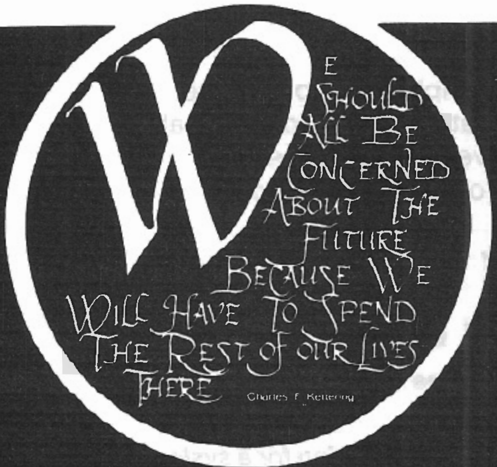
$5.95 U.S.

two/sixteen magazine

Volume 1, Number 6 May–June 1983

The journal for business, professional, and scientific members of the TRS-80 community



CONTENTS

LUCID

Does a powerful data base management system have to be complex, costly, and confusing?

If you're really interested in a data base management system with MBASIC code generation, acronym-based query languages, relational access paths, multi-key ISAM files, data dictionaries, machine language utilities, multi-user access, multi-level indexed dynamic pointer arrays, parameter files, data encryption, inverted name fields, many-to-many recursive relationships, and virtual buffer paging—then you could buy some of these...

dBASE II
QUICKCODE
dUTIL
dBASE WINDOW
TIM III
IDM-X
Maxi-Manager
CCA Data Manager
Infostar
FMS-80
MDBS
Selector IV
Condor
Profile

But, if you simply need to manage your information easily, economically, and effectively—then there's really only one system for you...

LUCID

Power borne of Simplicity

LUCID packs all the power you need in one genuinely easy-to-use package

It's the simplest data base management system of all...and second to none in its capacity to get your job done.

Yes, LUCID does use advanced techniques, such as machine language utilities. They're programmed to function automatically and effortlessly, when needed. So the only way you know they're working for you, is from the resulting speed and efficiency.

There's no delay between learning time & productive time

Every action has been designed by experts in human-computer interaction...based on years of research and development. The result: a system that puts full power at your fingertips instantly. And without costly training courses or time-consuming installation.

Novice and expert alike can learn LUCID in minutes without a manual, right at the keyboard. You learn by actually sorting files, generating reports, changing information, and screening-out selected records.

For example...

When you want to select only certain records, most other systems either require a complex query language statement...or limit the conditions you can choose or combine.

Instead, LUCID gracefully guides you step-by-step, in plain English. So you can select any records you want, for any operation you need, with no complex searching language...no rigid statement formats...and no limitations.

And, on two-drive systems, this fully integrated DBMS offers you word processing capabilities, too. Now you can generate reports and letters while you manage your records, with one efficient system.

Looking for a system to suit yourself, your customers, or your clients?

Then LUCID is designed for you. It saves valuable time and effort from initial installation through actual use. So everyone can spend more time getting the job done.

Haven't you done without a data base management system long enough? Send for our demonstration disk ($25.00) or complete system ($238.00), and get a taste of real power in just minutes.

Call **1-215-628-0400**
Or write to:

SofterWare INC.
200 Office Center
Fort Washington, PA 19034

Trademarks: dBASE II/Ashton-Tate, Inc.; QUICKCODE & dUTIL/Fox & Geller, Inc.; dBASE WINDOW/Tylog Systems, Inc.; TIM III/Innovative Software, Inc.; IDM-X/Micro Architect, Inc.; Maxi-Manager/Adventure International; CCA Data Manager/Personal Software; Infostar/Micropro; FMS-80/Systems Plus; MDBS/ISE; Selector IV/Micro-Ap; Condor/Condor; Profile/Tandy Corp.; MBASIC/Microsoft, Inc.

two/sixteen magazine

Editor In Chief/Publisher	Richard H. Young
Managing Editor	Barbara S. Albert
Art Director	Timothy P. Schleif
Administrative Assistant	Fay L. Smith
Director of Advertising & Circulation	Beverly J. Shank
Contributing Authors	Arnold Fischthal David Gash Roger Conant David Otten Robert VanNatta Jim Kloosterman

Typesetting by	Innovative Ink Lancaster, PA

Printed by	The McFarland Company Harrisburg, PA

two/sixteen + 12 (ISSN 0736-8194)
is published bi-monthly by

two/sixteen + 12

131 East Orange Street, Lancaster, PA 17602
Entire contents copyright 1982
by Richard H. Young
All rights reserved

Telephones

Editorial Staff	(717) 397-3365
Classified Advertising	(717) 397-3789
Display Advertising	(717) 397-3789
Subscriptions	(717) 397-3364

Annual Subscriptions:

Third Class	$30 in U.S.
First Class	$36 in U.S.

VISA and MasterCard Accepted

Foreign Subscriptions:

Canada and Mexico	$36 U.S.
Other Foreign:	$47 U.S.
Air Mail Foreign:	$67 U.S.

Drawn on U.S. Banks

Send all subscription correspondence,
including change of address (with label) to
two/sixteen +12
Box 1216, Lancaster, PA 17603

two/sixteen + 12 welcomes letters to the editor. All letters sent to the magazine become the property of **two/sixteen + 12 magazine** and are subject to publication unless otherwise requested by the author.

Editorial

With this issue *two/sixteen* enters its second year of publication (although, due to the date change on the Sept-Dec issue, subscriptions that began with May-June, 1982, do not expire until June 30, 1983). Renewals are coming in very well, and may amount to 90%. Our subscribers now number 3500, and with the addition to staff of Beverly Shank, Director of Advertising and Circulation, we expect to reach our projected break-even volume of 8000 by the end of 1983.

