

(4) Tabs should not automatically be inserted into text, even where "appropriate". The user should have control over this.

(5) Buffer to disk should be supported so the file doesn't have to fit in memory.

(6) Allow the configuration process to be based upon a previous copy of PRECISE, so you can make minor changes to the control characters, without having to re-key all of them in. (Note that the ability to press control-whatever does make this process quite fast, if you write down the order in which you have selected the keys. A new configuration then takes only a couple of minutes.)

(7) When using the "scroll one line", I'd prefer to have the cursor move with the line of the file, rather than staying on the same line of the screen. I found myself always doing five scrolls, then five cursor-ups.

(8) There should be an option as to whether the cursor should be able to move into "empty space". It defaults to being able to. That is a very seldom-needed requirement. Attempting to use control-I tabs takes some getting used to. If you press tab on a new line, a single tab is inserted into the buffer. However, if you then "cursor backspace", PRECISE backs, not over the tab like the other editors I have seen, but rather back only one space. If you then press any key, PRECISE fills in the proper number of spaces.

(9) Control characters in the file screw PRECISE up. They should either show up as "↑" then a character, or perhaps should be "filtered out" when the file is read in. Also, "hi bit" carriage returns totally mess it up.

To summarize, there is nothing basically wrong with PRECISE when you consider what you get versus what you pay.

Bugs

I did not discover any real "bugs", but came "close" in some of the areas mentioned above – such as PRECISE's handling of tabs and files with "funny" characters in them.

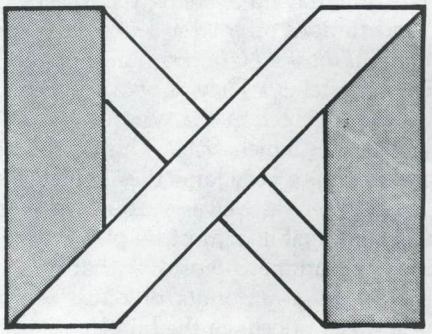
Conclusions

If you are still using ED for your ASM program editing, and just don't have the money for a good full screen editor such as WordMaster or PMATE, you owe it to yourself to scrape together $35 and buy PRECISE. Remember the limit of "must fit in memory".

More low-cost editors coming up in the months to come. Also, look for Mr. EDit: an editor that is packaged quite uniquely.

I greatly appreciate the comments and the offers to review additional editors that I have received. Thank you. I seem to have a permanent job reviewing editors. Candidly, I am amazed at how they vary. No two are very much alike. The fact that there are so many makes it clear that a "summary editor review" may not be practical. However, at some point, I would like to put together the best points of each, and blue-sky a bit on my opinion of "the perfect editor". Perhaps some ambitious programmer or company would make it happen, or enhance one of the existing editors. For example, WordMaster is a long-time favorite because of its adequate function and speed, but PMATE is what I usually use because I love to "hack" time-saving little programs in it. However, PMATE could be (1) faster, and (2) have a good repeat key.

Keep those comments coming in % *Lifelines/The Software Magazine*.



Feature

File Archiving and Backup for CP/M-80, CP/M-86, and MP/M-II

Kelly Smith and Kim DeWindt

Archival storage and data file backup are important considerations for today's microcomputer user. The day-to-day process of creating and updating large data bases, maintaining current records, and just "plain'ol" file maintenance is at the heart of most computer systems – a fact which is especially true for users who maintain their accounts payable/receivable or inventory control on such systems. These data-intensive systems cannot bear the loss of two or three days out of a file's history, much less the loss of all data ever acquired! Ideally, the user should not be encumbered with the tedious process of keeping track of these additions and changes. . . it should be as automatic as possible.

As an engineering programmer at Callan Data Systems, I have watched most of our non-engineering staff fight their way through the (not so interactive) dialog of CP/M-80. It became *very* clear that users of the systems were less than adept at computer etiquette (i.e., push-button resets of the systems during disk I/O, removing diskettes during directory writes, etc., etc.). It was even harder for them to remember all of the CP/M-80 jargon ("ahhh, errr. . . what does PIP stand for again?", and "wadi'ya mean "See colon equal two Bee colon star dot star"???"). Get the picture? But let's face it, CP/M-80 was written of the programmers, for the programmers, and by the programmers, definitely *not* for the non-technical person. But, the sophisticated data files and programs being developed on today's CP/M-80 systems, (and being maintained by clerical personnel) *demand* that the graceful archival of data is handled in the least painful manner possible.

The Callan Data Systems' CD100M CP/M-80 and CP/M-86 Workstations offer what I call the "Ultimate Single User System". (*Editor's Note*: Both authors are employees of Callan Data Systems.) They support any mix of 5 inch (604 Kilobyte) and 8 inch media types as well as 5.5 and 13 Megabyte Winchester 5 inch disks. This allows the instant interchange and access of very large files, which is excellent for software development as well as the business environment. But all this storage capability must be properly maintained in either in the programmer's world or that of the operator; otherwise loss of large amounts of data (meaning: "many days of work") can occur in the blink of an eye. With this in mind, let's give everybody a break, and interface CP/M-80 to the real world, starting with file archiving and backup.

An Archiving Implementation

Digital Research's CP/M-86 and MP/M-II Operating Systems incorporate a file attribute to support the archival back-up of any file addition or update of a disk directory. This attribute is reset when an application program writes a file to a disk. This archive attribute is the most significant bit of the third character of the FCB (File Control Block) directory entry i.e., position 11, identified as byte position T3. Therefore, the Archive attribute bit is identified as bit 7 of byte T3, alias T3'). It is a simple matter to identify a file that has been archived (e.g., backed-up). When T3' is a one, the file has been archived (backed-up). The file has not been archived, (e.g., it needs to be backed-up) when T3' is a zero.

The not-so-simple part is writing the utility that is required to keep track of the archive attributes on a disk. Enter ARCHIVE, which works with CP/M-86, MP/M-II, as well as with CP/M-80 (with a little help from a programming patch).

ARCHIVE – A File Archiving Utility

ARCHIVE is a user-oriented disk maintenance utility for archival storage and file backup. User facilities include the ability to selectively 'tag' files for archiving (or for not archiving), display file archive attributes, or backup non-archive files to a selected disk. Wild-card filenames (using '*' and/or '?') may be used freely to select some or all (i.e., '*.*') files for archiving.

ARCHIVE – Examples of Usage

At the CP/M-80 user command prompt, (usually 'A>'), the user has the option of invoking four, single letter ARCHIVE command directives, as follows:

S– Set file archive attribute.
R – Reset file archive attribute.
D – Display file archive attribute.
B – Backup non-archived files.

ARCHIVE is invoked by the user at the CP/M-80 command level by typing:

```
A>ARCHIVE (filename.typ) (option)<cr>
```

Filename.typ is an ambiguous or un-ambiguous filename and filetype, followed by any one of the single character options, and then by a keyboard RETURN entry. An actual example might be:

```
A>archive b:*.* b<cr>
```

We are executing ARCHIVE from the 'A' disk drive. All filenames and filetypes on disk 'B' whose archive attribute has been 'reset', are to be 'backed-up' to some (as yet) indeterminate destination disk.

Options 'S', 'R' and 'D' (Set, Reset, and Display) when invoked, will cause each file found in the directory, that matches the specified filename(s), to be displayed by multiple extension; that is, each directory entry will be displayed, with its extent number, followed by the archive attribute action/status as 'S' or 'R' (Set or Reset).

Option 'B' (Backup) will request the user to enter a destination disk letter (A to P); then ARCHIVE automatically scans the source disk directory for 'reset' archive file entries. Any 'reset' entries will be 'set' (indicating that the file has been archived); then these files will be copied to the destination disk. (Note however, that these files will not retain any previously set attributes of 'Read Only' or 'System' had they been set; all files copied to the destination disk will be 'tagged' as archived only). If a disk becomes full, the user will be requested to remove the disk from the destination drive and insert another for completion of the backup process. Any file not completely copied to the destination disk at the time that the destination disk is detected as 'full', will be deleted from the destination disk, and the file copy will resume intact on the next disk installed in the destination drive.

Users of CP/M-86 and MP/M-II can use ARCHIVE immediately, with no system modification required because these operating systems know about file archiving. However, CP/M-80 users must patch their system BDOS (Basic Disk Operating System) to "teach" file archiving to CP/M-80. The code in Listing 1 must be edited and assembled, then loaded (as a '.HEX' file) with DDT (Dynamic Debugging Tool) using your normal system installation procedure for merging your 'Boot Loader' and BIOS (Basic Input/Output System) into your system "image" for eventual SYSGEN'ing. (This code assumes the standard SYSGEN position of address 980H for the start of the CCP [Console Command Processor] with a CP/M-80 system 'size' of 56 Kilobytes; .change the value at label MSIZE to suit your needs.)

One word of warning however, is that this 'archive patch' can only be incorporated into an unmodified CP/M-80 (version 2.2) BDOS *only*; any other prior modifications will undoubtedly be overlaid by this one, with unpredictable results. As stated before, the CP/M-86 version of ARCHIVE does not require 'patching' of the operating system . . . 'nuf said!

ARCHIVE Source Code

The conclusion of this article is the source code for both versions of ARCHIVE; the '.ASM' file is for CP/M-80 (and MP/M-80 II, if you are using conditional assembly, switch label 'MPM' is set TRUE). The '.A86' file (translated from the 8080 version) is included for for CP/M-86 and MP/M-86 users.

Listing 1

```
;              CP/M-80 Version 2.2 File ARCHIVE Patch
;
msize           equ     56              ; size of this CP/M 2.2 system
;
bdos$loc        equ     (msize-20)*1024+3c00h   ; base address of BDOS
wrsec           equ     bdos$loc+03b8h  ; address of write sector routine
pntdir          equ     bdos$loc+055eh  ; address of directory pointer routine
reset$archive   equ     bdos$loc+0df0h  ; address of reset archive bit patch
;
ccp$base        equ     0980h           ; sysgen ccp base position
bdos$entry      equ     ccp$base+0806h  ; sysgen bdos entry position
force$r$w       equ     bdos$entry+05bah; force R/W disk always...
wrt$dir         equ     bdos$entry+05c8h; 'wrsec' call location, in DIR write routine
scratch         equ     bdos$entry+0deah; scratch RAM inside bdos for patch

        org     force$r$w
;
        ret                             ; return from 'check changed disk'
;
        org     wrt$dir                 ; patch 'call wrsec'
;
        call    reset$archive           ; call patch in scratch RAM
;
        org     scratch                 ; patch area for 'archive bit reset'
;
        call    pntdir                  ; point to directory entry in buffer
        lxi     d,11                    ; make offset to 't3' in FCB
        dad     d
        mov     a,m                     ; get 't3' character from FCB
        ani     07fh                    ; kill archive bit position
        mov     m,a                     ; return reset archive bit to FCB
        call    wrsec                   ; write directory sector
        ret                             ; return to 'wrt$dir' routine
;
;
;
```

Listing 2

```
        db      cr,lf,'3055 Waco Avenue, Simi Valley, CA 93063'
        db      cr,lf,'Z'-40h   ; force end-of-file for display
;
; make a new stack pointer at CP/M serial number
;
over:   lhld    bdos+1          ; set up a stack
        sphl                    ; at top of tpa
;
; get curent logged disk, and save it
;
        mvi     c,currdsk       ; return curent disk
        call    bdos
        sta     logdsk          ; save as logged disk
;
; reset disk system in case someone swapped in new disk on the system
;
        mvi     c,rsetdsk       ; reset disk system function
        call    bdos
;
; now return to original login disk
;
        lda     logdsk          ; get logged in disk number
        mov     e,a             ; do disk select
        mvi     c,seldsk
        call    bdos
;
; announcing, the new and improved ARCHIVE...it beats as it sweeps, as it cleans
;
        lxi     d,signon        ; point to signon message
        mvi     c,pmessg        ; print string fuction
        call    bdos            ; print it
;
; check for proper environment, we only live for CP/M 2.2
;
        mvi     c,version       ; check version number, must be 2.2
        call    bdos
        cpi     022h            ; version number correct?
        lxi     d,bad$ver$num   ; set-up bad version message, in case its not
        jnz     arcexit         ; bail-out now, if wrong version number
;
        if      not mpm         ; CP/M-80 does not know about archiving...
        lhld    bdos+1          ; see if archiving installed in CP/M
        lxi     d,05c9h         ; add offset to archive patch address
        dad     d
        shld    arch$addr       ; save address for later
        mov     a,m             ; get low byte content
        cpi     0b8h            ; is this address for write sector routine?
        jnz     patch$bdos      ; if not, patch it back into bdos
        lxi     d,noarch        ; tell user that archive is not installed
        jmp     arcexit         ; exit via CP/M warm boot
;
patch$bdos:
;
        lhld    bdos+1          ; patch write sector address back to bdos
        lxi     d,03b2h         ; add offset address of 'wrsec'
; *+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*
; *+                                                                       +*
; *+  ARCHIVE.ASM Utility Version 1.0,  as of 6-June-82, by Kelly Smith   +*
; *+                                                                       +*
; *+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*
;
;
;
true    equ     -1              ; define true
false   equ     not true        ; define false
;
mpm     equ     not true        ; conditional assembly for MP/M
```

(continued on page 49)

Volume 83, Abstracts and Catalogue

CP/M Users Group

Abstracts

1. Gerald A. Edgar's muMATH/muSIMP routines
2. E.R. Le Clear's Complex math routines for CBASIC 2
3. William R. Brandoni's CP/M-80 subroutines for FORTRAN-80

1. On this disk are routines for users of Microsoft's muMATH/muSIMP symbolic mathematics system. Tested at muSIMP-80 level 2.12, by Gerald A. Edgar, 107 W. Dodridge St., Columbus, OH 43202, 614-262-3232.

Several of these files extend the capabil-

Catalogue

DESCRIPTION: Extensions for commercial languages:

1. MuMATH/MuSIMP enhancements; (Gerald A. Edgar)
2. CBASIC complex math routines; (E.R. Le Clear)
3. CP/M-80 functions for FORTRAN-80; (William R. Brandoni)

NO.	SIZE	NAME	COMMENTS
	2K	-CATALOG.083	CONTENTS OF CPMUG VOL. 83
	4K	ABSTRACT.083	Abstract of volume
	—		
83.1	6K	ARITH%.MUS	muMATH arithmetic
83.2	2K	ATRG.TRG	muMATH inverse trig functions

83.3	1K	INT%.DIF	muMATH integrals
83.4	1K	LIM%.DIF	muMATH limits
83.5	1K	LOG%.ALG	muMATH logarithms
83.6	12K	MUMATH%.DOC	explanations of the files
83.7	10K	SERIES%.DIF	muMATH infinite series
83.8	5K	SOLVE4%.EQN	muMATH solution of equations
83.9	3K	TRGNEG%.ALG	muMATH trigonometric functions
	—		
83.10	11K	APPENDIX.DOC	Appendix to CMATH/DOC. matrix math and circuit relationships.
83.11	13K	CMATH.DOC	Documentation for other complex math file except COMCALC/BAS
83.12	12K	COMCALC.BAS	Source code for "calculator" mode complex math operation.
83.13	5K	COMCALC.INT	Compiled COMCALC/BAS
83.14	6K	COMDISK.DOC	DOC for complex subroutines XXXXXX.SRT
83.15	1K	DIVCOM.SRT	Divide two complex quantities
83.16	8K	LPFILTAN.BAS	Source code for multipole filter analysis program
83.17	4K	LPFILTAN.INT	Compiled LPFILTAN.BAS
83.18	1K	MPYCOM.SRT	Multiply two complex quantities
83.19	1K	MTXMPY.SRT	Matrix multiplication
83.20	1K	PAR/SER.SRT	Parallel to serial impedance convert
83.21	1K	POL/REC.SRT	Complex polar to rectangular convert
83.22	1K	REC/POL.SRT	Complex rectangular to polar convert
83.23	1K	REFC/Z.SRT	Reflection coefficient to Z convert
83.24	1K	SER/PAR.SRT	Serial to parallel impedance convert
83.25	2K	SMATX.SRT	Convert S matrix to X matrix
83.26	2K	XMATS.SRT	Convert X matrix to S matrix
83.27	2K	YZZY.SRT	Convert Y to Z or Z to Y matrix
83.28	1K	Z/REFC.SRT	Z to reflection coefficient convert
83.29	1K	ZXXZ.SRT	Convert Z to X or X to Z matrix
	—		
83.30	3K	CONTENT.DSK	Brief description of disk contents.
83.31	12K	CPMFN.MAC	Microsoft M80 source for part of CPMLIB.
83.32	1K	CPMFN.REL	Assembled version of CPMFN.MAC
83.33	3K	CPMFNA.MAC	Microsoft M80 source for part of CPMLIB.
83.34	1K	CPMFNA.REL	Assembled version of CPMFNA.MAC
83.35	30K	CPMINT.FOR	Microsoft F80 source for part of CPMLIB.
83.36	5K	CPMINT.REL	Compiled version of CPMINT.FOR
83.37	17K	CPMLIB.DOC	Documentation on the use of CPMLIB.
83.38	4K	CPMLIB.MOD	Documentation on methods to modify or add to the functions supported by CPMLIB.
83.39	6K	CPMLIB.REL	Library which can be used to interface Microsoft FORTRAN-80 to CP/M BDOS and other system-level functions.
83.40	5K	TEST.FOR	FORTRAN source for program to test library.
	—		
	5K	U-G-FORM.LIB	CPMUG contribution form
	2K	CRCK.COM	CRC check program.
	2K	FILES.CRC	CRC of files on disk

CPMUG does not have phone service, but welcomes your written requests for ordering information. Send your inquiries to: The CP/M Users Group, 1651 Third Ave., New York, N.Y. 10028.

ities of the files provided with the muMATH/muSIMP distribution disk. For example: evaluation of inverse trigonometric functions, formulas for the solution of fourth degree equations, summation of infinite series. Generally speaking, the added capabilities must be paid for in space and time.