We're always interested in purchasing lists of Model II, 12, and 16 owners. If you have such a list, or know of any, please call Beverly at (717) 397-3789. We pay a premium price ($9/hundred on diskette). Also, our mailing list (currently 9500 names in all) is available for rent to those selling products designed to aid II/12/16 users. And we've just introduced a "co-operative" mailing program (first class mailing up to five vendors' flyers in one envelope), which can reduce the cost of a mailing by as much as 60%.

Xenix

Now that XENIX has burst upon the scene, we're getting a lot of calls asking for assistance with it. Unfortunately, we are not able to be of much help yet. Patches to XENIX which will allow use of our Corvus 20-megabyte hard disk are not yet available, and we understand that Corvus will not develop them until they are reasonably sure that the Shack has finished with any further Model 12/16 hardware aberrations. I'm not willing to put up with the RS hard disk because it is too costly and backup is only to floppy disks.

In the meantime, anyone interested in using XENIX should subscribe to *UNIQUE*, subtitled "Your Independent UNIX and C Advisor." It's available from InfoPro Systems, P. O. Box 33, East Hanover, NJ 07936, (201) 625-2925, at $54 per year ($12 additional for foreign air mail). The February issue of this monthly publication contains 12 pages. A few will not understand why this small specialized publication should cost so much. I suggest that such people are in the wrong business. The cost of *UNIQUE* is trivial compared to other XENIX costs. *UNIQUE* also provides a free public access bulletin board devoted exclusively to UNIX and the C language. Call them for details.

We're beginning to get reports that suggest, although not conclusively, that XENIX on the 16 may be unacceptably slow. An engineer reports that his MBASIC interpreter programs run at only 70% of the speed that they did under the TRSDOS 2.0 RSBASIC interpreter, and this is with just a single user. As readers obtain experience with XENIX, we'd like to receive information about comparative timings and other impressions. We'd especially like to hear about application software that runs on the 68000, regardless of operating system.

Information has just reached us about a two-day UNIX course being held at four locations (Hasbrouck Heights, NJ, June 16-17; Philadelphia, PA, June 23-24; Washington, DC, June 27-28; Anaheim, CA, July 7-8). This course is unbelievably expensive at $695, and I suspect that only large corporations will send representatives. If interested, contact The American Institute for Professional Education, 100 Kings Road, Madison, NJ 07940, (201) 822-1240.

Multi-user Comments

I've always been very dubious of the value of multi-user systems on microcomputers. First, there's the possibility that one user can bring down the system (and the other users with it). Second, the overhead required to minimize this possibility can be substantial. Third, if memory is not adequate for all jobs, swapping will take place; in this case it often takes far longer to run each job than on a single user machine. In fact, it will frequently take longer to complete *any one job* of several concurrent jobs than it will take to do *all the jobs* consecutively on a single user system.

The greatest appeal of the microcomputer (as compared to timesharing systems) is its "personalness," allowing each user freedom from interference by other users.

Editorial

Because the time of professional people using micros is worth far more than the time of the most expensive micro (probably at least ten to one), I predict that the misguided multi-user fad will die out in one or two years. Those who need either simultaneous access to common data or interprocess communication will migrate to the relatively inexpensive "highway" networks such as the Corvus OMNINET.

There is, however, considerable appeal to multi-tasking single-user systems. This method (an example is "concurrent" CP/M) allows a user to run one or more "background" jobs (which don't affect the CRT) while running a "foreground" job on the CRT. Under this arrangement, you could run a long sort in the background while updating other files in the foreground. Since you have control of all the jobs, contention is minimized and total throughput can be optimized.

Radio Shack Model 100 Portable Computer

On the day it was announced, we bought the first 24K Model 100 to be delivered to our local Computer Center. The machine is ideal for writing and editing "in the field." It's 12 x 8 x 2 inches and weighs four pounds. It includes a standard typewriter keyboard, a 40 character by 8 line LCD screen, a built-in 300 baud modem with auto-dial and auto-logon, an RS-232 port, a printer port, a bar code reader port, a cassette port, and a BASIC with more features than the RS Model II BASIC. It has numerous other features, but the main reason we bought it was for writing in airplanes, trains, and meetings. The copy can then be "down-loaded" to our 16 and incorporated with our other editorial material.

COBOL Bug

We have a report that Ryan-McFarland's COBOL contains a serious bug which has not yet been resolved. It seems that when a sequential file ends at a sector boundary it is impossible for COBOL to sense end-of-file and the job aborts to the operating system. Until this is fixed, users may be able to avoid this kind of crash by adjusting record sizes so as to minimize the probability that the last record will end on a sector boundary.

Tandy Business Users' Conference

The "First Annual Users' Conference" is scheduled for May 9-11 at the Americana Hotel in Fort Worth. I plan to attend. The press release from Radio Shack implies that Tandy is cooperating wholeheartedly with the user group program. This is good news, as some manufacturers in the past have greatly feared user groups and have discouraged or ignored them. This cooperation from Tandy is especially welcome in view of the continuing flurry of complaints we get about Tandy's failure to adequately support its software (although reports on hardware service are, with a few exceptions, excellent). Perhaps the group can convince Tandy to improve software support in general and to support CP/M-68K so that users who cannot (or should not) afford the XENIX system will still be able to utilize the 68000's power.

My experience with other computer user groups (IBM, DEC) indicates that Tandy may be in for a good deal of harsh and even vindictive criticism. They will almost certainly receive a lot more concentrated pressure than they have been subjected to before. We'll report on this meeting in the next issue.

Programming Services

Users who need to obtain programming services may wish to be listed in the monthly *Directory of Software Buyers* published by the National Association of Free-Lance Programmers. Contact them at Box 813P, Vienna, VA 22180, or call the president, Dick German, at (703) 938-1203.

CBASIC/CB-80 Users Groups

Caddy McCall, P. O. Box 2365, Sherman, TX 75090 is interested in forming users groups for CBASIC (a Digital Research CP/M-based product which was added to the Radio Shack line in January, though not yet delivered) and CB-80, reportedly one of the fastest micro BASICs. It appears that emphasis will be on small business use of these languages.

IBM 9000 Microcomputer

To say the least, I'm not a fan of Big Blue. For over 20 years they have been anathema to me, what with their anti-competitive practices, suppression of technology to protect existing rental bases, very high price/performance ratios, and general hornswoggling of the public. However, a recent letter from Jim Kloosterman, one of our regular contributors, indicates that they have (perhaps inadvertently) come up with a 68000-based microcomputer that may outstrip the market in performance. As you will see, this machine is no relative to the puny PC toy computer, but has been developed by an independent group of scientists with serious work in mind. We herewith reproduce verbatim that portion of Mr. Kloosterman's letter which pertains to this marvel:

IBM MAY HAVE A SUPERIOR 68000 COMPUTER AND NOT EVEN REALIZE IT!

Another "out-lying group" of The Giant (maybe not unlike the "renegades" in Boca Raton) have a desk-top-sized machine. And, contrary to so many adequate-but-not-outstanding IBM computers, this one has ALL the latest "goodies" plus. After three calls to what proved to be non-working 800 numbers, having my two collect calls refused, and being shuttled between two 250-mile-distant sales offices and waiting 24-hours for a rep to call me back, I got a salesman who'd never heard of an IBM 9000 or a 68000 chip! He promised to get the information, but I haven't heard from him in the 2+ weeks since! IBM once had superior marketing anyway!

Meanwhile, I saw an ad for the machine — with an 800 number! Called that and got a very pleasant gal who assured me they were prepared to make immediate shipments — including BASIC — and agreed to send me literature and prices BUT APOLOGIZED THAT IT MAY TAKE TWO WEEKS "because it has to go through Boston office!" Exactly 14 days later, I received it. If you haven't seen "the poop" on this IBM 9000 Instruments Computer, call 800-243-7054. That's the Eastern Support Center (and plant?) of IBM Instruments Inc, Orchard Park, P.O. Box 332, Danbury, CT 06810.

Model numbers must be out-of-style at IBM: Like the PC, all the literature on this big-little 9000 beast says simply "The IBM Instruments Computer System." Apparently, this bunch makes the IBM LC/9533 Liquid Chromatograph (lab analyzer). I've heard that the many of the newest lab instruments are "pushing" the capabilities of a Z-80 chip for data-gathering/processing. More powerful li'l computers are needed to realize the instruments' potentials. It seems likely that's why this group has a powerful, new computer.

They list three versions: General Purpose, LC/934, and Chromatography Application Program. The latter two are for their lab-analyzer (with 128K ROM program, I think) or for other, similar instruments with applications program using RAM. I don't fully understand all the options yet.

The standard General Purpose System includes a 68000 CPU on 8 MHz clock. (ALTOS 68000 is also 8 MHz, but Radio Shack uses the "reject" 6 MHz-rated chips. That means the others are 33• faster internally!) Standard RAM is 128K and it goes up in 128K byte increments ON 1 MEGABYTE CARDS! (A 5-slot Expansion Feature is $99 option.) Also standard are three RS-232C serial ports, one 8-bit parallel I/O port and one IEEE-488 parallel I/O port. Standard too is a year/month/day/hour/minute/hundredth-second clock with battery back-up that is ACCURATE (and is not turned-off for disk operations!). Plus, three 16-bit timers. A system bus accepts Motorola VERSAbus cards (whatever those are?). And, there are 32 programmable interrupts on four hardware levels with seven hardware levels total, along with four Direct Memory access channels at 1 MHz maximum (using 250 ns RAM).

The CRT display offers 12-inch raster scan, green-on-black, tilt and swivel adjustment, 30 lines of 80 characters and super-high-resolution of 768 x 480 pixels! At the bottom of the screen are ten programmable "soft" keys (a la Hewlett-Packard). "The display software facilitates separately-pageable windows."

IBM's operating system is referred to simply as "the multi-tasking, real-time operating system." It permits "several, concurrent system activities" (like data-gathering, processing and reduction, display and plotting in real-time, possibly plus some general-computational "background" tasks — but not multi-users). "The operating system permits easy use of disk files. Sequential, contiguous and random files of either binary or ASCII character types can be created, listed, changed or deleted." (What in heck are "contiguous" files?)

Apparently, one can mix-or-match up to four disks with both 5-1/4" and 8" diskettes (BOTH DOUBLE-DENSITY/DOUBLE-SIDED !!) or 5 Megabyte or 10 Megabyte hard-disks.

A little weirdly (because it's a lab instrument processor), THE STANDARD MACHINE DOES NOT COME WITH A KEYBOARD! Instead, there's a "keypad" with 57 user-definable keys and six keys with program-controlled LEDs. It uses overlays for semni-dedicated applications (that could be VERY user-friendly!).

The $270 optional keyboard has 83 keys, full ASCII plus a numeric keypad AND 10 programmable function keys. All repeat if held-down. The keyboard is moveable on a flexible cord. (If you like to push keys, you'll love this! With standad keypad and CRT program-keys plus optional keyboard, they total exactly 156!)