The file MUMATH%.DOC explains the capabilities of each of the files on the disk. Instructions for using each of the files (usually by merging it with an existing muMATH file) are given as a comment at the beginning of the file.

A text editor (not included) must be used to merge these files with existing muMATH files. Instructions have been placed at the beginning of each file.

2. These are complex math routines for CBASIC 2, including information on using them for circuit design, etc., by E.R. Le Clear, 209 Alverno Dr., Ft. Wayne, Ind. 46816, 219-745-7652.

Documentation on the disk describes a method for generating CBASIC "public library" files to be incorporated as subroutines in other programs. Several complex math subroutines are included as well as two application programs to illustrate the use of the subroutines and the method, such as a multipole filter analysis program.

Files APPENDIX.DOC, CMATH.-DOC, and COMDISK.DOC supply more information.

3. William R. Brandoni's CP/M-80 subroutines for FORTRAN-80 are included in this volume.

CPMLIB.REL is a library of subroutines which permits the Microsoft FORTRAN-80 user to take advantage of some CP/M-80 and BDOS functions not supported in the FORTRAN language. For example, file manipulation such as renaming or erasing files. Direct console input is also supported. The library is set up so that it can be searched during a link using LINK-80.

More complete documentation is available in the CPMLIB.DOC and CPMLIB.MOD files.

-Abstract compiled 06/28/82
by Ward Christensen

Opinion
Zoso

NCC '82

This story actually began a few months ago when I first learned of the scheduling conflicts between NCC and CES (Consumer Electronics Show – summer edition). Both were slated for the second week of June, in Houston and Chicago respectively. The choice I faced was basically one of consumer quality toys or industrial quality tools; although the distinction is somewhat blurred by the fact that many CES toys are designed to industrial tool quality and vice-versa.

As of June 1, I was still wavering. With each passing moment, decent accommodations for both shows were becoming all too scarce as were seats on the direct flights. Little did I suspect, but a decision was almost upon me. In fact, the very next day, an OEM requested my presence at NCC to help assess what the competition was up to. This request was sweetened by promises of all the usual perks plus a couple of bonus ones. Pack my spurs and six-shooter, Mama. I'm going to Texas.

June 6 – THE DAY BEFORE

It had been at least ten years since my last visit to Houston so I decided to come a day early for some reorientation . . . I arrived in the early afternoon. After getting my luggage, I parked myself on one of the nice rock hard benches outside the airport arrival area to await the car rental 'courtesy bus'. Within the first few minutes, my suit wilted, and me along with it. Later, while still waiting, I nodded off and dreamed of escape from some tropical penal colony. (Could it have been the climate?)

Eventually the courtesy bus showed up. I climbed aboard and was clobbered by the smell of stale cigar butts. How very courteous! I prayed that the ride would be brief. It wasn't. Eventually the driver dropped me at the rental location. (Actually, I alighted; it was my luggage which he dropped.) After some suspiciously perfunctory paperwork, I was handed a set of keys and shown to my vehicle. I could have died on the spot! I mean the odometer on this thing must have rolled over at least once while it was still in service as a Central American police car. Later I was to learn that securing this rolling death trap on such short notice represented a stroke of exceptional good fortune – I'll explain anon.

I turned the ignition key and within minutes all seven cylinders roared to life. I drove South, in the general direction of my hotel. Several miles before the downtown area, I noticed signs indicating exits for the 'Outer Loop' and the 'Inner Loop'. Hmmm, concentric beltways, nested loops, computer show – how very prophetic . . .

By now, a few hours had passed since my plane had touched down. Since I no longer eat anything on domestic flights (except for Macadamia nuts in sealed snak-paks), I was becoming ravenous. I stopped at a franchise type quick food restaurant and ordered the house specialty, a roast beef sandwich. I had to spit out my first hungry mouthful. The meat part was warship gray (something resulting from even more overcooking than the dark brown kind which I also can't swallow). All of a sudden, it came back to me; Texans like their meat well done. Ugh!!! On my way out, I noticed a sign on the wall, which bore Roy's signature. (Roy was a legendary celluloid cowboy and he's the nominal 'host' at these places.) Anyway, according to Roy, 'Cowpokes and cowgals show their good Western manners by cleaning off their tables when they've finished chowing down' (or some such hokey wording). Roy, you must be kidding! You have enough money to buy Rhode Island. Hire some real busboys (and some real chefs too)! No matter; it's all academic. I'll never have to worry about being a polite cowpoke because I'll never finish off any vittles which have been subjected to such excess incineration.

My hunger pangs were getting critical. I could ill afford a second helping of 'pitching grub' (a term of my coinage, meaning spoiled or badly prepared food which should be discarded uneaten). I tried to play it safe with candy bars from the machine at a nearby gas station. Even this expedient proved less than ideal. These munchies of last resort had succumbed to the ambient temperature. I ate them anyway. Flaccid or not, they provided enough carbos to sustain me for the remainder of my drive.

One wrong turn and many miles later, I arrived at my hotel. After a refreshing shower and an even more refreshing six pack, I got down to work. In this case, my work involved figuring out which hotspots to visit that evening. I had initially presumed that I would begin my rounds at Gilley's. This famous Houston watering hole provides some three acres of enclosed floor space. On an average night there are several thousand foot-stomping patrons. Still not impressed? Gilley's is the birthplace of the mechanical bull. When I consulted my map, I realized that Gilley's was on the other side of town – too distant for my opening foray. This minor dilemma was soon resolved with the help of 'Houston at Night', a supplemental section in Computerworld's 'NCC Preview' issue. My heartfelt thanks go out to Henry de La Garza who compiled this godsend. I made a tentative decision to start at a place called Cooter's because it was near my hotel and I liked the name.

Earlier in the day, as I was registering, I asked the young lady at the front desk what I should do after dark. The real reason I did this was because I'd hoped that she would join me. This was not in the cards but she sounded sincere when she suggested that Houston's hottest nightspot was the hotel's very own cocktail lounge. I was suspicious but, as I headed out to the parking lot, I decided to poke my head in the door, just in case. The place was small and dark except for a few pseudo disco lights

which flashed in time with the jukebox. Three American Gothic couples were drinking beer in catatonic silence. Two merchant seamen were making lurid remarks to the cocktail waitress. Perhaps they wanted more from her than cocktails, but I couldn't imagine why. Perhaps it was her platinum beehive – surely one of the tallest I'd ever seen. The only other person there was the bartender – probably her husband. He was muttering muted oaths at the race results in his newspaper. This lounge was surely not the place. Adios!

I decided to go with my first hunch and I headed to Cooter's. It was a large club, jampacked with merrymakers, an incredible percentage of whom were absolute knockouts. In no time at all, I had hit it off with one of these striking lovelies. After minimal small talk, we split so that she could give me the grand tour of her hometown. This consisted of a brief but interesting drive, a few more ginmills, a couple of video arcades and what I will discreetly call 'just desserts'. All of this was great, but as the sun began to rise over the Lone Star state, I remembered that I owed an appearance at the big doings (NCC). So, several pleasant memories richer, I settled in for a few hours of desperately needed sleep.

This story will soon bog down with NCC highlights. Before that happens, I'd like to report on some of the video games which I saw in the arcades.

MAKE TRAX – One of the silliest concepts of all time stands ready to swallow your quarters. Your persona is a paintbrush which must coat the floor of a maze. This maze is patrolled by hostile goldfish (Yep, you heard me right). The best method of teaching these goldfish some respect is to flatten them with a paint roller. I'll spare you further details and you'll be glad that I did.

ZAXXON – Truly splendid graphics completely fail to exonerate the rest of the idea. You must pilot an attack plane over a heavily defended enemy redoubt. Unfortunately, the graphics impose a rather unusual placement of your seat in the cockpit – above and behind the right elevator by roughly half a mile. It's entirely probable that even with hours of practice, America's top combat pilots won't do nearly as well as twelve year old video junkies. Think of Zaxxon as a grossly misaligned flight simulator and save your money.

DIG DUG – This is a masterpiece! Our video hero wields a bodily protruberance with a nasty looking barbed tip. When you press the right button, this thing extends itself and strikes like an angry King Cobra. If all goes well, the barb lodges inside the abdominal cavity of an enemy monster. Once this happens, the hero pumps some sort of fluid or gas through his fighting unit until the monster explodes. DIG DUG is lots of fun and bound to offend many. Hooray! Note: Modesty forbids me to tell you exactly where on the hero's body his 'weapon' can be found. Find one of these machines and see for yourself. If for some reason you can't, wait for the movie, 'Digging Deep', starring John C. Holmes.

PHOENIX – Shoot Phoenix birds and dodge their 'eggs'. The eggs are depicted as a continuous barrage of small white spots, expelled from somewhere beneath every enemy bird. Unless I learn on good authority that the Phoenix empire has fielded an all female 'air force', I will continue to wonder about these so called eggs. I grew up in New York City where common pigeons provide similar action at no cost. There's no way I'm about to invest two bits merely to dodge bird spatter. On the other hand, the graphics are pretty neat and the birds make just the kind of strident noises which which one might wish to silence with well aimed gunfire. Actually the sound effects bring to mind the frequent whining fits thrown by the dragon lady (my ex). Oh yes, PHOENIX... The birds keep squawking until every last one has been blown from the sky. I wonder if my enthusiasm for this game owes some kind of subliminal debt to the countess of conflict and her many demanding moods.

June 7 – DAY 1

The 10 A.M. wake up call came all too soon. I arose with some major misgivings which some would dismiss as a hangover. Still, duty called – I headed to the show.

For those of you who may not have heard, NCC '82 was held in the Astrodomain. This catchall name is given to the complex which houses the Astrodome where the Astros play baseball on Astroturf. Right next to that is the Astrohall where most of the computer exhibits were located. Past the Astrohall lies Astroworld, an amusement park. Do you detect a motif in any of this?

I arrived at about 11 A.M. and parked in the Astrolot (after some Astrotroll had relieved me of two bucks for the privilege – a rather unecessary gouging in my opinion). The lot was fairly full and I had to park about half a mile from the exhibit area. The thermometer was climbing through a hundred and the humidity was even worse than the day before. I would describe the walk as Astrotorture.

During the month before showtime, I'd taken note of the many press releases describing some new system which was supposed to keep waits in the registration lines down to twenty minutes or less. Balderdash!

I entered the main hall and was immediately underwhelmed by my first glimpses. NCC just didn't look as large as the one they held in Chicago last year. I decided to warm up with some good old fashioned freebie hunting. I am sure many of you are wondering about this, so I'll tell you: I've never, ever seen such paltry handouts. With the exception of the Micropolis frisbees and one or two lesser items, the freebies at NCC were a ripoff at the price. I blame this major disappointment on the economy; work on it, Ron!

Shortly after noon, I gave up on the freebies. I entrusted my barely bulging booty bag to the kind folks who had sponsored my visit and I set out to investigate the exhibits. Actually, once I

(continued next page)

got down to it, I was pleasantly surprised by some of the new goodies. Highlights follow.

The success of the Osborne I personal/portable has apparently fostered some serious competition. It seems that the newcomers have decided to charge more and do it better. Bravo! My favorite amongst these was an STD based unit from Micro Source, the M6000P, whose 9 inch CRT does justice to a full 24 by 80 display. The disk drives were five inchers but at least they worked well. Even better, one can hang aftermarket eight inchers on the controller (at some expense in portability). The video card can emulate several popular terminals.

Micro Source's NCC demo system was running Archivist, a splendid program distributed by Ecosoft of Indianapolis. In fact, this program is so sharp that, rather than describing what it does, I'll use the allotted space to urge each and every one of you to request complete details directly from Ecosoft. You won't regret it!

Speaking of STD systems, there were more of these than I had ever seen before. In case you didn't know, STD refers to a bus structure, rather tightly defined for eight bit applications. Unlike the S-100, STD equipment cannot easily be goosed into serving larger machine architectures (although Micro Source is reportedly planning a 68000 card - ask them how they're going to do it - I have no idea).

I remain especially partial to the STD based computers which were shown by DY-4 Systems of Ottawa. The last time I mentioned DY-4 equipment, I was mainly impressed by the solid hardware and the working RAM-disk option. Since then, the software which runs it all has been exquisitely fine tuned. Rugged and reliable design or racecar performance; take your pick. In this case, pick both.

Half height and double sided drives are another rage. Virtually all drive manufacturers are in on this new trend. Many of the new five inchers claimed incredible storage capacities. Just as soon as the next generation of media technology becomes available, we'll find out whether these claims are legitimate.

There were lots of new dot matrix printers. Some of these purportedly produce letter quality documents. If this is true, why would anyone pay extra for the slower and more expensive thimbles or daisy wheels? Other manufacturers of dot matrix printers more modestly tout raw speed. Of these, I was especially impressed by Okidata's most recent offering. I was equally awed by the Siemens ink jet printer which is slightly slower but ever so much quieter. My guess is that the Siemens ink jet will prove highly reliable. I'd be happy to evaluate one. (Are you listening, Siemens)?

I'm not the kind of guy who can afford to take time from my busy schedule just to speak kindly of IBM, but they are developing some neat stuff. They demonstrated a plasma display, roughly the same size as an LP record, with more than 700K individually addressable pixels. Another IBM gem was a two gigabyte disk with a maximum access time of 16 milliseconds. Don't start drooling, these were both prototypes of what you won't be able to afford later on.

Lifeboat formally introduced the Backgammon/Gomoku package which I hinted about several months ago.

Micropro showed a full color version of WordStar for the IBM P.C. on an unmanned terminal. Note to Micropro: I'm quite satisfied that the search and replace macros work perfectly. In fact, I'm the one who tested them by replacing much of your sales pitch with the naughty words.

It was now almost 5:00 P.M. and I had done more than enough for one day. I got in my car for the fifteen minute ride back to the hotel. Wouldn't you know it, the one time when I was too beat to care about freebies, I was obliged to accept another - an extra hour mired in Astrotraffic.

Later, about a dozen cars departed from my hotel, headed for Gilley's. I was at the helm of one of them. Gilley's was everything I'd expected and perhaps more. My only gripe was that I couldn't find refuge from the music. Don't get me wrong; I love country music - just as long as the country in question is not in North America . . .

She cried when her cattle got anthrax,
She cried when I shot her in the heart.
If this woman I love don't stop crying,
I'm afraid she and I must soon part.

I may not be a fan of such mindless doggerel, but I'm well aware that some people earn megabucks just by churning it out. I'm seriously considering a leave of absence for this very reason.