The $2,095 optional, integrated Printer/Plotter takes 8-1/2" regular or 9-1/2" pin-feed/fan-fold paper. Bi-directional, a wire-matrix, "impact-head" runs 200 char/sec in "draft mode." IT HAS A FOUR COLOR RIBBON and plots with superb 220 dots/inch horizontal and 336 dots/inch vertical resolution! "High-quality graphics, such as chromatographs and spectra, can be produced in various colors." A single key transfers a hi-res display to the printer.

BASIC — on 5-1/4" or 8" disk — is a $195 option. Something called an "Operating System Extension" is listed at $155. Options for FORTRAN or PASCAL (available now?) are listed at $595 each.

Base price (128K, no keyboard, disks, or printer) is $5,695. That's 31.5 pounds and only 22.3 x 17.2 x 6.7 inches! Apparently, one or two 5-1/4" floppies can be mounted in the display: the 8" drives use separate cabinets. As usually photographed WITH the CRT on top of "bridge" over the integrated printer/plotter/paper-tray and keyboard on front "shelf," it's 78.8 pounds and 22.3 x 29.5 x 23.2 inches — still very compact on the table-top!

I assume the system is now priced based upon expected sales to analytic-labs ... speculate upon re-pricing for much, much higher volume if "pushed" as an Advanced Small/Medium Business System by IBM's awesome Marketing Army!

VanNatta on

COMPUTER CHIPS

Robert P. VanNatta
2 S. First Street
St. Helens, Oregon 97051

Why would anyone want a computer with a Motorola 68000 chip in it? Since I ordered my Model 16 last year, I have been waiting for someone to answer that question for me. A satisfactory answer has not been forthcoming, so I shall undertake an explanation of the relative merits of the chips myself.

One of life's paradoxes is that those who understand the technical difference between a 8080, a 8088, and a MC68000 are either unwilling or unable to tell anyone about it, and those who have the ability to write and explain such differences usually lack even the faintest grasp of the difference between the various processing chips.

The disparity between engineer and writer results in most publications simply mentioning that the 68000 microprocessor is "more powerful" and leaving it at that.

I am not an engineer, but rather a country lawyer, and at the risk of making a complete fool of myself, I shall undertake to describe some of the differences in the various microprocessor chips.

What's An 8-Bit Chip?

To have any chance at all of understanding microprocessor architecture, the first thing you need to understand is the difference between an 8-bit chip and a 16-bit chip. Once you understand this distinction, you will have the world by the tail.

Unfortunately, there seems to be no general agreement on a definition of the difference. It is fairly easy to define a bit or a byte. A bit is a binary part of a number. For example, if you have ever seen a computer working on paper punch tape, you can understand a bit to be a hole (or non-hole) in the paper tape, and a byte to be a row of 8-bits, which in turn represent a number or character. In a rough sort of way, you can also visualize the dots and dashes of Morse code to be bits, and the combinations which represent a character to be a byte.

The basic unit of the microcomputer is the byte, which in turn consists of 8-bits. Because of its 8-bit makeup, it is possible to represent hexadecimal numbers from 00h to FFh (0 to 255 in decimal notation) within a single byte.

Chips such as the Z80 and the 8080 are said to be 8-bit chips because they generally work a byte (8-bits) at a time.* The 8080 chip is designed around seven 8-bit registers, a 5-bit flag register, a 16-bit stack pointer and a 16-bit program counter.

What's A Register?

A register is simply a logical location for putting a number before doing something with it. For example, the command 'ADD L' is the assembly language command for adding the contents of register "L" to the contents of register "H". The result would be placed into register "A".

Register "A", called the accumulator, and registers "H" and "L" are probably the most important in the 8080 chip. The accumulator is important because it returns the results of all 8-bit arithmetic operations. The H-L register pair is important because those two registers can be paired to hold a 16-bit memory address. Not surprisingly, the "H" register holds the high order byte and the "L" register holds the low order byte.

I will explain what is meant by a high-order byte in a minute, but the important point here is the significance of the ability to pair two registers. A single 8-bit register can only hold 256 different numbers, because that is the maximum number of different combinations of 8 binary bits which by definition can only be "off" or "on".

The ability to pair the registers for loading a memory address effectively means that instead of being limited to a mere 256 different memory addresses, it is possible to address a total of 65536 (64k) different memory locations with the 8080 microprocessor. Have you ever wondered why the Z80-8080 computers have no more than 64k of memory? Mathematically, only 64k of different addresses can be represented by the 16 bits of the H-L register pair.

The Data Bus

The input and output (I/O) for a microprocessor is handled through what is known as a data bus. The data bus is labeled by the physical manner in which the microprocessor is "wired in." An 8-bit data bus is "wired in" with eight connections so that a byte at a time can be moved in to and out of the microprocessor. The 8080-Z80 microprocessors have an 8-bit data bus.

*For purposes of this article, I shall treat the Z80 as being the same as an 8080 even though it has a number of features not present in the 8080. My justification for this is that these extended features are rarely implemented by programmers (Digital Research in particular) due to a general desire to retain compatibility with the 8080 instruction set.

Since you can't move 16 items of data at once down eight data paths, it is logically necessary to break the 16-bit number in half and move it eight bits at a time. Reference to a high-order byte refers to the order in which the bytes are moved through the bus.

The Intel 8086 And The 8088

If you are a devoted Shackie, you probably have heard very little about the Intel 8086 and 8088 microprocessors. Unless the Shack makes some surprise product announcement after I write this article, and before it gets into print, I believe it is correct to say that the Shack does not use either of these chips.

The IBM PC, which has hogged its way into position as one of the three top selling computers in the nation, is built around the 8080. IBM presents the 8088 as a 16-bit chip. Curiously, however, until IBM declared that it was a 16-bit chip, most of us thought of it as an 8-bit chip. The reason for the confusion is that the 8088 has an 8-bit data bus for communication with the rest of the world, but, internally, contains 16-bit registers which execute the 8086 instruction set.