June 8 - DAY 2

For the second day running, things got off to a late and rocky start but somehow I made it back to the show. My goal was to cover whatever I had missed yesterday, wheel and deal a bit and to revisit a few especially interesting exhibits.

Sony's microcomputer! It's small, it's pretty! An instant contender if ever I saw one. Sony's CP/M operating system and BASIC reside in a separate memory bank. Both support extensive color graphics yet leave the user with lots of TPA. Sony's 3 ½" drives are impressive, considering. Note: Shortly after the show, I read that Shugart and some other manufacturers were convening a session to propose standards for 3 ½" microfloppies. Good thinking, Guys!

The huge crowds gathered at the Apple and Commodore booths didn't look like my type so I didn't bother elbowing my way to the front. I'm sure that neither Apple nor Commodore missed me any more than I missed them.

When the exhibits had closed for the day, Mini-Micro Systems (a magazine) hosted a neat party at the amusement park. Rides, food and beer were all free.

June 9 - DAY 3

By now, I'd seen and talked enough computers to hold me for another year so I headed down to the Galveston area for some heavy duty beachwork.

POSTSCRIPTS:

What was the show really like? For one thing, it was too big for Houston. As it turned out, I had apparently snagged one of the very last rental cars in the whole city and, despite my initial shock, I was spared the agonies which awaited those unfortunates who had to ride the NCC shuttle buses. By all credible accounts, there were far too few of these cramped swelterboxes. I kept hearing that even one short hop in them was enough to ruin the entire NCC experience. There must be lots of taxis in Houston, (or so the Yellow Pages would have you believe). The people I met who had counted on using or finding taxis all ended up logging lots of shuttle bus time.

When I had first checked into my hotel I overheard the desk clerk informing a young honeymoon couple who had driven through the night that because of 'the big computer convention' there were no vacancies in the entire city. Throughout NCC week, pitiable stories such as this could be heard at virtually every hotel desk in town. A few NCC visitors with 'firm' reservations got burned because of overbooking in their hotels. Others were nailed when guests (from the previous week) decided to extend their stays. Either way, I was entertained by lots of indignant and rather well founded grumbling.

Still, all things considered, I liked Houston. I may well take another look when all the conventions are out of town.

Last month I removed Priority One from my good guys list. Now they're back on! My friend finally received his Mitsubishi drives and they work like a charm. This is mentioned merely as a follow up. The real reason that Priority One is back in my good graces is their recent (and exclusive) offer of the Compupro Disk 1 controller plus fully configured CP/M 2.2 for a mere $450.00 - that's the kind of bargain which can easily mend fences.

Zoso

Feature

Z80 Programming Tutorial, Techno-Speak For Techno-Freaks

Kim West DeWindt

Words

Welcome to the wonderful word of techno-speak. I have compiled a list of buzz words, some of which were not covered in the 8080 tutorial, and some of which were covered but need expanded meanings for the Z80 world. I will not bother to redefine common microprocessor jargon. If something is unfamiliar, I would suggest that you refer to Ward Christensen's section on terminology.

The definitions of these words are those which appear in everyday usage. Some of the technical terminology was gleaned from data sheets, and from the Zilog Z80 Databook.

ALU

The Arithmetic Logic Unit (ALU) of a computer can be a complex array of high-speed mathematical adders and multipliers. In the microprocessor realm, the term ALU is frequently used to refer to the accumulator, affectionately known as the A register. The accumulator is one type of an ALU (a rather primitive type).

Actually, the Z80 has two A registers, A and A' (A prime). It does not, however, have two accumulators. A' is a storage register whose contents can be exchanged swiftly, but only with the contents of A. This is useful in CONTEXT SWITCHING. For more information, see the definition of prime registers and context switching in this glossary, and the discussion of prime registers in the section about the Z80's architecture.

Context Switching

When the Z80 (or any microprocessor) abruptly changes where it is operating (and what it is doing), and begins, midstream, another program or subroutine, it has switched context. The operating environment, registers' contents, etc. switch from the logical flow of one program to another's. This usually occurs when the Z80 receives an interrupt. In a multiuser system, (two or more terminals sharing the same processor) the Z80 may be forced to switch from one user's program to that of the next user at regular intervals. It is imperative that the interrupting programs do some bookkeeping, save the contents of the registers, stack pointer and program counter. The next interrupt restores the old values and the Z80 continues on from the point at which it was so rudely interrupted. This is context switching. The Z80 moves back and forth between two or more programs in an orderly manner.

The Z80 has a nifty set of special registers, the prime registers, that simplify context switching. The current operating conditions are in the normal registers. A second set of values waits in the prime registers. Switching context is almost as simple as exchanging the contents of the normal and prime registers. See the definition of primes in this section and in the discussion of the Z80's architecture. Graceful context switching will be covered in the section on the Z80's exchange opcodes.

Demultiplexing

Originally, DEMULTIPLEXING was the art of unraveling one signal into many signals. In the microcomputer world, demultiplexing is used to select one of many sources for use by the microprocessor. This may sound like a contradiction, but is it really just two different uses of the same idea. A demultiplexing chip has three or four inputs, and many more outputs; eight is a common number on a microprocessor board. The inputs are often called select lines because they act like an address, selecting and activating one of the outputs. If a demultiplexer has eight outputs, three select lines provide eight addresses (000 - 111, binary code). If three of the Z80's address lines are connected to the select lines, a given address will select one output line of the demultiplexer. This output can then be connected to the chip select function of a peripheral. Now, the Z80 address selects one, and only one, peripheral. Because of this ability to decode address lines into discrete outputs, demultiplexers are also known as decoders.

A demultiplexer that has eight outputs and three select lines is frequently referred to as a 3 to 8 DEMUX.

Dynamic RAM

Those pesky RAM boards that are so cheap, but never seem to work quite right, are usually designed with dynamic RAMs (Random Access Memory). Dynamic RAMs are so called because they have to be continually refreshed (a special kind of RAM cycle, see REFRESH in this section). If they are not refreshed, they forget the data that was written in them. 16K and most 64K RAMs must have all row addresses refreshed every 2 milliseconds. This all sounds like a bother, so why not use static RAMs and be done with it? For several reasons: dynamic RAMs are cheaper than static RAMs; they have a higher bit density (more bits per square micron of silicon) and come in much smaller packages. The newest dynamic RAMs have 64K bits in one package. Just think, the entire memory space of an eight bit processor in eight 16 pin packages. So much for large, power hungry static RAM boards.

One drawback of dynamic RAMs is that designs using them require careful consideration of memory cycle timing and refresh control. One can't just plunk them in and hope they will run. They won't. Dynamic RAM board de-

sign could be the subject of a whole different article. Suffice it to say that, despite problems, there are reasons to use dynamic RAM.

Knowing all this, Zilog cleverly included some special hardware features to simplify the dynamic RAM interface. These included a special refresh register (actually a 7 bit counter), and a special refresh cycle that occurs after every M1 (instruction fetch) cycle. See the paragraph on the REFRESH REGISTER in this glossary for more information about its purpose. I will discuss refresh cycles in the section covering Z80 architecture.

Index Registers

The Z80 has two sixteen bit index registers that support index addressing. In this mode, the address is calculated by the Z80 at run time, every time the instruction is executed. The actual address is the summation of the contents of one of the index registers and an offset (0 to 255, one byte). The offset is included in the opcode, i.e. it is not a variable.

There are two index registers IX, and IY. Index addressing is handy for accessing look-up tables, jump tables, and interrupt handlers. (Interrupt handlers are subroutines that are executed when the Z80 gets a hardware interrupt.) See the section on Z80 architecture for a more detailed description of the index registers, and some of their limitations (they are *not* full function registers, but are intended primarily for use in these flexible addressing modes).

Interrupt Register

This is a special eight bit register used in interrupt mode 2 (IM2). The Z80 has three interrupt modes; its response to an interrupt depends on the selected mode. In mode 2, the Z80 expects a byte of data from the interrupting device. This byte is concatenated with the contents of the interrupt register. This sixteen bit value is an address that replaces the current contents of the PC register forcing the Z80 to begin operation at the new location. The details of all three operating modes will be covered in the discussion of Z80 opcodes.

LSI

LSI stands for Large Scale Integration. Simply said, LSI refers to how complex an IC is. The more transistors in an IC, the more complex it is. Microprocessors, I/O controllers, and programmable devices with 10,000 transistors or more are LSI devices. Little chips, logic gates, flip-flops and the like (16 pin packages) are manufactured using SSI (Small Scale Integration) techniques. Larger counters, registers, multiplexers, and demultiplexers are MSI (that's right, Medium Scale Integration) devices.

On the horizon is VLSI (Very Large Scale Integration). Some of the new sixteen bit processors, and the newest thirty-two bit processors are examples of VLSI. VLSI is the technology that will enable manufacturers to build microprocessors with the throughput of big computers.

Microcomputer

Microcomputers can refer to computers (a microprocessor-based system that includes CRTs [Cathode Ray Tubes]), disks, and peripherals. It also refers to ICs that incorporate smaller capacity microprocessors, RAM, ROM or EPROM, counters, serial I/O, and lots of parallel I/O on a single chip. The Intel 8048 is a good example of a microcomputer. Zilog's microcomputer is called the Z8. These compact computers are used in systems that require a lot of high speed I/O - keyboards, CRT controllers, disk controllers and other intelligent peripherals. They have a specialized instruction set, optimized to move chunks of data and process multifarious forms of I/O. To top it all off, they run very fast: 10 to 12 Mhz is becoming standard.

Microcomputers handle the bulk of the grunt work, turning random data into neat packets of bytes and leaving the sophisticated software processing to the microprocessor. The rising capabilities and falling costs of today's smart peripherals and controllers are due to the use of these speedy little chips.

MPU

MicroProcessing Unit is another term for CPU (Central Processing Unit). While CPU can refer to any size processor, from IBM types to micros, the term MPU refers strictly to microprocessors. MPU can also refer to the microprocessor board of a system. However, the mnemonic SBC (Single Board Computer) is more commonly used to refer to the whole board.

Multiplex

'... relating to a system of transmitting several messages simultaneously on the same circuit...' (Webster's New Collegiate Dictionary). Multiplexers are handy devices that accept two or more inputs and, depending on the address, connect one of them to an output. Frequently, memory boards and systems are accessed by more than one source (perhaps the CPU and a DMA device). Multiplexers are used to select memory control signals from one, and only one device. A MUX (the familiar form of MUltipleXer) may have two, four, or eight inputs, one output and address select lines. The address lines determine which input is connected to the output.

MUXs are frequently defined by the relation of their inputs and outputs. A 4 to 1 MUX has four inputs and one output. Two address lines are needed to decode the inputs (00 - 11, binary code).

Peripheral

Peripherals are those 'extras' that make your computer handy: the printer, disks, etc. The term is also Zilog's word for all of the chips that are designed to run with the Z80. Zilog loves mnemonics; here are some of the common parts that you will encounter.

CTC - Counter/Timer Circuit - has a bunch of counters and timers (four 8-bit counters to be exact). They can be programmed to perform a variety of timing functions and are used in real time systems, or any system that requires lots of counting or regular interrupts.

SIO - Serial Input/Output - turns the Z80 parallel data into an outgoing serial data stream, then turns right around and assembles incoming serial data into parallel data. Other names for this type of device are UART (Universal Asynchro-

(continued next page)

nous Receiver/Transmitter), EPCI (Enhanced Peripheral Communication Interface and DART (Dual Asynchronous Receiver/Transmitter). Later in this glossary I discuss UARTs in more detail.

PIO - Parallel Input/Output -Operates in a data-byte I/O transfer mode. Sends buffered parallel I/O, with handshaking, to peripherals that require parallel data.

DMA - Direct Memory Access - Can bypass the microprocessor and access dual-ported memory. Used to perform high-speed data transfers from peripherals into memory.

Prime Registers

The Z80 has a special set of registers that echo the normal registers A through HL. These are known as the prime registers and are designated by a quote, A', B', C', D', E', H', and L'. They are not directly accessible to the programmer, but their contents can be rapidly switched with the contents of the normal registers. The Exchange instructions are used to switch the contents of the normal and prime registers. Interrupt handling routines are greatly simplified, so there's no more pushing and popping onto and off of the stack to save the current status. After a couple of quick exchanges you are ready to run.

A detailed discussion of these properly belongs in a discussion of the Z80 architecture, so that is where it will be.

Refresh Register

The refresh register is a unique concept designed by Zilog to ease the interface between microprocessor and dynamic RAM. The refresh register is actually a counter, and its contents provide a refresh address for dynamic RAMs.

At the end of each M1 cycle, the Z80 initiates a refresh cycle. The contents of the refresh register are put on the address lines, and, if sent to a dynamic RAM array, can be used to address a row for refresh. After each cycle, the refresh register is incremented, getting it ready for the next refresh cycle. The counter cycles through all the addresses needed to refresh 16K and most 64K dynamic RAMs.

This automatic refreshing scheme works well in medium speed systems. Some problems arise when interfacing the fastest Z80 (6 Mhz) to slower RAMs. The programmer should not be concerned with these timing considerations. Just pray that the hardware designer did a good job.

Static RAM

Ah, those wonderful, forgiving static RAMs. Just plug them in and they will remember anything, until the power goes off. The latest trend in statics is byte-wide packaging, 2K by 8 bits. These RAMs are pin compatible with 2716 EPROMs; all they are missing is the window. I only suggest that you do not plug one of these into your boot loader socket.

UART

UART has become the generic name for all parallel to serial I/O controllers. Universal Asynchronous Receiver/Transmitters are the heart of the RS-232 interface. The UART takes parallel data from the CPU, adds a start bit, an optional parity bit, and stop bits, and sends it out as serial data stream. Incoming serial data is stripped of all extraneous bits, assembled into a nice, neat byte, and presented to the CPU. Most UARTs have software programmable baud rates, ranging from 45.5 baud to 19.2K baud. Just do not change the baud rate on the fly.

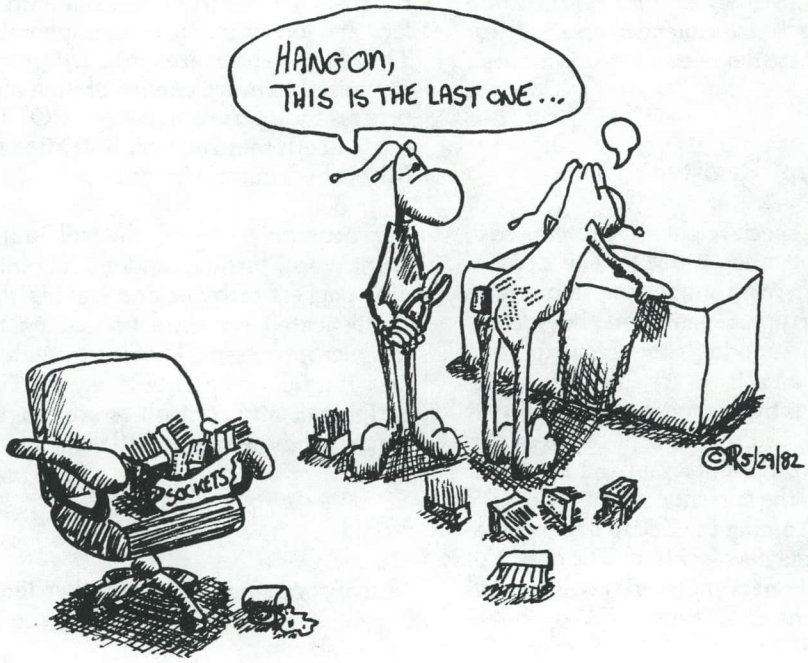
The newest UARTs include on-board baud rate generators. Those UARTs with two complete channels are referred to as DUARTs, for Dual UART.

WORD

A WORD is two bytes, sixteen bits. A sixteen bit address is a word. As sixteen bit microprocessors become more common, you will hear references to word and byte transfers. A byte transfers eight bits at a time, a word transfers sends out (or receives) sixteen bits at a time. Since the Z80 has an eight bit data bus, it can only send and receive one byte at a time. Instructions that manipulate double byte, or word data, must access memory twice, once for each byte. Double register instructions, like DSBC (Double SuBtract with Carry) are word operations.