If you regard the size (horsepower) of a chip as being defined in terms of the size of the data bus, the 8088 is unquestionably an 8-bit chip. IBM calls it a 16-bit chip because it has 16-bit registers, but then if you apply this yardstick to other chips, you must wonder if the 8080-Z80 family members are properly called 16-bit chips as well. After all, the program counter and stack pointer are 16-bit registers and six of the other seven registers can be paired for 16-bit use. Along the same lines, we have previously noted that the 8080-Z80 will address memory defined by a 16-bit address. Additionally, certain 16-bit instructions are supported. For example, the instruction "ADD D" will combine the contents of registers D & E with registers H & L, effectively performing a 16-bit addition.

As we shall see presently, if we measure the Motorola 68000 chip by the same yardstick that calls the 8088 a 16-bit chip, the Motorola 68000 measures up as a 32-bit chip. (It has 32-bit registers and a 16-bit data bus.)

What's Wrong With The 8088-8086 Microprocessor?

As far as the programmer is concerned, the 8088 and the 8086 are identical chips. Their internals are the same. They execute the same instruction set. The difference in the data bus affects only the way the chip is wired in and, to some extent, the overall performance in terms of speed. Otherwise, they are the same.

The similarity of the 8086 architecture to the 8080 is both its strength and its weakness. The strength in the similarity is that transportation of programs from the 8080 instruction set to the 8088 is less of a chore than it might be if the 8088 were radically different. The weakness is, of course, that the 8088 is not profoundly better than the 8080-Z80.

The 8088 expands all registers of the 8080 to a full 16-bits, and provides some additional registers. It does not, however, allow register pairing, in the sense that I described for the 8080. A major improvement in performance is provided by the fact that the accumulator is a 16-bit register, but otherwise computational horsepower is not that much better than the 8080, simply because the 8080 supports quite a few 16-bit operations itself.

A major enhancement of the 8088 over the 8080 is in the area of memory management. As mentioned earlier, the 8080 can address only 64k bytes of memory without resorting to devious software tricks such as bank-switching.

The 8088 can conceptually address up to a megabyte of memory by use of memory segmentation, which means essentially that bank-switching is built in. Memory segmentation is accomplished with four segment registers: the code segment (CS) register, the stack segment (SS) register, the data segment (DS) register, and the extra data segment (ES) register. With each memory operation, four bits are extracted out of the appropriate register and combined with the 16-bit address in one of the other registers to form a 20-bit address.

By the nature of the beast, only four 64k byte segments may be active at any one time. If you think about it for a moment you should begin to grasp the enormous possibilities for implementation of multiuser activities and such things as Concurrent CP/M on the 8088. The segment registers provide natural divisions of memory and invite the development of software that involves getting several things going at once in different segments. It is a small wonder that Concurrent CP/M (to support exactly this kind of juggling) hit the bricks almost as soon as CP/M-86 itself.

The problem with memory segmentation is that it makes life very complicated if you want to put together a very large portion of memory for a single activity. For example, as I understand it, most of the high level languages now implemented on the 8088 require that all the code be located in a single code segment. The bottom line on this is: unless you are pretty clever at writing assembly language bank-switching routines, don't expect to load a program that consists of more than 64k of code.

This is, of course, better than the 8080 where you have to fit the code, the stack, and the data all in 64k, rather than having 64k for each. But then I didn't say that the 8088 wasn't more powerful, just that it wasn't that much more powerful!

What About CP/M PLUS?

By now you surely have heard of CP/M PLUS, also known as CP/M version 3.0. Likewise, you probably also know that it will support more than 64k on the II/12/16 computers. "How is this?", you ask.

The answer is, of course, that clever programmers can be expected to accomplish the impossible if given the incentive. There is nothing about CP/M PLUS that is inconsistent with what I have said here. What CP/M PLUS does is to implement bank-switching on the 8080-Z80 version of CP/M. If you have a kitchen with a single plug-in and happen to own both a toaster and a waffle iron, you understand the concept of bank-switching. Stated another way, bank-switching permits use of additional memory by recognizing that it is feasible to redirect the memory output to alternate memory boards.

Such extended-addressing, as it is called, has its limitations, however. The problem arises from the need to avoid confusing the system during the switching process. Programs that involve bank-switched memory must be rigorously structured, as, by definition, bank-switching involves use of multiple-memory segments with the same address. Translated into English, this effectively means that the activities in the various banks must be isolated by function and the communication between the banks can only be circumspect and indirect.

Digital Research has dealt with these limitations very logically. The operating system itself is a very logical target of bank- switching because its interface with the user programs is already very structured by use of the standardized function calls. CP/M PLUS accomplishes its magic by loading itself primarily in the alternate bank. Function calls, which represent the exclusive interface between the applications programs and the operating system, are routed to a small portion of memory that is not switched. When a function call is made, it is logically a simple task to save the registers, switch to the alternate memory bank, perform the function, load the result in the return registers, restore the other registers, and switch back to the main memory.

The advantage of this is that the operating system no longer competes with the applications program for space. Thus, the operating system can be made larger, and hence more user-friendly, and the file buffering can be made larger, and therefore faster, without gobbling up precious applications program area.

The Motorola 68000

By contrast to the kludge characteristic of the 8088 architecture, the Motorola 68000 is elegant. The instruction set is relatively small and dreadfully powerful. The Motorola 68000 contains eighteen registers (compared to 14 on the 8088). All the registers are 32-bits in size. There is a status register and user stack pointer (USP), together with eight data registers and eight address registers.

The eight data registers are identical, as are the eight address registers. Additionally, the difference between the data registers and the address registers is only minor. (The address registers will do indirect addressing and will not do byte arithmetic operations.)