A DOUBLE WORD, surprisingly enough, is 32 bits of data, usually arranged as four bytes. The Z80 does not have any double word instructions. Most 16 bit microprocessors have double word, or double precision, instructions.

And so we come to the end of our techno-speak orientation. In the next issue, I will discuss the Z80 architecture. Some of its unique features include the refresh register, two true sixteen bit index registers, and those prime registers that I keep mentioning. The three different interrupt modes, and refresh cycles will also be covered.



MicroMoneymaker's Forum

$$$$$$$$$$$Digital Dollars Department

Charles E. Sherman

Announcing Micro-Moneymaker's Imagination Sweepstakes

"The Truth" is often an elusive item, especially when it is sought in such an anachronistic field as microcomputers. Our goal here is to pin it down a bit, to round up some information, and to stimulate dialogue with our readers.

Anachronistic? Computers are at the cutting edge of advanced technology, yet it often seems like we're still a bunch of enthusiastic kids tinkering with cat-whisker crystal radios. A blitz of slick media material and good PR has the public firmly fixed on the image of sophisticated programmers and technicians hunched intently over their terminals, punching codes to work acts of silicon subtlety. Sometimes I believe it myself. But I strongly suspect that we have over anticipated. The future is still coming tomorrow and not today. By analogy to the airplane industry, we are still in back of our bike shops with Wilbur and Orville, tinkering and waiting for our Henry Ford to show up.

I offer the audience of this magazine as proof of my "suspicion." I challenge you to a brief venture into soul-searching cogitation. Here is the proposition: If all of the puff-and-whistle about the microcomputer revolution is here-and-now true, then we will by now have found the micro put to real and practical use in the lives of great numbers of micro enthusiasts. They will have donned their leather helmets, adjusted their goggles, flung their white scarves around their necks, cranked up their Blue Max Bytecrunchers, and pushed off against life's most persistent problem - income.

Look at it logically. Right after death and taxes, the next most certain thing in life for most of us is the generally burdensome need to earn a living. That's premise number one. Our second premise is that the readers of this magazine are an especially educated, talented, and imaginative group of microcomputer literates. Any disagreement? Therefore, if micros are all they are cracked up to be, and if we are as good as we think or hope we are, then we can expect that many of us will by now have figured out some relatively painless or maybe even fun way to make a buck with our microcomputers.

If this isn't the case, then the whole computer revolution thing *must* be a crock created by PR people, and swallowed whole by our own gullible selves. On the other hand, maybe it is all quite true and we just haven't yet brought the facts to light. Who gets the last laugh? To restate the proposition: If micro moneymaking *is* a widespread fact and we don't know it, then we lack important information. If it is *not* a fact, then we lack creative imagination. Either way something is missing, and with your help we hope to get it and share it.

Introducing the Emps

Whatever your use of the micro, however elevated scientifically or professionally, the effort requires support. Traditionally, purist efforts have been supported by patrons, be they public or private, or by struggling, dedicated, compulsive people. But hold on, now. The computer is capable of earning its own way, isn't it? What the heck good is a computer if you can't make a buck with it? Who are you if you don't? By definition, if you spend your time at computering without reward you are a hobbyist, an enthusiast, or maybe a hacker. If you earn money at it, even a little, you are a professional. It's that simple. A regular paycheck identifies the common garden varieties of hired professional, whereas the entrepreneurial professionals are a more diverse and independent bunch, and harder to classify. You must look for subtler signs, such as the addict's pallor, or the wild abandon and reckless enthusiasm of the silicon surfers and skateboarders.

Micros have already given birth to rising classes of Entrepreneurial Microcomputer Professionals, which we shall call **Emps**. Emp also stands for "employed," which is something lots of our fellow citizens are increasingly not. Great changes are obviously afoot in employment patterns, and a lot of buggy-whip makers are going to have to be retrained to make car horns. There are already many new things to do, and new ways to do old things. Right now, anyone who wants to can set up any of dozens of income generating activities by making a relatively small investment in off-the-shelf hardware and software. Your local computer store is a place where dreams and schemes can be assembled, and where you can meet tomorrow's Emps today. Here's a partial checklist of some obvious possibilities:

Common and Obvious Emp Activities

DATA BASE MANAGERS:
- Rental finders
- Real estate listings for owners
- Mailing list management
- Direct sales
- Car pool organizers
- Horoscopes and bio-rhythm charts
- Dating services

WORD PROCESSING:
- Typing services
- Secretarial services
- Bookmaking
- Publishing
- Newsletters

(continued next page)

KNOWLEDGE AND INFORMATION:
- Consulting
- Programming
- Freelance article writing

HARDWARE SERVICES:
- Sales and rentals
- Computer game parlor
- Debugging and repairs

BUSINESS ORIENTED SERVICES:
- Accounting
- Inventory control
- Mail list management
- Consulting

Notice that these off-the-shelf computer occupations are fairly straightforward and obvious. They also look a lot like work, and you don't get very many points for inventing a New Age kind of drudgery. Yet this category is not to be despised, and we who are professional programmers and consultants should be interested in and familiar with these applications for the sake of our clients, and we should try to develop more of them. If you can add to this list, please share with us.

In Search of the Wild Hemp

If you, your friends, or your clients are sailing in the above-described waters, you do indeed qualify as Emps, and such people are in fact a part of the New Wave, although not necessarily on the frothy front of it. That exalted position is reserved for the rarest of all Emps, the hardest to classify, and the most difficult to locate – the Heroic Entrepreneurial Microcomputer Professional (Hemp).

The true, archetypal definition of a Hero is one who departs from the lumpen mass to adventure down untrod paths, exploring new worlds, learning new truths, and eventually returning to play show and tell with the rest of us. These are the pioneers, the innovators, and inventors. In the world of Emps, these are the Hemps. You may be one, you may know one, or you may have heard of one. If so, let us know. Share.

Just for example, let me tell you briefly about the genius who installs terminals in hotel lobbies. A sensor sets off a tape loop which calls quietly to passers-by and invites them to explore the city's possibilities on the terminal. Pleasant dialogue between terminal and tourist narrows the fields of interest (restaurants, shops, theaters, etc.) and after eliciting preferences (Mexican? Hot or not? Skiing? Beginner or advanced?) a list of specific recommendations are displayed. Income is derived from every business listed in the data base, and royalties are paid every time a specific referral is printed out.

There are more examples, and there's more detail, but that's for later columns. First, we'd rather hear from you. Share with us your favorite schemes. The ones you know about, the ones you've heard about, the ideas you've thought about.

It's A Matter of Imagination

More than almost any other technology, computer applications is a matter of imagination. If you can think of it, you can probably make it happen. Innovation requires the imagination to visualize beyond the ruts and routines we all live in, to see new connections and new solutions. What we now propose to do is to sample the imaginative powers of the readers of this magazine as applied to the field of microcomputer moneymaking. What *can* you dream up? What *have* you dreamt up? We want to know what you are doing, what you have heard of being done, and especially what you can imagine being done to dig up digital dollars.

To overcome inertia and other obstacles, we are turning this into a contest. Fill out our snoop sheet (see page 55) and send in a brief description of your best ideas for microcomputer moneymaking applications. You don't have to give away any trade secrets, just tell enough for us to get the general idea. Let us know if the concept described is merely an idea, a rumor you have heard about, something you have seen in action, or something you are doing yourself. The results of this outpouring of input will be shared with all readers in future columns, and we will try to provide any background research or information requested by readers. Join the dialogue.

The interesting information and cross-fertilization that will happen when we share our ideas should be reward enough, but just to make it more fun and tempting, the good folks and *Lifelines/The Software Magazine* are offering $50 to each winner – along with the recognition you'll gain.

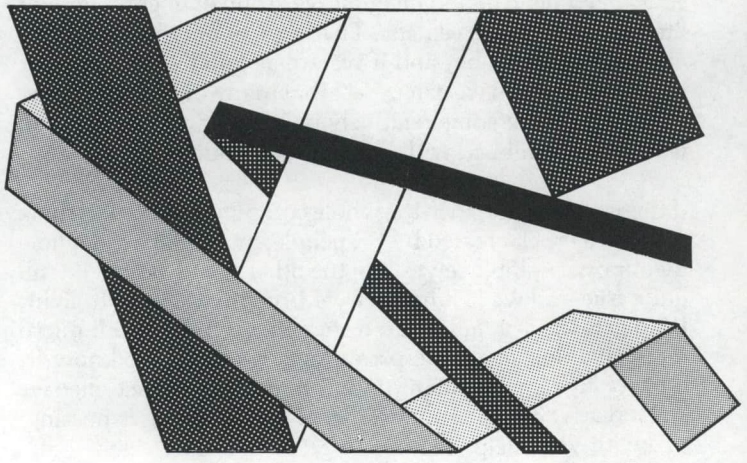
Renew

If your subscription began in *September,*
It would behoove you to *remember*

to send in your renewal right now. We've been waiting to hear from you. You should have received a letter and reader opinion survey that solicits not only your subscription, but your viewpoint on how we can make *Lifelines/The Software Magazine* better. We pride ourselves on our responsiveness to your suggestions.

Please send your check right away. Or get out your VISA or MasterCard and call *Lifelines/The Software Magazine* Subscription Dept. at (212) 722-1700. The address is: 1651 Third Ave., New York, N.Y. 10028.

(P.S. If you've already renewed, but still received a second or third notice, don't be alarmed. Sometimes our letters get crossed in the mail.)



(Ideal Communications Software continued from page 30)
house; each lets you look in a different room), with one holding the comments of the microcommunicator, another holding the replies from the remote computer, and a third displaying a changing array of processing options. If keeping a transcript is important, the microcomputer can keep track of the conversation, paragraphing ideas as users hit carriage returns (word wrap is assumed in such microcommunications software) and weaving the ideas of one user with those of others in something that looks like the transcript of a normal conversation.

If we want to get really fancy, information can be reprocessed and verbalized (via computer voice synthesis) or turned into graphics for games or business. Over the long term, moreover, we may even bypass the keyboard and enter text by talking to the microcomputer (voice recognition) and transmitting the recognized and edited text to the remote computer.

Taking Advantage of Human Factors

I am suggesting that microcommunications packages enhance the way we communicate by taking advantage of the things we know about people. If we know that people can read faster than they can write (and we know that almost anyone can read six times faster than they can write), then we should write microcommunications software that allows both people to write at the same time.

If we know that people tend to write messages that carry over into second, third, fourth and fifth lines on the computer, and that they often forget carriage returns (and we know that), then we should write microcommunications software that takes care of the carriage return for them. If we know that microcommunications use is occasional and that people will forget options and commands, then we should menu-drive microcommunications software and keep the menu (changing as needed) on the screen at all times.

If we know that people often want to access other information while microcommunicating, we should create multifunctional microcommunications software that shows different kinds of information in different windows on the screen. If we know that people will be reprocessing the information they receive while microcommunicating, then we should make that reprocessing a feature of the microcommunications package.

In Microconclusion

It wasn't very long ago that *microcommunications* wasn't even a word. No one had seriously thought about it meaning for the way people communicate. It had a history, of course, represented in terms like terminal communications, data communications, record communications and, more distantly, mail, but it is only in the last year that it has been clearly distinguishable from its closest relation, terminal communications.

Today we are standing on the edge of a future in which the microcomputer, and the microcommunications option, promises to change the ways in which we work and think. That future will be built on the kind of software options that have been discussed in this article, with microcomputer applications merged both with microcommunications software and with other applications. Some microcommunications software packages already have some of the features discussed. Several microcommunications games are already on the market, as is a spreadsheet package for the Apple II that integrates microcommunications and reprocessing software (The Connector, from Context Management Systems). More of this kind of software is on the way, moreover.

I think the day is not very far off when my computer, having been on controlling the house all night, wakes me up in the morning, having checked my electronic mailbox and scanned the newspaper for me while I was sleeping. When I turn on the screen and let the computer know who I am, it will display the day's schedule and advise me on what I should eat for breakfast, then show me the day's mail and the important news of the day. Considering some of the news important, I will place a call to an associate's computer and, discovering he is not yet up, leave a message for him to call. Then, invoking a database, I will save the important information already received for future reference. Meanwhile, my wife will join me at another screen, and checking the mail, notice something that slipped my attention.

As I have said, we are standing on the edge of a future in which the microcomputer, and the microcommunications option, promises to change the ways in which we work and think. I suspect my current use of microcommunications won't seem quite so weird then as it was when I started writing this series.

Product Reports What Are They?

Product Status Reports in *Lifelines/The Software Magazine* come from two sources: you and the software authors. We look to you for bug reports and then, where possible, test out your reports. Software authors are informed of your problems and can respond; sometimes bugs are corrected in new versions before the bug notices even reach you. In addition, some authors provide bug information, often with patches.

Update reports are provided solely by software authors. You'll often notice a boldfaced new version in the *Lifelines/The Software Magazine* Version List. If you don't see a corresponding update analysis in the New Versions section, we haven't received documentation. In many cases, information on an update is one issue behind the announcement in the Version List. Unfortunately, however, many authors just won't tell us why they have issued a new version – which bugs are corrected, what enhancements are provided. So you miss out on information which might help you in making a buying decision.

We think it's pretty obvious by now that even the best software requires real life use before all the kinks can be worked out. So we're asking you to help us serve you, by reporting the bugs you find so that others may help in finding a cure; by urging your software suppliers to keep us informed on updates and bugs. (We're also happy to hear about the new products they are offering.) The resulting forum will help everybody, by creating an atmosphere where software can grow and thrive.

Product Status Reports

New Products

The products described below are available from their authors, computer stores, software publishers and distributors. Information has been derived from material supplied by the authors, and *Lifelines/The Software Magazine* can assume no responsibility for its veracity. Software of interest to our readers will be tested and reviewed in depth at a later date.

Access Manager
Digital Research

This data access management tool uses the keyed file access method; indexed multiple keys and identical key names are handled. Access Manager is linked to the run time system of application programs running under CP/M-80 2.x or MP/M II and written in Digital Research's PL/I-80, Pascal/MT+, CBASIC Compiler, or CB-80.

Display Manager
Digital Research

This 8080/Z80 programmer's tool is designed for the creation of interactive screen displays to be used by application software. Such attributes as flashing, reverse video, underlining and highlighting are supported. A full screen editor permits the programmer to design screens as they will appear to the end user. The run time library included has its subroutines linked with the application program. Display Manager incorporates attributes of the user's terminal type, in an effort to prevent limiting the application product to certain CRT displays. Display Manager runs under CP/M-80 2.x, compatible with PL/I-80, Pascal/MT+, CBASIC Compiler, and CB-80.

High Yield
Hourglass Systems

This mutual fund investment management tool allows users to enter transactions and receive records of amounts invested, withdrawn, paid as dividend, and reinvested. Performance measurements, included rates of return and average share prices, are calculated.

The results of all funds can be summed up for a portfolio; projected prices may be entered for "what if" analyses. High Yield also handles investment trust, IRA plans and Keogh plans. High Yield requires CP/M-80 and 38K.

Market Time
Hourglass systems

This product is intended to aid the investor who wishes to spot stock market trends and turning points. A data base of New York Stock Exchange statistics can be averaged or smoothed for plotting on screen or printer. The user may enter statistics under Weekly Closing DJIA, Advances, Declines, Total Volume, Number Issues Traded, Total Short Interest as a Ratio of Total Volume, Put-Call Ratio of the CBOE and Free Reserves of the Federal Reserve System. Or, menu prompts permit more customization. CP/M-80 and 34k are required.

New Versions

BUG
Version 3.3

This update contains several enhancements: the backspace key (↑H) is now handled; BUG can now debug the .PRG files created by Plink-II; the "H" command, "Go with trap expressions", has been added. A program can be executed at full speed, testing the trap expressions at selected program addresses. In addition, a block move command is included with this new version.

A spurious error #99 which sometimes occurred on the first command given to BUG has been corrected. The bug which caused Displayed or Examined bytes in the ASCII radix to show such characters as CR and LF (and others) incorrectly, has been fixed. Garbage is no longer left in the default buffer at 80H after a program is loaded.