With 32-bit registers available for memory addressing, memory segmentation is not a relevant issue. When you understand that 8-bits will address 256 bytes of memory, that 16-bits will address 65535 bytes of memory, and that 20-bits will address over 1-megabyte of memory, you can't help but recognize that 32 bits will address more memory than you can afford.

The Motorola chip does not have an accumulator, or a primary data pointer as it is known the in 8080 microprocessor. All of the data registers are identical, as are the address registers. It is incumbent upon the programmer to design some structure into his programs by making arbitrary assignments of the registers for various purposes. The Radio Shack assembler identifies the address registers as .A0 to .A7 and the data registers as .D0 to .D7. As an example, the command ADDL .A0,.A1 will perform a 32-bit addition of the contents of register .A1 to the contents of register .A0, leaving the result in .A0.

What To Look For If We Ever Get Any Software For The 16

The implementation of BASIC and PASCAL on the 68000 chip should provide a new standard of performance. One of the obvious things such an implementation should do is invite the use of 32-bit integers to replace the 16-bit integers common to existing languages, which will result in almost all whole numbers being processed as integers. This should perk up many number crunching routines! Likewise, for real number operations, the 16-bit register will not be a limitation, and we can expect BCD math operations to conduct mathematical calculations to considerably more than 14 digits of significance. (I understand that 20 digits of significance would not be a strain.)

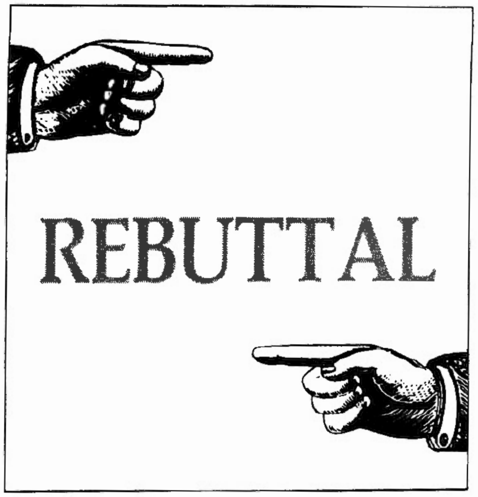
There will be no excuse for any high level language failing to support strings to 64k in length. The tendency on the 8080 has been to use a one-byte string length pointer, which of course limits overall string length to 256 characters. I doubt if we will see many one-byte pointers flopping around in the 4-byte registers.

It would appear that the 68000 microprocessor will allow for brute-force processing using about as much memory as you can install. However, there is no inherent reason why concurrent processing cannot be implemented on the 68000, and it is reasonable to expect that to be the norm.

Conclusions

The brute horsepower apparent from a review of the architecture of the 68000 suggests that, once implemented, it should perform considerably better than the 8088. None of the benchmarks I have seen show the 8088 walking away from the Z-80 in performance, and with the advent of bankswitch operating systems for the Z80 such as CP/M PLUS, it appears to me that the Z-80 chip is holding its own quite well from a performance standpoint with the 8088, at least in the single user field.

The Z-80 is having a run for its money due to the increasing popularity of the 8088. However, the reason for this is more attributable to marketing horsepower than to computer horsepower. By contrast, the MC68000 has the computer horsepower, and, with the Apple Lisa and the Radio Shack Model 16 both using it, we should expect new standards of performance as 68k software makes its way to the market.



REBUTTAL

Mr. William Strating of Holland, Michigan reviewed H & E Computronic's General Ledger Accounting Package, VERSALEDGER II, for **two/sixteen + twelve magazine,** *and his review was published in the March/April issue (Vol. 1, No. 5, pp. 34-38). Richard Kaplan, author of VERSALEDGER II, has taken the time to respond to the review, and his comments may be of interest to you.*

I just received a copy of your review of VERSALEDGER II for *two/sixteen magazine*. The review seems to be generally balanced; however, there are several specific points which I believe merit qualification.

In your review, you mentioned that "If Supporting Schedules are what you want and expect, you will not find them in VERSALEDGER II." Quite to the contrary, VERSALEDGER II has *better* facilities for generating supporting schedules than any general ledger program I have seen, even at a much higher price. The FORMAT REPORTS option can be used, as you alluded in your review, to format a report which looks precisely as the user wants it to look. It is suggested that FORMAT REPORTS be used to generate income statements and balance sheets; however, it would be perfectly feasible to generate supporting schedules as well. If you have a specific supporting schedule you would like to duplicate, I would be glad to show you how.

Another area which I feel may be quite misleading is the paragraph which reads, "VERSALEDGER II offers you no protection against the possible posting of a check register more than once. *"Transactions are posted immediately... ."* This, surely you will agree, is dead wrong. Transactions in the check register are posted *only* when the specific option to do so, POST CHECK REGISTER, is done. This is clearly outlined in the manual.

As far as the posting procedure being a "potentially serious flaw," I must respectfully disagree. Let me give you some background on the development of VERSALEDGER II. Many other accounting systems, including some that I had supported prior to writing VERSALEDGER II, required the user to "close out" a period at the end of a month, and after this closing out the user could never go back again. If a transaction had to be entered, there was nothing the user could do. This was a particularly common criticism, and therefore VERSALEDGER II was designed to not have this design flaw.

The VERSALEDGER II check register is posted to the check register at the end of every month. However, since we are dealing with the real world, it is quite possible, as I am sure you realize, that a disk could go bad or that another check has to be entered and then the checkbook has to be posted again or any of numerous other scenarios. Therefore, there intentionally is no check against posting the check register twice. I think, and our customers seem to agree, that the slight inconvenience of making sure a check register is posted only at the end of the month *more* than outweighs the frustration of being forced into an unyielding computerized system. Don't you agree?