Bugs

Bugs

DocuMate/Plus
Version 1.4

Unbalanced parentheses can be caused by WordStar setting the high order bit on the double quote character. It can be corrected by deleting and re-entering while word-wrap and auto-justification are turned off.

A Note On The Version List

You'll probably notice that *Lifelines/The Software Magazine*'s Version List is missing this month. In response to reader suggestions, we will be publishing the Version List every other month. If this new policy creates an inconvenience for you, please let us know by phone or card; address your comments to Products Editor, *Lifelines/The Software Magazine*, 1651 Third Ave., New York, N.Y. 10028; phone: (212) 722-1700.

You have also requested that prices be included with the software itemized in the Version List. *Lifelines'* research staff is presently checking on the manufacturers' suggested retail prices for all of these products.

(File Archiving and Backup continued from page 35)

```
;
bdos    equ     5               ; CP/M entry point
exit    equ     0               ; CP/M exit point
dfcb    equ     5ch             ; CP/M default fcb
fcbext  equ     dfcb+12         ; fcb extent byte
fcbrno  equ     dfcb+32         ; fcb record number byte
dbuff   equ     80h             ; default disk buffer
;
coninp  equ     1               ; console character input function
pchar   equ     2               ; print character function
dircon  equ     6               ; direct console i/o function
pmessg  equ     9               ; print message function
constat equ     11              ; console status function
version equ     12              ; return version number function
rsetdsk equ     13              ; reset disk system function
seldsk  equ     14              ; select drive function
open    equ     15              ; open file function
close   equ     16              ; close file function
srchfst equ     17              ; search for first file match function
srchnxt equ     18              ; search for next file match function
delete  equ     19              ; delete file function
read    equ     20              ; read record function
write   equ     21              ; write record function
make    equ     22              ; make file function
currdsk equ     25              ; return current logged disk function
stdma   equ     26              ; set dma address function
attrib  equ     30              ; set file attributes function
;
tab     equ     09h             ; tab character
lf      equ     0ah             ; line feed character
cr      equ     0dh             ; carriage return character
;

        org     100h

;
; jump over authorship notice
;
        jmp     over
;
        db      cr,lf,'ARCHIVE Ver.1.0, 6-Jun-82:  Kelly Smith'
        dad     d
        xchg                    ; address patch info to [DE]
        lhld    arch$addr       ; get address to patch in [HL]
        mov     m,e             ; low byte address patched first
        inx     h               ; bump pointer
        mov     m,d             ; high byte address patched second
        endif                   ; endif MP/M
;
; check if filename specified, abort if not
;
        lda     dfcb+1          ; this will be a space, if no filename
        cpi     ' '
        jnz     gotname         ; if not space, probably filename
        lxi     d,no$file$nam   ; inform user of error of their way...
        mvi     c,pmessg
        call    bdos
        lxi     d,opts          ; tell'em about backup options also
        jmp     arcexit         ; bail-out now, no filename
;
; got a name, now check to see which (if any) disk specified
;
gotname:lda     dfcb            ; check for specific drive
        dcr     a
        mov     e,a             ; set up for select disk call
        mvi     c,seldsk
        inr     a               ; if no specified drive, skip call
        sta     src$dsk$num     ; save drive specifier for later check
        cnz     bdos
        xra     a               ; now zap out drive specifier
        sta     dfcb
        mvi     a,'?'           ; force extent number wild
        sta     dfcb+12
        lda     dfcb+17         ; get "B", "S", "R" or "D" option
        sta     option
        cpi     'B'             ; backup operation?
        jz      okopt
        cpi     'S'             ; set archive operation?
        jz      okopt
        cpi     'R'             ; reset archive operation?
        jz      okopt
        cpi     'D'             ; just display?
        jz      okopt
;
badopt: lxi     d,ilgopt        ; let'em know it was an illegal option
arcexit:mvi     c,pmessg        ; bitch, bitch, bitch...
        call    bdos
        jmp     exit
;
; findout now, if operation is for backup or general lobotomy
;
okopt:  lda     option          ; get option specifier
        cpi     'B'
        jnz     arc$s$r         ; if not backup, must be set/reset archive
dest:   lxi     d,req$dest      ; request destination disk for backup files
        mvi     c,pmessg
        call    bdos
        mvi     c,coninp        ; get console input character
        call    bdos
        ani     05fh            ; force upper case
        sbi     'A'             ; force binary digit
        cpi     16              ; disk >P?
        jnc     dest            ; must satisfy ourselves with 16 disks maximum
        lxi     h,logdsk        ; is this jerk backing up to the same disk?
        ani     0fh             ; strip high nibble for match to logged disk
        cmp     m
        push    psw             ; save flags
        inr     a               ; save re-adjusted disk number
        sta     dest$dsk$num
        pop     psw             ; get flags
        jz      dsksame         ; if same, tell'em so now
        inx     h               ; bump pointer to destination disk
        mov     a,m             ; get disk number
        inx     h               ; bump pointer to source disk number
        cmp     m               ; any chance he specified destination as source
        jnz     backup          ; if not, we can (finally) proceed with backup
dsksame:lxi     d,same$dsk      ; sob...(read this anyway you want)
        mvi     c,pmessg
        call    bdos
        jmp     dest            ; computers never lose their patience...
;
; ready for directory write operations now...do archive set/reset
;
arc$s$r:call    crlf            ; tidy up display
        xra     a               ; zero out file count
        sta     filcnt
        lxi     d,dfcb          ; find the first file and get its block map
        mvi     c,srchfst
        call    bdos
        inr     a               ; search successful?
        jnz     gotfile         ; yes, go process rest
        lxi     d,no$file$fnd   ; oops, no file found
        jmp     arcexit         ; exit via CP/M warm boot
;
gotfile:dcr     a               ; compensate for 'inr' above
        rrc                     ; file offset to bits 5 and 6
        rrc
        rrc
        ani     60h
        lxi     h,dbuff         ; point to base of buffer
        mov     c,a
        mvi     b,0
        dad     b               ; index by file offset
        push    h               ; save directory entry, for the moment
        lxi     b,filetable     ; point to base of file table
        call    filepoint       ; get table pointer to [HL]
        xchg                    ; [DE] now points to place in table
        lda     filcnt          ; keep track of number of files in table
        inr     a
        sta     filcnt          ; bump file count
        pop     h               ; [HL] points to directory entry
        mvi     b,32
        call    blkmov          ; copy entry into table
getnext:mvi     c,srchnxt       ; search for another entry
        lxi     d,dfcb
        call    bdos
        inr     a               ; returns 0ffh at end of search
        jnz     gotfile         ; got another one, go save it
;
;  end of directory encountered, now process them
;
tagfile:call    abort           ; check for user abort of process
        lxi     b,filetable-32  ; allow for file count one greater than desired
        call    filepoint
        push    h
        lxi     d,dfcb          ; copy next name to default fcb
        mvi     b,32
        call    blkmov          ; someday I will grow up to be a Z80...
        xra     a
        sta     dfcb            ; clear drive number
        lxi     d,-20           ; point back to extent field
        dad     d
        mvi     m,'$'           ; tag end of print here
        pop     d               ; get back pointer to start of entry
        inx     d               ; bump forward to name
        mvi     c,pmessg
        call    bdos            ; say what we're working on
        mvi     e,' '           ; make space between filename.typ and extent
        mvi     c,pchar
        call    bdos
        lda     dfcb+12         ; get extent number
        push    psw             ; save it
        adi     '0'             ; convert to ascii
        mov     e,a
        mvi     c,pchar         ; display extent number
        pop     psw
        call    bdos
        lda     option          ; get B, S, R, or D
        cpi     'D'             ; display only?
        jz      nextfile
        rrc                     ; bit 7 = 0 for B, R, and D, 1 for S
        ani     80h
        mov     b,a             ; save mask
        lxi     d,dfcb+11       ; point to t3
        ldax    d               ; get it
        ani     7fh             ; strip t3'
        ora     b               ; set bit if option was S
        stax    d               ; put it back
        lxi     d,dfcb          ; point to start of fcb
        xra     a               ; zap out drive field
        stax    d
        mvi     c,attrib        ; do set attributes function
        call    bdos
;
nextfile:
;
        lda     dfcb+11         ; get t3
        rlc                     ; isolate t3'
        ani     1
        adi     'R'             ; make an R or S
        sta     donmsg+1
        lxi     d,donmsg
        mvi     c,pmessg        ; print completion message for this file
        call    bdos
        call    crlf            ; tidy up display
        lxi     h,filcnt        ; point to file counter
        dcr     m               ; count it down
        jz      exit            ; exit if done
        jmp     tagfile         ; tag next file
;
; backup routine - does multiple file search and copy
;
backup: call    crlf            ; tidy up display
noback: call    abort           ; check for user abort of process
        call    mfname          ; set-up multi-file search (seek and ye shall find?)
        jnc     movname         ; file found, if no carry
        lda     mfflgl          ; check if anything ever found...
        ora     a
        lxi     d,bakup$done    ; set-up backup done message
        jnz     nofile          ; if not...indicate, no file found
        lda     got$arc         ; check archive found flag
        ora     a
        jnz     arcexit         ; if not zero, must have found one (or more)
        lxi     d,narcs         ; indicate no archive files found
```