The same concept regarding VERSALEDGER II's being designed with user flexibility in mind applies to your criticism of the option which allows an unbalanced journal entry to be made. In fairness to all involved, let me first explain this option *in context with the other* journal entry option *not* specifically mentioned in your review.

VERSALEDGER II has two options on the GENERAL LEDGER SUBMENU with which to enter journal entries — the JOURNAL ENTRY option and the MULTIPLE JE option. The former is the option which allows an "unbalanced entry" to be made. In theory, a computer should never allow the system to become out of balance. However, our experience with other accounting systems (and this parallels our experience with the posting feature described above) has shown that, in the real world, accounts really *do* become out of balance no matter what. There are two reasons for this: (1) Sometimes a disk may be damaged, destroying half a company's data; and (2) Some companies, for whatever reason, actually *want* their reports to be out of balance at certain times. Don't ask me why, but I have had this request before.

VERSALEDGER II was designed not to force arbitrary restrictions on its users. Therefore, everyone has the *option* of using the MULTIPLE JE option, which does force the system to be balanced, or the JE ENTRY option, which allows unbalanced entries. If you do not want your system to be out of balance, simply don't use the one-sided journal entry option. (If someone wants, I will give him a simple one-line patch to disable this option, in fact.)

The bottom line concerning journal entries is this: Is it fair to criticize a program because of an *extra* feature which most other systems do not have? If someone does not personally like this option, he need not use it. But, since we are dealing with the real world of thousands of users entering millions of journal entries each year, you must surely agree that there are, indeed, instances when this feature can truly save hours of frustration.

Again, I do think the review was generally fair and balanced. I just wanted to share with you some of the reasoning behind the specific areas upon which you had criticism. True, VERSALEDGER II is lower priced than virtually all other general ledgers available. However, the reasons why the program was designed as it is was absolutely *not* due to economy reasons, but rather due to a desire to make the program more versatile for a wide variety of users.

A TIP FOR TRSDOS 2.0 USERS

Robert J. McCurdy
501 Wayne Drive
Cinnaminson, NJ 08077

Here's a patch for you TRSDOS users who wanted an underline cursor.

```
PAUSE Provides blinking underline cursor.
PATCH SYSRES/SYS A = 06B1 F = 65 C = 69
PATCH SYSRES/SYS A = 06B1 F = 65 C = 69

PAUSE Provides no-blink underline cursor
PATCH SYSRES/SYS A=066E F=65 C=09
PATCH SYSRES/SYS A=06B1 F=65 C=09
```

A COUNTER REVIEW OF CP/M VARIANTS

R. David Otten, President
SIGNATURE Software Systems, Inc
5602 Stouder Place NW
Pickerington, Ohio 43147

The review published in the January/February issue of *two/sixteen* (+ twelve) magazine, "Review of CP/M Variants: Lifeboat, Pickles & Trout, And Aton," by Robert P. VanNatta (Vol. 1, No. 4, pp. 3-9) was most valuable, in that it did something not done before: it reviewed the remaining implementations of CP/M for the Radio Shack Models II, 16 and 12. However, enough "bugs" ppeared in the review that I feel additional comment is necessary.

CP/M Is An Operating System

The actual meaning of the letters "CP/M", according to Digital Research publications, is "Control Program/Microcomputer", and not "Control Program/Module", as described in the review.

CP/M Must Be Customized To Each Computer

We wish to emphasize here that the implementations of CP/M on most microcomputers are supported by the manufacturer of the computer. With Tandy, this has not been the case. However, with the advent of CP/M 3.0 or CP/M Plus for the Models II and 12 from Tandy, this is no longer the case.

The "System" to which Mr. Otten refers hroughout this article is SIGNATURE Software's Community Pharmacy Prescription Profiling System. His "users," about whom he speaks throughout, are independent pharmacists located around the United States.

BIOS Gives Computers Their Personality

While Tandy apparently did not offer direct aid to the various vendors who have implemented CP/M on their machines, they have offered aid in the form of some of the best technical information manuals available. Also, Tandy did offer direct aid to the four major CP/M implementors when problems with the disk controller chip were identified and solved. We agree that the different versions of CP/M implementations on the Model II/16 have different "personalities"; however, the same can be said when running different versions of TRSDOS from Radio Shack, which is one of the reasons we switched to CP/M early in the game.

If the implementation of the BIOS is done by or through the computer's manufacturer, the source code for the BIOS is usually supplied to the user. The independent implementors of CP/M for the II/16/12 computers do not supply the BIOS source, which is, frankly, a problem. Since Radio Shack does not supply the source on TRSDOS either, most of us have accepted this as a way of life, but it should be noted that we have the sources for our other computers, including the IBM PC, the Vector-Graphic and the Teletek.

The Versions

We have also used a number of versions of CP/M, including two not mentioned in the review, Cybernetics and FMG. The first version of Lifeboat to which we had access was 2.1, which did not always boot properly, if at all. Version 2.3 was a marked improvement; however, if you looked at MOVCPM.COM, using DDT.COM, you found that the imbedded version number was *not* 2.2, or greater. Oh, well. We also had the opportunity to review an early version of the P&T implementation, which was most impressive with its speed, but less impressive with its formatting of disks, without error checking. We picked Lifeboat's implementation over the others because we liked the CONFIG.COM program which allowed us to easily change printer and serial port parameters, and because the FORMAT and COPY (Backup) utilities supplied by Lifeboat let the user know something was happening while the function was executing. Shortly after our applications package was placed on the market, Lifeboat's 2.24a version became available, so most of our experience lies with that version. We should note at this point that at least two important features were handled differently by different Lifeboat versions, and by other implementations, namely the *screen memory* and *direct memory access* (DMA) features.