(continued next page)

```
        jmp     arcexit
nofile: lxi     d,no$file$fnd   ; oops, no file found
        jmp     arcexit         ; exit with perturbed message
movname:lxi     h,dfcb+11       ; point to t3 in filetype
        mov     a,m             ; get t3 byte
        rlc                     ; archive bit set?
        jc      noback          ; if so, no backup required
        rrc                     ; adjust back to normal
        sta     got$arc         ; set archive file found flag with non-zero
        ori     080h            ; set 'archived' status back
        mov     m,a             ; put it back
        push    h               ; save pointer
        lxi     d,dfcb          ; point to start of fcb
        mvi     c,attrib        ; do set attributes function
        call    bdos
        pop     h               ; restore pointer
        dcx     h               ; point to t2 in filetype
        mov     a,m             ; get t2 byte
        ani     07fh            ; strip-off t2'
        mov     m,a             ; put it back
        dcx     h               ; point to t1 in filetype
        mov     a,m             ; get t1 byte
        ani     07fh            ; strip-off t1'
        mov     m,a             ; put it back
        lxi     h,dfcb+1        ; point to filename for move and display
        lxi     d,fname         ; display destination for filename
        mvi     b,8             ; set-up to move 8 characters in filename
        call    blkmov          ; too bad that 8080 is defacto standard
        lxi     h,dfcb+9        ; point to filetype for move and display
        lxi     d,fname+9       ; display destination for filetype
        mvi     b,3             ; set-up to move 3 characters in filetype
        call    blkmov          ; LDIR is so nifty...
dspname:lxi     d,fname         ; display filename and filetype
        mvi     c,pmessg
        call    bdos
;
; save first fcb for use later as destination file name
;
        mvi     b,11            ; number of characters to move
        lxi     h,dfcb+1        ; from default fcb
        lxi     d,destfcb+1     ; to destfcb
        call    blkmov          ; round'em up and head'em out
;
; open the source file
;
        lda     logdsk          ; select disk, in case of disk full status
        mov     e,a
        mvi     c,seldsk        ; select disk function
        call    bdos
        lxi     d,dfcb          ; point to source fcb
        mvi     c,open          ; attempt open
        call    bdos
        cpi     0ffh            ; file not found?
        jnz     openok
        lxi     d,src$open$err  ; oops, source file open error
        jmp     arcexit         ; exit via CP/M warm boot
;
; open the destination file
;
openok: lxi     d,destfcb       ; point to destination fcb
        lda     dest$dsk$num    ; get destination disk number
        stax    d               ; set-up destination disk fcb disk number
        mvi     c,delete        ; erase any old file
        call    bdos
        lxi     d,destfcb
        mvi     c,make          ; make the new one
        call    bdos
        cpi     0ffh            ; succesfully created?
        jz      full            ; oops, disk full...time for a new one
;
; read source file to buffer, write to destination as copy
;
copy:   lxi     h,(table$end-filetable)/128     ; save buffer size
        shld    bufmax
        xra     a               ; clear eof flag
        sta     eof$flg
copy1:  call    abort           ; check for user abort on ctrl-c
        lxi     h,0             ; set current buffer counter to zero
        shld    bufcnt
        lxi     h,filetable     ; set buffer start pointer to begin
        shld    bufpnt
;
; file source reading loop to read all of buffer full or stop on eof
;
copy2:  lhld    bufpnt          ; set dma address to buffer pointer
        xchg
        mvi     c,stdma
        call    bdos
        lxi     d,dfcb          ; point at default fcb for reading
        mvi     c,read          ; function set for record read
        call    bdos
        ora     a               ; check if read was o.k., or eof
        jnz     copy3           ; end of file so set eof flag
        lhld    bufpnt          ; set buffer pointer up one sector
        lxi     d,128
        dad     d
        shld    bufpnt
        lhld    bufcnt          ; increase buffer sector count
        inx     h
        shld    bufcnt
        xchg                    ; check to see if memory is full
        lhld    bufmax          ; maximum sector count
        call    cdehl           ; compare
        jnz     copy2           ; if not full go get next sector
        jmp     copy4           ; go handle write operation
;
; here if read operation indicates that the file is at its end on read
;
copy3:  mvi     a,0ffh          ; set eof flag
        sta     eof$flg
;
; write output file processing loop to send memory buffer to destination disk file
;
copy4:  lxi     h,filetable     ; set buffer pointer to start
        shld    bufpnt
copy5:  call    abort           ; check for user abort on ctrl-c
        lhld    bufcnt          ; see if buffer is empty yet
        mov     a,h
        ora     l
        jz      copy6           ; buffer empty so check eof flag
        dcx     h               ; dec buffer sector count for each write
        shld    bufcnt
        lhld    bufpnt          ; set up dma address
        push    h               ; save for size bump
        xchg
        mvi     c,stdma
        call    bdos
        pop     h
        lxi     d,128           ; increase address for sector size
        dad     d
        shld    bufpnt
        lxi     d,destfcb       ; point to output file fcb
        mvi     c,write         ; write record function code
        call    bdos            ; go write
        ora     a               ; check if any error
        jz      copy5           ; o.k., so do next record
        jmp     full            ; oops...disk full, time for a new one
;
copy6:  lda     eof$flg         ; buffer all written so go check eof
        ora     a
        jz      copy1           ; go to read next buffer full
        lxi     d,destfcb       ; point at fcb for file close
        mvi     c,close         ; close file function code
        call    bdos
        cpi     0ffh            ; check if close error
        jnz     backup          ; if not, backup more files
        lxi     d,src$close$err ; oops, close error on destination disk
        jmp     arcexit         ; exit with error message
;
; subroutine to compare [DE] to [HL], [Z] set if equal
;
cdehl:  mov     a,d             ; high bytes equal?
        cmp     h
        rnz
        mov     a,e             ; yes, how'bout low bytes?
        cmp     l
        ret                     ; set zero, if equal
;
; subroutine to allow disk change, to continue backup process
;
full:   lxi     d,destfcb       ; delete partial file on destination disk
        mvi     c,delete
        call    bdos
        lxi     d,dsk$full      ; indicate that disk (or dir) is full
        mvi     c,pmessg
        call    bdos
        lda     dest$dsk$num    ; get destination disk number
        adi     040h            ; ASCII'ize it
        mov     e,a             ; display it
        mvi     c,pchar
        call    bdos
        lxi     d,now$full      ; display remaining portion of full message
        mvi     c,pmessg
        call    bdos
req$cnt:lxi     d,enter$ret     ; tell'em to remove/insert disk, hit return
        mvi     c,pmessg
        call    bdos
        mvi     c,coninp        ; wait for user response
        call    bdos
        cpi     cr              ; carriage return?
        jnz     req$cnt         ; if not, leave reminder on what to do
        call    crlf            ; tidy up screen
        mvi     c,rsetdsk       ; reset disk system for newcomer
        call    bdos
        call    rset$fcb        ; reset fcb for retry of previous file
        jmp     dspname         ; continue on new disk, with last file
;
; Multi-file access subroutine.  Allows processing of multiple
; files (i.e. *.*) from disk.  This routine builds the proper name
; in the fcb each time it is called.  Carry is set if no more names
; can be found.
;
mfname: mvi     c,stdma         ; set dma address
        lxi     d,dbuff
        call    bdos
        xra     a               ; clear fcb extension and  record number
        sta     fcbext
        sta     fcbrno
        lda     mf1g1           ; get multi-file flag
        ora     a
        jz      mfile1          ; if zero, not 1st time flag
        lxi     h,dfcb          ; save filename as requested name
        lxi     d,mfreq
        mvi     b,12
        call    blkmov
        lda     dfcb
        sta     mfcur           ; save disk in current fcb
        lxi     h,mfreq         ; set-up for filename search
        lxi     d,dfcb
        mvi     b,12
        call    blkmov
        mvi     c,srchfst       ; do search first
        lxi     d,dfcb
        call    bdos
        jmp     mfile2          ; check if file found
;
mfile1: lxi     h,mfcur         ; do search first on current filename
        lxi     d,dfcb
        mvi     b,12
        call    blkmov
        mvi     c,srchfst
        lxi     d,dfcb
        call    bdos
        lxi     h,mfreq         ; do search next on requested filename
        lxi     d,dfcb
        mvi     b,12
        call    blkmov
        mvi     c,srchnxt
        lxi     d,dfcb
        call    bdos
mfile2: inr     a               ; return carry set, if file not found
        stc
        rz
;
; move name found to current filename
;
        dcr     a               ; adjust location found
        ani     3
        add     a
        add     a
```

```
        add     a
        add     a
        add     a
        adi     81h
        mov     l,a             ; make filename pointer
        mvi     h,0
        push    h               ; save filename pointer
        lxi     d,mfcur+1
        mvi     b,11
        call    blkmov          ; move filename to current filename
;
; move filename found to fcb
;
        pop     h               ; get filename pointer
        lxi     d,dfcb+1
        mvi     b,11
        call    blkmov          ; move filename to fcb
;
; setup fcb for subsequent file write operation
;
rset$fcb:
;
        lxi     d,dfcb          ; point to source fcb
        lda     src$dsk$num     ; force disk number, in case not logged disk
        stax    d
        xra     a               ; clean-up for new file backup
        sta     fcbext
        sta     fcbrno
        sta     destfcb
        sta     destfcb+12
        sta     destfcb+32
        sta     mfflgl          ; turn off 1st time flag
        ret
;
;
; subroutine to do block moves
;
blkmov: mov     a,m             ; copy byte from [HL] to [DE]
        stax    d
        inx     h               ; bump pointers
        inx     d
        dcr     b               ; loop for count in [B]
        jnz     blkmov
        ret
;
; subroutine to index [BC] by file counter
;
filepoint:
;
        lhld    filcnt          ; get file counter
        mvi     h,0             ; force hi ord to 0
        dad     h               ; multiply by 32
        dad     h
        dad     h
        dad     h
        dad     h
        dad     b               ; use as index to file table
        ret
;
; subroutine to check for user abort (any key pressed)
;
abort:  push    h               ; save all registers
        push    d
        push    b
        push    psw
        mvi     c,dircon        ; check direct console i/o (status)
        mvi     e,0ffh          ; input only
        call    bdos
        ora     a               ; set flags
        jz      abortx          ; return, if zero result
        cpi     'C'-40h         ; control-c for abort?.
        jnz     abortx          ; just return, if not
        lxi     d,abort$process ; indicate we are aborting process
        mvi     c,pmessg
        call    bdos
        jmp     exit            ; exit via CP/M warm boot
abortx: pop     psw             ; restore all registers
        pop     b
        pop     d
        pop     h
        ret
;
; subroutine to do carriage return/line feed
;
crlf:   lxi     d,crlf$msg
        mvi     c,pmessg
        call    bdos
        ret
;
;
signon:         db      cr,lf,'ARCHIVE (CTRL-C to Abort) - Ver.1.0'
                db      cr,lf,cr,lf,'$'
;
ilgopt:         db      'Invalid or Unspecified Option - Must Be Specified As:'
                db      cr,lf,cr,lf
opts:           db      tab,'B - Backup File Archive or,'
                db      cr,lf
                db      tab,'S - Set File Archive or,'
                db      cr,lf
                db      tab,'R - Reset File Archive or,'
                db      cr,lf
                db      tab,'D - Display File Archive'
                db      cr,lf,'$'
;
no$file$fnd:    db      cr,lf,'File Not Found, Aborting$'
;
donmsg:         db      ' ','$'
;
crlf$msg:       db      cr,lf,'$'
;
bad$ver$num:    db      'Must be CP/M Version 2.2 to Archive, Aborting$'
;
        if      not mpm         ; archive must be 'installed' in CP/M-80
noarch:         db      'Archiving Not Installed, Aborting$'
        endif                   ; endif MP/M
;
req$dest:       db      'Destination Disk for BACKUP Files (A to P)? $'
;
same$dsk:       db      cr,lf,'Can''t BACKUP to Source Disk You TWIT!',cr,lf,'$'
;
abort$process:  db      cr,lf,'User Abort of ARCHIVE Process$'
;
narcs:          db      cr,lf,'No ARCHIVE Files Found to BACKUP$'
;
no$file$nam:    db      'No Filename or Option Specified - '
                db      'ARCHIVE must be invoked as:'
                db      cr,lf,cr,lf
                db      tab,'ARCHIVE FN.FT OPTION<cr>'
                db      cr,lf,cr,lf
                db      'Where;',tab,'FN.FT is Filename and Filetype '
                db      '(? and * allowed)'
                db      cr,lf,cr,lf
                db      'And;',tab,'OPTION is specified as:'
                db      cr,lf,cr,lf,'$'
;
src$open$err:   db      cr,lf,'Oops, Can''t Open File on Source Disk$'
;
dsk$full:       db      cr,lf,'Destination BACKUP Disk $'
;
now$full:       db      ': is now FULL - Remove, insert NEW Disk$'
;
enter$ret:      db      cr,lf,'Enter RETURN when ready to continue BACKUP process: $'
;
src$read$err:   db      cr,lf,'Oops, Read Error on Source Disk$'
;
src$close$err:  db      cr,lf,'Oops, Bad Close on Destination Disk$'
;
bakup$done:     db      cr,lf,'BACKUP Complete',cr,lf,'$'
;
fname:          db      '           $' ; 11 spaces for filename.typ
;
got$arc:        db      0       ; archive file found flag (1 = found)
;
mfflgl:         db      1       ; 1st time flag for multi-file access
;
mfreq:          ds      12      ; multi-file requested filename
;
mfcur:          ds      12      ; multi-file current filename
;
arch$addr:      ds      2       ; archive patch address
;
logdsk:         ds      1       ; current logged-in disk number
;
dest$dsk$num:   ds      1       ; destination disk number
;
src$dsk$num:    ds      1       ; source disk number
;
filcnt:         ds      1       ; count of files in filetable
;
option:         ds      1       ; storage for B, S, R or D option letter
;
destfcb:        ds      33      ; destination disk file control block
;
bufmax:         ds      2       ; buffer maximum size
;
bufcnt:         ds      2       ; buffer sector count
;
bufpnt:         ds      2       ; buffer pointer
;
eof$flg:        ds      1       ; end-of-file flag
;
filetable       equ     $               ; file table/buffer starts here
                ds      4*4096          ; 16k buffer
table$end       equ     $               ; file table/buffer ends here
;
;
```

Listing 3

```
; *+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*
; *+                                                                   +*
; *+ ARCHIVE.A86 Utility Version 1.0, as of 6-June-82, by Kelly Smith  +*
; *+                                                                   +*
; *+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*
;
;
;
M       EQU     Byte Ptr 0[BX]          ; xlt86's derived 'M'
;
dfcb    EQU     5ch                     ; CP/M default fcb
fcbext  EQU     dfcb+12                 ; fcb extent byte
fcbrno  EQU     dfcb+32                 ; fcb record number byte
dbuff   EQU     80h                     ; default disk buffer
;
coninp  EQU     1                       ; console character input function
pchar   EQU     2                       ; print character function
dircon  EQU     6                       ; direct console i/o function
pmessg  EQU     9                       ; print message function
constat EQU     11                      ; console status function
version EQU     12                      ; return version number function
rsetdsk EQU     13                      ; reset disk system function
seldsk  EQU     14                      ; select drive function
open    EQU     15                      ; open file function
close   EQU     16                      ; close file function
srchfst EQU     17                      ; search for first file match function
srchnxt EQU     18                      ; search for next file match function
delete  EQU     19                      ; delete file function
read    EQU     20                      ; read record function
write   EQU     21                      ; write record function
make    EQU     22                      ; make file function
currdsk EQU     25                      ; return current logged disk function
stdma   EQU     26                      ; set dma address function
attrib  EQU     30                      ; set file attributes function
;
tab     EQU     09h                     ; tab character
lf      EQU     0ah                     ; line feed character
cr      EQU     0dh                     ; carriage return character
;
        ORG     100h
;
; jump over authorship notice
;
```

(continued next page)

```
        JMPS    over
;
        DB      cr,lf,'ARCHIVE Ver.1.0, 6-Jun-82:  Kelly Smith'
        DB      cr,lf,'3055 Waco Avenue, Simi Valley, CA 93063'
        DB      cr,lf,'Z'-40h           ; force end-of-file for display
;
; get curent logged disk, and save it
;
over:   MOV     CL,currdsk              ; return curent disk
        INT     224
        MOV     Byte Ptr logdsk,AL      ; save as logged disk
;
; reset disk system in case someone swapped in new disk on the system
;
        MOV     CL,rsetdsk              ; reset disk system function
        INT     224
;
; now return to original login disk
;
        MOV     AL,Byte Ptr logdsk      ; get logged in disk number
        MOV     DL,AL                   ; do disk select
        MOV     CL,seldsk
        INT     224
;
; announcing, the new and improved ARCHIVE...it beats as it sweeps, as it cleans
;
        MOV     DX,(Offset signon)      ; point to signon message
        MOV     CL,pmessg               ; print string fuction
        INT     224                     ; print it
;
; check for proper environment, we only live for CP/M 2.2
;
        MOV     CL,version              ; check version number, must be 2.2
        INT     224
        CMP     AL,022h                 ; version number correct?
        MOV     DX,(Offset bad@ver@num) ; set-up bad version message, in case its not
        JZ      L_3
        JMP     arcexit                 ; bail-out now, if wrong version number
L_3:
;
; check if filename specified, abort if not
;
        MOV     AL,Byte Ptr .dfcb+1     ; this will be a space, if no filename
        CMP     AL,' '
        JNZ     gotname                 ; if not space, probably filename
        MOV     DX,(Offset no@file@nam) ; inform user of error of their way...
        MOV     CL,pmessg
        INT     224
        MOV     DX,(Offset opts)        ; tell'em about backup options also
        JMPS    arcexit                 ; bail-out now, no filename
;
; got a name, now check to see which (if any) disk specified
;
gotname:MOV     AL,Byte Ptr .dfcb       ; check for specific drive
        DEC     AL
        MOV     DL,AL                   ; set up for select disk call
        MOV     CL,seldsk
        INC     AL                      ; if no specified drive, skip call
        MOV     Byte Ptr src@dsk@num,AL ; save drive specifier for later check
        JZ      L_4
        INT     224
L_4:
        XOR     AL,AL                   ; now zap out drive specifier
        MOV     Byte Ptr .dfcb,AL
        MOV     AL,'?'                  ; force extent number wild
        MOV     Byte Ptr .dfcb+12,AL
        MOV     AL,Byte Ptr .dfcb+17    ; get "B", "S", "R" or "D" option
        MOV     Byte Ptr option,AL
        CMP     AL,'B'                  ; backup operation?
        JZ      okopt
        CMP     AL,'S'                  ; set archive operation?
        JZ      okopt
        CMP     AL,'R'                  ; reset archive operation?
        JZ      okopt
        CMP     AL,'D'                  ; just display?
        JZ      okopt
;
badopt: MOV     DX,(Offset ilgopt)      ; let'em know it was an illegal option
arcexit:MOV     CL,pmessg               ;
        INT     224
        MOV     CL,0                    ; exit via CP/M warm boot
        MOV     DL,0
        INT     224
;
; findout now, if operation is for backup or general lobotomy
;
okopt:  MOV     AL,Byte Ptr option      ; get option specifier
        CMP     AL,'B'
        JNZ     arc@s@r                 ; if not backup, must be set/reset archive
dest:   MOV     DX,(Offset req@dest)    ; request destination disk for backup files
        MOV     CL,pmessg
        INT     224
        MOV     CL,coninp               ; get console input character
        INT     224
        AND     AL,05fh                 ; force upper case
        SBB     AL,'A'                  ; force binary digit
        CMP     AL,16                   ; disk >P?
        JNB     dest                    ; must satisfy ourselves with 16 disks maximum
        MOV     BX,(Offset logdsk)      ; is this jerk backing up to the same disk?
        AND     AL,0fh                  ; strip high nibble for match to logged disk
        CMP     AL,M
        LAHF                            ; save flags
        XCHG    AL,AH
        PUSH    AX
        XCHG    AL,AH
        INC     AL                      ; save re-adjusted disk number
        MOV     Byte Ptr dest@dsk@num,AL
        POP     AX                      ; get flags
        XCHG    AL,AH
        SAHF
        JZ      dsksame                 ; if same, tell'em so now
        INC     BX                      ; bump pointer to destination disk
        MOV     AL,M                    ; get disk number
        INC     BX                      ; bump pointer to source disk number
        CMP     AL,M                    ; any chance he specified destination as source
        JZ      L_5
        JMP     backup                  ; if not, we can (finally) proceed with backup
L_5:
dsksame:MOV     DX,(Offset same@dsk)    ; sob...(read this anyway you want)
        MOV     CL,pmessg
        INT     224
        JMPS    dest                    ; computers never lose their patience...
;
; ready for directory write operations now...do archive set/reset
;
arc@s@r:CALL    crlf                    ; tidy up display
        XOR     AL,AL                   ; zero out file count
        MOV     Byte Ptr filcnt,AL
        MOV     DX,dfcb                 ; find the first file and get its block map
        MOV     CL,srchfst
        INT     224
        INC     AL                      ; search successful?
        JNZ     gotfile                 ; yes, go process rest
        MOV     DX,(Offset no@file@fnd) ; oops, no file found
        JMP     arcexit                 ; exit via CP/M warm boot
;
gotfile:DEC     AL                      ; compensate for 'inr' above
        ROR     AL,1                    ; file offset to bits 5 and 6
        ROR     AL,1
        ROR     AL,1
        AND     AL,60h
        MOV     BX,dbuff                ; point to base of buffer
        MOV     CL,AL
        MOV     CH,0
        LAHF                            ; index by file offset
        ADD     BX,CX
        RCR     SI,1
        SAHF
        RCL     SI,1
        PUSH    BX                      ; save directory entry, for the moment
        MOV     CX,(Offset filetable)   ; point to base of file table
        CALL    filepoint               ; get table pointer to [HL]
        XCHG    BX,DX                   ; [DE] now points to place in table
        MOV     AL,Byte Ptr filcnt      ; keep track of number of files in table
        INC     AL
        MOV     Byte Ptr filcnt,AL      ; bump file count
        POP     BX                      ; [HL] points to directory entry
        MOV     CH,32
        CALL    blkmov                  ; copy entry into table
getnext:MOV     CL,srchnxt              ; search for another entry
        MOV     DX,dfcb
        INT     224
        INC     AL                      ; returns 0ffh at end of search
        JNZ     gotfile                 ; got another one, go save it
;
;  end of directory encountered, now process them
;
tagfile:CALL    abort                   ; check for user abort of process
        MOV     CX,(Offset filetable)-32; allow for file count one greater than desired
        CALL    filepoint
        PUSH    BX
        MOV     DX,dfcb                 ; copy next name to default fcb
        MOV     CH,32
        CALL    blkmov                  ; someday I will grow up to be a Z80...
        XOR     AL,AL
        MOV     Byte Ptr .dfcb,AL       ; clear drive number
        MOV     DX,-20                  ; point back to extent field
        LAHF
        ADD     BX,DX
        RCR     SI,1
        SAHF
        RCL     SI,1
        MOV     M,'$'                   ; tag end of print here
        POP     DX                      ; get back pointer to start of entry
        LAHF                            ; bump forward to name
        INC     DX
        SAHF
        MOV     CL,pmessg
        INT     224                     ; say what we're working on
        MOV     DL,' '                  ; make space between filename.typ and extent
        MOV     CL,pchar
        INT     224
        MOV     AL,Byte Ptr .dfcb+12    ; get extent number
        LAHF                            ; save it
        XCHG    AL,AH
        PUSH    AX
        XCHG    AL,AH
        ADD     AL,'0'                  ; convert to ascii
        MOV     DL,AL
        MOV     CL,pchar                ; display extent number
        POP     AX
        INT     224
        MOV     AL,Byte Ptr option      ; get B, S, R, or D
        CMP     AL,'D'                  ; display only?
        JZ      nextfile
        ROR     AL,1                    ; bit 7 = 0 for B, R, and D, 1 for S
        AND     AL,80h
        MOV     CH,AL                   ; save mask
        MOV     DX,dfcb+11              ; point to t3
        MOV     SI,DX                   ; get it
        MOV     AL,[SI]
        AND     AL,7fh                  ; strip t3'
        OR      AL,CH                   ; set bit if option was S
        MOV     SI,DX                   ; put it back
        MOV     [SI],AL
        MOV     DX,dfcb                 ; point to start of fcb
        XOR     AL,AL                   ; zap out drive field
        MOV     SI,DX
        MOV     [SI],AL
        MOV     CL,attrib               ; do set attributes function
        INT     224
;
nextfile:
;
        MOV     AL,Byte Ptr .dfcb+11    ; get t3
        ROL     AL,1                    ; isolate t3'
        AND     AL,1
        ADD     AL,'R'                  ; make an R or S
        MOV     Byte Ptr donmsg+1,AL
        MOV     DX,(Offset donmsg)
        MOV     CL,pmessg               ; print completion message for this file
        INT     224
        CALL    crlf                    ; tidy up display
        MOV     BX,(Offset filcnt)      ; point to file counter
        DEC     M                       ; count it down
        JNZ     L_6
        MOV     CL,0                    ; exit if done
        MOV     DL,0
        INT     224
```

```
L_6:
        JMP     tagfile                 ; tag next file
;
; backup routine - does multiple file search and copy
;
backup: CALL    crlf                    ; tidy up display
noback: CALL    abort                   ; check for user abort of process
        CALL    mfname                  ; set-up multi-file search (seek and ye shall find?)
        JNB     movname                 ; file found, if no carry
        MOV     AL,Byte Ptr mfflgl      ; check if anything ever found...
        OR      AL,AL
        MOV     DX,(Offset bakup@done)  ; set-up backup done message
        JNZ     nofile                  ; if not...indicate, no file found
        MOV     AL,Byte Ptr got@arc     ; check archive found flag
        OR      AL,AL
        JZ      L_7
        JMP     arcexit                 ; if not zero, must have found one (or more)
L_7:
        MOV     DX,(Offset narcs)       ; indicate no archive files found
        JMP     arcexit
nofile: MOV     DX,(Offset no@file@fnd) ; oops, no file found
        JMP     arcexit                 ; exit with perturbed message
movname:MOV     BX,dfcb+11              ; point to t3 in filetype
        MOV     AL,M                    ; get t3 byte
        ROL     AL,1                    ; archive bit set?
        JB      noback                  ; if so, no backup required
        ROR     AL,1                    ; adjust back to normal
        MOV     Byte Ptr got@arc,AL     ; set archive file found flag with non-zero
        OR      AL,080h                 ; set 'archived' status back
        MOV     M,AL                    ; put it back
        PUSH    BX                      ; save pointer
        MOV     DX,dfcb                 ; point to start of fcb
        MOV     CL,attrib               ; do set attributes function
        INT     224
        POP     BX                      ; restore pointer
        DEC     BX                      ; point to t2 in filetype
        MOV     AL,M                    ; get t2 byte
        AND     AL,07fh                 ; strip-off t2'
        MOV     M,AL                    ; put it back
        DEC     BX                      ; point to tl in filetype
        MOV     AL,M                    ; get tl byte
        AND     AL,07fh                 ; strip-off tl'
        MOV     M,AL                    ; put it back
        MOV     BX,dfcb+1               ; point to filename for move and display
        MOV     DX,(Offset fname)       ; display destination for filename
        MOV     CH,8                    ; set-up to move 8 characters in filename
        CALL    blkmov                  ; too bad that 8080 is defacto standard
        MOV     BX,dfcb+9               ; point to filetype for move and display
        MOV     DX,(Offset fname)+9     ; display destination for filetype
        MOV     CH,3                    ; set-up to move 3 characters in filetype
        CALL    blkmov                  ; LDIR is so nifty...
dspname:MOV     DX,(Offset fname)       ; display filename and filetype
        MOV     CL,pmessg
        INT     224
;
; save first fcb for use later as destination file name
;
        MOV     CH,11                   ; number of characters to move
        MOV     BX,dfcb+1               ; from default fcb
        MOV     DX,(Offset destfcb)+1   ; to destfcb
        CALL    blkmov                  ; round'em up and head'em out
;
; open the source file
;
        MOV     AL,Byte Ptr logdsk      ; select disk, in case of disk full status
        MOV     DL,AL
        MOV     CL,seldsk               ; select disk function
        INT     224
        MOV     DX,dfcb                 ; point to source fcb
        MOV     CL,open                 ; attempt open
        INT     224
        CMP     AL,0ffh                 ; file not found?
        JNZ     openok
        MOV     DX,(Offset src@open@err); oops, source file open error
        JMP     arcexit                 ; exit via CP/M warm boot
;
; open the destination file
;
openok: MOV     DX,(Offset destfcb)     ; point to destination fcb
        MOV     AL,Byte Ptr dest@dsk@num; get destination disk number
        MOV     SI,DX                   ; set-up destination disk fcb disk number
        MOV     [SI],AL
        MOV     CL,delete               ; erase any old file
        INT     224
        MOV     DX,(Offset destfcb)
        MOV     CL,make                 ; make the new one
        INT     224
        CMP     AL,0ffh                 ; succesfully created?
        JNZ     L_8
        JMP     full                    ; oops, disk full...time for a new one
L_8:
;
; read source file to buffer, write to destination as copy
;
copy:   MOV     BX,((Offset table@end)-(Offset filetable))/128  ; save buffer size
        MOV     Word Ptr bufmax,BX
        XOR     AL,AL                   ; clear eof flag
        MOV     Byte Ptr eof@flg,AL
copy1:  CALL    abort                   ; check for user abort on ctrl-c
        MOV     BX,0                    ; set current buffer counter to zero
        MOV     Word Ptr bufcnt,BX
        MOV     BX,(Offset filetable)   ; set buffer start pointer to begin
        MOV     Word Ptr bufpnt,BX
;
; file source reading loop to read all of buffer full or stop on eof
;
copy2:  MOV     BX,Word Ptr bufpnt      ; set dma address to buffer pointer
        XCHG    BX,DX
        MOV     CL,stdma
        INT     224
        MOV     DX,dfcb                 ; point at default fcb for reading
        MOV     CL,read                 ; function set for record read
        INT     224
        OR      AL,AL                   ; check if read was o.k., or eof
        JNZ     copy3                   ; end of file so set eof flag
        MOV     BX,Word Ptr bufpnt      ; set buffer pointer up one sector
        MOV     DX,128
        LAHF
        ADD     BX,DX
        RCR     SI,1
        SAHF
        RCL     SI,1
        MOV     Word Ptr bufpnt,BX
        MOV     BX,Word Ptr bufcnt      ; increase buffer sector count
        LAHF
        INC     BX
        SAHF
        MOV     Word Ptr bufcnt,BX
        XCHG    BX,DX                   ; check to see if memory is full
        MOV     BX,Word Ptr bufmax      ; maximum sector count
        CALL    cdehl                   ; compare
        JNZ     copy2                   ; if not full go get next sector
        JMPS    copy4                   ; go handle write operation
;
; here if read operation indicates that the file is at its end on read
;
copy3:  MOV     AL,0ffh                 ; set eof flag
        MOV     Byte Ptr eof@flg,AL
;
; write output file processing loop to send memory buffer to destination disk file
;
copy4:  MOV     BX,(Offset filetable)   ; set buffer pointer to start
        MOV     Word Ptr bufpnt,BX
copy5:  CALL    abort                   ; check for user abort on ctrl-c
        MOV     BX,Word Ptr bufcnt      ; see if buffer is empty yet
        MOV     AL,BH
        OR      AL,BL
        JZ      copy6                   ; buffer empty so check eof flag
        DEC     BX                      ; dec buffer sector count for each write
        MOV     Word Ptr bufcnt,BX
        MOV     BX,Word Ptr bufpnt      ; set up dma address
        PUSH    BX                      ; save for size bump
        XCHG    BX,DX
        MOV     CL,stdma
        INT     224
        POP     BX
        MOV     DX,128                  ; increase address for sector size
        ADD     BX,DX
        MOV     Word Ptr bufpnt,BX
        MOV     DX,(Offset destfcb)     ; point to output file fcb
        MOV     CL,write                ; write record function code
        INT     224                     ; go write
        OR      AL,AL                   ; check if any error
        JZ      copy5                   ; o.k., so do next record
        JMPS    full                    ; oops...disk full, time for a new one
;
copy6:  MOV     AL,Byte Ptr eof@flg     ; buffer all written so go check eof
        OR      AL,AL
        JNZ     L_9
        JMP     copy1                   ; go to read next buffer full
L_9:
        MOV     DX,(Offset destfcb)     ; point at fcb for file close
        MOV     CL,close                ; close file function code
        INT     224
        CMP     AL,0ffh                 ; check if close error
        JZ      L_10
        JMP     backup                  ; if not, backup more files
L_10:
        MOV     DX,(Offset src@close@err)       ; oops, close error on destination disk
        JMP     arcexit                 ; exit with error message
;
; subroutine to compare [DE] to [HL], [Z] set if equal
;
cdehl:  MOV     AL,DH                   ; high bytes equal?
        CMP     AL,BH
        JZ      L_11
        RET
L_11:
        MOV     AL,DL                   ; yes, how'bout low bytes?
        CMP     AL,BL
        RET                             ; set zero, if equal
;
; subroutine to allow disk change, to continue backup process
;
full:   MOV     DX,(Offset destfcb)     ; delete partial file on destination disk
        MOV     CL,delete
        INT     224
        MOV     DX,(Offset dsk@full)    ; indicate that disk (or dir) is full
        MOV     CL,pmessg
        INT     224
        MOV     AL,Byte Ptr dest@dsk@num; get destination disk number
        ADD     AL,040h                 ; ASCII'ize it
        MOV     DL,AL                   ; display it
        MOV     CL,pchar
        INT     224
        MOV     DX,(Offset now@full)    ; display remaining portion of full message
        MOV     CL,pmessg
        INT     224
req@cnt:MOV     DX,(Offset enter@ret)   ; tell'em to remove/insert disk, hit return
        MOV     CL,pmessg
        INT     224
        MOV     CL,coninp               ; wait for user response
        INT     224
        CMP     AL,cr                   ; carriage return?
        JNZ     req@cnt                 ; if not, leave reminder on what to do
        CALL    crlf                    ; tidy up screen
        MOV     CL,rsetdsk              ; reset disk system for newcomer
        INT     224
        CALL    rset@fcb                ; reset fcb for retry of previous file
        JMP     dspname                 ; continue on new disk, with last file
;
; Multi-file access subroutine.  Allows processing of multiple
; files (i.e. *.*) from disk.  This routine builds the proper name
; in the fcb each time it is called.  Carry is set if no more names
; can be found.
;
mfname: MOV     CL,stdma                ; set dma address
        MOV     DX,dbuff
        INT     224
        XOR     AL,AL                   ; clear fcb extension and  record number
        MOV     Byte Ptr .fcbext,AL
        MOV     Byte Ptr .fcbrno,AL
        MOV     AL,Byte Ptr mfflgl      ; get multi-file flag
        OR      AL,AL
        JZ      mfile1                  ; if zero, not 1st time flag
        MOV     BX,dfcb                 ; save filename as requested name
        MOV     DX,(Offset mfreq)
        MOV     CH,12
```

(continued next page)

```
        CALL    blkmov
        MOV     AL,Byte Ptr .dfcb
        MOV     Byte Ptr mfcur,AL       ; save disk in current fcb
        MOV     BX,(Offset mfreq)       ; set-up for filename search
        MOV     DX,dfcb
        MOV     CH,12
        CALL    blkmov
        MOV     CL,srchfst              ; do search first
        MOV     DX,dfcb
        INT     224
        JMPS    mfile2                  ; check if file found
;
mfile1: MOV     BX,(Offset mfcur)       ; do search first on current filename
        MOV     DX,dfcb
        MOV     CH,12
        CALL    blkmov
        MOV     CL,srchfst
        MOV     DX,dfcb
        INT     224
        MOV     BX,(Offset mfreq)       ; do search next on requested filename
        MOV     DX,dfcb
        MOV     CH,12
        CALL    blkmov
        MOV     CL,srchnxt
        MOV     DX,dfcb
        INT     224
mfile2: INC     AL                      ; return carry set, if file not found
        STC
        JNZ     L_12
        RET
L_12:
;
; move name found to current filename
;
        DEC     AL                      ; adjust location found
        AND     AL,3
        ADD     AL,AL
        ADD     AL,AL
        ADD     AL,AL
        ADD     AL,AL
        ADD     AL,AL
        ADD     AL,81h
        MOV     BL,AL                   ; make filename pointer
        MOV     BH,0
        PUSH    BX                      ; save filename pointer
        MOV     DX,(Offset mfcur)+1
        MOV     CH,11
        CALL    blkmov                  ; move filename to current filename
;
; move filename found to fcb
;
        POP     BX                      ; get filename pointer
        MOV     DX,dfcb+1
        MOV     CH,11
        CALL    blkmov                  ; move filename to fcb
;
; setup fcb for subsequent file write operation
;
rset@fcb:
;
        MOV     DX,dfcb                 ; point to source fcb
        MOV     AL,Byte Ptr src@dsk@num ; force disk number, in case not logged disk
        MOV     SI,DX
        MOV     [SI],AL
        XOR     AL,AL                   ; clean-up for new file backup
        MOV     Byte Ptr .fcbext,AL
        MOV     Byte Ptr .fcbrno,AL
        MOV     Byte Ptr destfcb,AL
        MOV     Byte Ptr destfcb+12,AL
        MOV     Byte Ptr destfcb+32,AL
        MOV     Byte Ptr mfflgl,AL      ; turn off 1st time flag
        RET
;
;
;
;  subroutine to do block moves
;
blkmov: MOV     AL,M                    ; copy byte from [HL] to [DE]
        MOV     SI,DX
        MOV     [SI],AL
        LAHF                            ; bump pointers
        INC     BX
        SAHF
        LAHF
        INC     DX
        SAHF
        DEC     CH                      ; loop for count in [B]
        JNZ     blkmov
        RET
;
;  subroutine to index [BC] by file counter
;
filepoint:
;
        MOV     BX,Word Ptr filcnt      ; get file counter
        MOV     BH,0                    ; force hi ord to 0
        SHL     BX,1                    ; multiply by 32
        SHL     BX,1
        SHL     BX,1
        SHL     BX,1
        SHL     BX,1
        LAHF                            ; use as index to file table
        ADD     BX,CX
        RCR     SI,1
        SAHF
        RCL     SI,1
        RET
;
; subroutine to check for user abort (any key pressed)
;
abort:  PUSH    BX                      ; save all registers
        PUSH    DX
        PUSH    CX
        LAHF
        XCHG    AL,AH
        PUSH    AX
        XCHG    AL,AH
        MOV     CL,dircon               ; check direct console i/o (status)
        MOV     DL,0ffh                 ; input only
        INT     224
        OR      AL,AL                   ; set flags
        JZ      abortx                  ; return, if zero result
        CMP     AL,'C'-40h              ; control-c for abort?
        JNZ     abortx                  ; just return, if not
        MOV     DX,(Offset abort@process)       ; indicate we are aborting process
        MOV     CL,pmessg
        INT     224
        MOV     CL,0                    ; exit via CP/M warm boot
        MOV     DL,0
        INT     224
abortx: POP     AX                      ; restore all registers
        XCHG    AL,AH
        SAHF
        POP     CX
        POP     DX
        POP     BX
        RET
;
; subroutine to do carriage return/line feed
;
crlf:   MOV     DX,(Offset crlf@msg)
        MOV     CL,pmessg
        INT     224
        RET
;
;
;
signon          DB      cr,lf,'ARCHIVE (CTRL-C to Abort) - Ver.1.0'
                DB      cr,lf,cr,lf,'$'
;
ilgopt          DB      'Invalid or Unspecified Option - Must Be Specified As:'
                DB      cr,lf,cr,lf
opts            DB      tab,'B - Backup File Archive or,'
                DB      cr,lf
                DB      tab,'S - Set File Archive or,'
                DB      cr,lf
                DB      tab,'R - Reset File Archive or,'
                DB      cr,lf
                DB      tab,'D - Display File Archive'
                DB      cr,lf,'$'
;
no@file@fnd     DB      cr,lf,'File Not Found, Aborting$'
;
donmsg          DB      '  ','$'
;
crlf@msg        DB      cr,lf,'$'
;
bad@ver@num     DB      'Must be CP/M Version 2.2 to Archive, Aborting$'
;
req@dest        DB      'Destination Disk for BACKUP Files (A to P)? $'
;
same@dsk        DB      cr,lf,'Can''t BACKUP to Source Disk You TWIT!',cr,lf,'$'
;
abort@process   DB      cr,lf,'User Abort of ARCHIVE Process$'
;
narcs           DB      cr,lf,'No ARCHIVE Files Found to BACKUP$'
;
no@file@nam     DB      'No Filename or Option Specified - '
                DB      'ARCHIVE must be invoked as:'
                DB      cr,lf,cr,lf
                DB      tab,'ARCHIVE FN.FT OPTION<cr>'
                DB      cr,lf,cr,lf
                DB      'Where;',tab,'FN.FT is Filename and Filetype '
                DB      '(? and * allowed)'
                DB      cr,lf,cr,lf
                DB      'And;',tab,'OPTION is specified as:'
                DB      cr,lf,cr,lf,'$'
;
src@open@err    DB      cr,lf,'Oops, Can''t Open File on Source Disk$'
;
dsk@full        DB      cr,lf,'Destination BACKUP Disk $'
;
now@full        DB      ': is now FULL - Remove, insert NEW Disk$'
;
enter@ret       DB      cr,lf,'Enter RETURN when ready to continue BACKUP process: $'
;
src@read@err    DB      cr,lf,'Oops, Read Error on Source Disk$'
;
src@close@err   DB      cr,lf,'Oops, Bad Close on Destination Disk$'
;
bakup@done      DB      cr,lf,'BACKUP Complete',cr,lf,'$'
;
fname           DB      '           $'          ; 11 spaces for filename.typ
;
got@arc         DB      0               ; archive file found flag (1 = found)
;
mfflgl          DB      1               ; 1st time flag for multi-file access
;
mfreq           RS      12              ; multi-file requested filename
;
mfcur           RS      12              ; multi-file current filename
;
arch@addr       RS      2               ; archive patch address
;
logdsk          RS      1               ; current logged-in disk number
;
dest@dsk@num    RS      1               ; destination disk number
;
src@dsk@num     RS      1               ; source disk number
;
filcnt          RS      1               ; count of files in filetable
;
option          RS      1               ; storage for B, S, R or D option letter
;
destfcb         RS      33              ; destination disk file control block
;
bufmax          RS      2               ; buffer maximum size
;
bufcnt          RS      2               ; buffer sector count
;
bufpnt          RS      2               ; buffer pointer
;
eof@flg         RS      1               ; end-of-file flag
;
filetable       EQU     (Offset $)      ; file table/buffer starts here
                RS      4*4096          ; 16k buffer
table@end       DB      0               ; file table/buffer ends here
;
;
;
```

MICROCOMPUTER MONEYMAKER'S INFO SHEET

1. How long have you been involved with *micro*-computers?

2. What equipment do you use?

3. What off-the-shelf programs do you *rely* upon?

4. Do you earn money through the operation of your microcomputer?
   ☐ No ☐ Yes

4a. If no, why not? (use as many as apply):
   ☐ easier income doing something else
   ☐ more fun doing something else
   ☐ other things are more important
   ☐ independently wealthy
   ☐ sponging off of someone
   ☐ don't know enough about micros to make them work
   ☐ don't know enough about life to make it work
   ☐ haven't thought of or heard of any good ideas
   ☐ lack confidence
   ☐ need more information and advice
   ☐ need a counselor
   ☐ if none of the above, then what is it with you? (describe):

4b. If yes, is your microcomputer income from:
   ☐ salary?
   ☐ independent operation?
   (if both give relative %)

4c. If you used the second box in 4b, please indicate the activity, and the relative %.
   ☐ program royalties
   ☐ consulting
   ☐ equipment sales, rentals, repairs
   ☐ freelance writing
   ☐ publishing books, newsletters
   ☐ business or profession support for self (state what bus. or prof.):
   ☐ business support services for others
      ☐ bookkeeping
      ☐ word processing
      ☐ mail list management
      ☐ other (describe):
   ☐ other (describe):

5. Do you *know* about any interesting microcomputer moneymaking methods?
   ☐ No ☐ Yes (describe):

6. Have you *heard* of any interesting microcomputer moneymaking methods?
   ☐ No ☐ Yes (describe):

7. Can you *imagine* any interesting microcomputer moneymaking methods?
   ☐ No ☐ Yes (describe):

8. Would you enjoy a regular column on Microcomputer Moneymaking?
   ☐ No ☐ Yes

9. What topics would you most like to read about?

* Regarding descriptions: feel free to use extra sheets; keep in mind that concise is good, thorough is better, and both is bliss.

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

LIFEBOAT'S PRODUCT LIST NO. 22½

NEW — 16-Bit Software Available for the IBM PC, plus . . .

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

Telecommunications:
ASCOM

Languages:
Lattice C Compiler

DataBase Management Systems:
T.I.M. III

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

Media & Formats
IBM Personal Computer G1
GodBout.....E1
Seattle E1
TecMar E1

8-Bit Software Available

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

Telecommunications:
ASCOM
BSTAM
BSTMS
MicroLink-80
RBTE-80

Languages:
ALGOL-60
APL/V80
BASIC Compiler
BASIC-80
baZic II
Precision BASIC
BD Software C Compiler
CBASIC-2
CIS COBOL (Standard)
CIS COBOL (Compact)
COBOL-80
FORTRAN-80
KBASIC
muLISP/muSTAR-80
Nevada COBOL
JRT Pascal
Pascal/M
Pascal/MT
Pascal/M +
Pascal/Z
PL/I-80
STIFF UPPER LISP
S-BASIC
Timin's Forth
Tiny-C
Tiny-C Two
UCSD Pascal
Whitesmiths' C Compiler
XYBASIC

Language and Applications Tools:
BASIC Utility Disk
DataStar
FABS
FABS II
Forms 1 for CIS COBOL
Forms 2 for CIS COBOL
MAGSAM III
MAGSAM IV
MAGSORT
M/SORT for COBOL 80
Programmer's Apprentice
PSORT
QSORT
STRING/80
STRING BIT
SUPERSORT
ULTRASORT II
VISAM

Word Processing Systems and Aids:
Benchmark
DocuMate/Plus
MicroSpell
Letteright
Magic Wand
Spellguard
TEX
Textwriter III
WordIndex
WordStar
WordStar Customization Notes

Data Management Systems:
CONDOR
Formula
HDBS
Hoe
Microseed
MDBS
MDBS:DRS,.QRS,.RTL
dBASE II
PRISM/LMS
PRISM/IMS
PRISM/ADS
T.I.M. III

General Purpose Applications:
CBS
CBS Label Option
Selector III-C2
Selector IV

Mailing List Systems:
Benchmark Mailing List
Postmaster
Mailing Address
MailMerge for WordStar
NAD

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

Numerical Problem-Solving Tools:
T/MAKER II
fpl
PLAN80
Analyst
Microstat
muSIMP/muMATH
Statpak

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

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

Disk Operating Systems
Software Bus Family
SB-80
CP/M-80
MP/M

Hard Disk Integration Modules

Media & Formats

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

ADDS Multivision .....RT
ALSPA 8 in.....A1
Altair 8800 .....B1
Altos.....A1
Apple CP/M 13 Sector .....RG
Apple CP/M 16 Sector .....RR
Archives 1 .....SG
AVL Eagle II .....ST
BASF System 7100 .....RD
Blackhawk Micropolis Mod II Q2
BMC IF-800.....SR
California Computer Sys 8 in.....A1
CDS Versatile 3B Q1
CDS Versatile 4.....Q2
Columbia Data Products 8 in. A1
Columbia Data Products 5.25 in. ...S4
Commodore CBM/PET w/SSE Box + 8050 .....C2
Commodore CBM/PET w/Madison Z-RAM + 8050 .....C4
COMPAL-80 .....Q2
Computer Ops N.C. HQ S2
Control Data 110.....A1
CPT 8000 .....A1
Cromemco System 3.....A1
Cromemco System 2 SD/SS R6
Cromemco System 2 DD/SS .....RX
Cromemco System 2 DD/DS.....RY
CSSN Backup.....T1
Datapoint 1550/2150.....A1
DEC VT 18X.....SD
Delta Systems.....A1
Digi-Log Microterm II.....RD
Direct OA1000.....M2
DTC Micro 210A.....SC
Durango F-85 .....RL
Dynabyte DB8/2.....R1
Dynabyte DB8/4.....A1
Exidy Sorcerer + Lifeboat CP/M-80.....Q2
Exidy Sorcerer + Exidy CP/M-80 5.25 in.....RW
Exidy Sorcerer + Exidy CP/M-80 8 in. A1
EXO.....A1
Findex P6
Heath H8 + H47.....A1
Heath H89 + Magnolia CP/M-80 ..P7
Heath H89 + Heath CP/M-80.....P7
Helios II.....B2
Heurikon MLZ, SS .....SN
Heurikon MLZ, DS.....SO
Hewlett-Packard 125, 5.25in.....SB
Hewlett-Packard 125, 8 in. A1
IBEX 7100.....RQ
ICOM 2411 Micro Floppy R3
ICOM 3712.....A1
ICOM 3812.....A1
IMSAI VDP-40/VDP-42.....R4
IMSAI VDP-44.....R5
IMSAI VDP-80.....A1
Industrial Microsystems 5000 .....RA
Industrial Microsystems 8000.....A1
Intel MDS SD A1
Intertec Superbrain DOS 0.5-2.x....RJ
Intertec Superbrain DOS 3.x .....RK
Intertec Superbrain QD.....RS
ISC Intercolor 8063/8360/8963 .....A1
Lexitron VT 1303 DS/DD S8
Lexor Alphasprint Model S1 .....S1
Meca Delta-1 5.25 in. P6
MICOM 2001.....B3
MICOM 2001E.....B4
MICOM 3003 .....M1
Micromation A1
MicroMega 85.....SC
Micropolis Mod 1 .....Q1
Micropolis Mod II Q2
MITS 3200-3202.....B1
Monroe OC 8820, DD/SS .....SW
Morrow Discus.....A1
Mostek A1
MSD 5.25 in. .....RC
MULTI-TECH-I Q2
MULTI-TECH-II Q2
Nascom (Gemini drives).....R3
Nascom II with Lucas Drives.....SL
NCR 8140/9010.....A1
NEC PC-8001.....RV
Nicolet Logic Analyzer Model 764 .SX
NNC-80/80W.....A1
North Star SD.....P1
North Star DD P2
North Star QD P3
Northern Telecom 503 .....SM
Nylac Micropolis Mod II Q2
Ohio Scientific C3.....A3
OKI IF-800 + MSA CP/M-80 .....SP
OKI IF-800 + OKI/Lifeboat CP/M-80 .....SR
(Above OKI entries replace catalog entry for OKI IF-800 format code RZ)
Pertec PCC 2000.....A1
PET/CBM w/Small Systems Engineering Box + 8050.....C2
PET/CBM w/Madison Z-RAM + 8050 .....C4
Philips MICOM 2001 8 in.....B3
Philips MICOM 2001E.....B4
Philips MICOM 3003 .....M1
Processor Technology Helios II B2
Quay 500.....RQ
Quay 520 .....RP
RAIR DD.....RE
Research Machines 5.25 in. .....RH
Research Machines 8 in. A1
Sanco 7000 5.25 in.....RQ
Sanyo MBC 1000.....SY
Sanyo MBC 2000.....SS
Sanyo MBC 3000.....A1
SD Systems 5.25 in R3
SD Systems 8 in. A1
Spacebyte.....A1
Tarbell 8 in. A1
TEI 5.25 in.....R3
TEI 8 in.....A1
Televideo DD/DS S5
T.I.P. (Alloy Engineering, Inc.).....T3
Toshiba T200.....SF
Triumph Adler Alphatronic .....SV
TRS-80 Model 1 + Shuffleboard 8 in. A1
TRS-80 Model II A1
Vector MZ Q2
Vector System 2800 .....A1
Vector System B/VIP.....Q2
Vista V-80 5.25in. SD.....R8
Vista V200 5.25in. DD P6
Wangwriter.....SE
WORDPLEX .....SZ
XEROX 820, 5.25 in.....S6
XEROX 820, 860 8 in.....A1
ZEDA 580 .....SH
Zenith Z89 + Magnolia CP/M-80 ...P7
Zenith Z89 + Zenith CP/M-80.....P7
Zenith DD/SS.....SK
Zenith DD/DS.....SJ

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

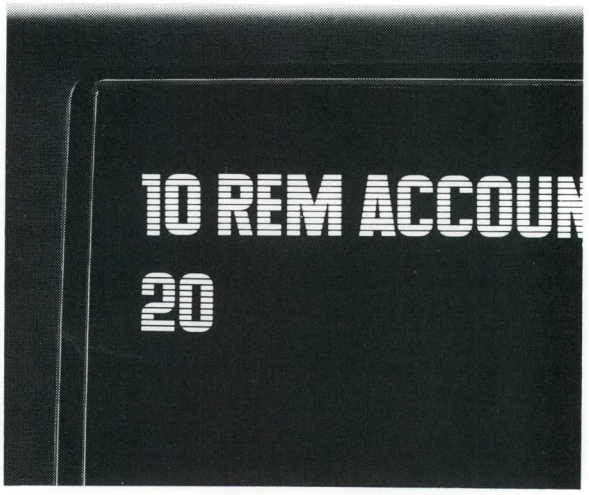
Send for full Lifeboat Associates Software Desk Reference with descriptions of all the above plus a whole lot more.

LIFEBOAT HAS THE ANSWER

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

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

BOY, IS THIS COSTING YOU.



It's really quite basic: time is money.

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

Especially when you could speed things up with dBASE II.

dBASE II is a complete applications development package.

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

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

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

Instead, you write your code in concepts.

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

dBASE II uses English-like commands.

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

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

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

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

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

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

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

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

You can do automatic calculations on fields, records and entire databases with a few keystrokes, with accuracy to 10 places.

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

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

Expand your clientbase with dBASE II.

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

Your competitors know of this offer.

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

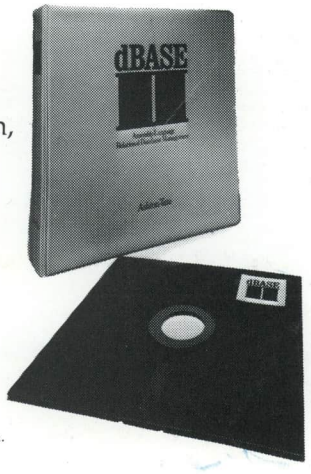
Call for our Dealer Plan and OEM run-time package prices, then take us up on our money-back guarantee. Send us your check and we'll send you a copy of dBASE II that you can exercise on your CP/M® system any way you want for 30 days.

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

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

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

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



Ashton-Tate

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

Also available from Lifeboat Associates.

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

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