As you may know, Radio Shack stores the 2K of video RAM on the video board, which can be bank selected in or out by the operating system, leaving the "user" RAM in the computer free for the user's use. Some of the Lifeboat implementations did not use this feature, using "user" RAM to store the video display, which meant, in turn, that the user had 2K less RAM available for his programs. It did, however, make it very easy for the user to access the "video" RAM from BASIC and other high level languages, which meant that "screen dumps" to a printer required no assembly language programming expertise. By not using the RAM on the video board, the Lifeboat implementation did not have to worry about banking the board's RAM in and out to maintain the display. The 2.25 versions of Lifeboat do use the video board's RAM.

The DMA feature was a different story. For some reason, some of the early versions of Lifeboat used it, others used it only on the A drive and finally, on all drives. DMA (Direct Memory Access) merely allows the moving of data directly from the disk to the computer's RAM, bypassing the microprocessor completely. DMA is usually much faster than reading the data through the port with the microprocessor and putting it into the desired RAM area. To get a comparison of this feature, read a disk file from an expansion drive, using Lifeboat 2.24a, and then read the same file from the A drive. The speed of the DMA read is about four times faster than the non-DMA read from the expansion drive. Doing the same thing with a CP/M version which uses DMA on all drives will produce about equal results. Note that we use a variety of 8" drives on our IIs and 16s, namely, CDC drives (what Radio Shack used to use), both single and double sided, plus Shugart 801s and Tandon double sided drives.

Lifeboat Is Vanilla?

Our jaded view is that "Lifeboat used to be vanilla". Lifeboat has exhibited some very undesirable traits over the past year and a half, namely very slow response to orders for updates. Further, they have never responded to "bug reports", at least as far as we are concerned. We had one of our users request an update from 2.25a to 2.25d recently, just to see what would happen. Six weeks later, the user received the two-byte patch in the mail, along with his check. This is better than our last update, but certainly not satisfactory, since our user had just purchased the new Radio Shack expansion drives which will not work with 2.25a. On our last update, we sent our check and disk, and three telephone calls and four weeks later, received our update, COD! We called Lifeboat, and they said to go ahead and take the COD, and they would give us a refund. That was about eighteen months ago, and we are still waiting.

What's An I/O Byte?

In addition to the comments regarding the I/O Byte in the review, we would also like to point out that the I/O Byte (address 0003) can also be modified from a high-level language, and therefore, from an applications program. This means that with the use of PEEKs and POKEs, or their equivalents, you could re-direct output from the screen to the printer, the printer to the screen, and/or to the serial ports. This, in real life, means you could be running two printers, and have the program select which to access through which port, which is easy to do if you have a serial and a parallel printer or two serial printers. Try that from TRSDOS 1.2!

Regarding the programs furnished by Lifeboat, P&T, and ATON for configuring the serial ports, the printers, etc., we really do not appreciate a good thing enough. Most implementations of CP/M on other machines do not include a nifty program, such as CONFIG or SYSDEF for doing these things. You have to patch and reassemble the BIOS source codes.

Drive Support

This area of Mr. VanNatta's review gave us the most problems. How could a person using a computer system with disk drives choose the operating system with the weakest disk support?

All three versions support the time-honored single density format, which makes your Model II/12/16 compatible with a huge number of other computers.

Perhaps at this point we should expand a bit upon the subject of disk formats. When a disk is formatted, "tracks" are recorded, which contain "sectors", and the sectors contain identification information (ID Bytes), which tell the disk controller which track and sector is being read or written, as well as the type of format (single or double density) that was used when the disk was formatted. The good, old-fashioned, CP/M single density format says that there are 77 tracks, with 26 sectors in each track, on an 8" disk, and that each sector is 128 bytes long (in data). CP/M then handles directory information and all files on this basis, putting the information on the disk and in the directory in "groups" or "blocks" of "sectors". A single directory entry (not including extents or extensions) will handle 128 "records" or "sectors", divided into "blocks" or "groups" of 1K (1024 bytes or 8 sectors). Block 0 (zero) on a disk is the beginning of the directory, which is on the third track, which is track 2 (0, 1, 2 — remember, computers and programmers count zero as a number).

Formatting a disk in other than single density is done by changing the sector size and the block size. For example, Radio Shack used to use 26 sectors of 256 bytes each in their double density format. P&T uses 512 byte sectors, but puts fewer sectors on a track, and ATON and Lifeboat use 1024 sectors, again with fewer sectors per track.

Early on, we said that one of the things on each sector, besides data, was called the "ID Bytes". These ID Bytes must appear on every sector, so if you have fewer sectors, you have fewer ID Bytes, and therefore, more space for data bytes.

Next, since CP/M handles data as 128 byte records (sectors), some "block" manipulation must be done to allow the directory entries to find the desired record, plus keep the space required for and the format of the directory acceptable to CP/M. This is done by changing the amount of space controlled by each block entry in the directory. That is why, when you STAT a bunch of files on a disk using CP/M, if the disk is single density, you see the space taken by a short file as 1K, while the same file on a double density disk takes 2K, and on hard disk 4K, 8K or 16K, depending upon your hard disk initialization choices.

What all of this is leading into is simply this. Lifeboat CP/M will allow you to read and write single density, double density (a la early Radio Shack), and extended density. The system disk must be formatted in extended density. Single density gives you about 241K per disk, double density 481K per disk and extended density 596K per disk. P&T CP/M allows you the 241K single density and the 596K extended or double density, and ATON gives you all of the above, plus it is able to read and write P&T formatted double density disks (but not format them that way). A separate version of P&T CP/M adds the ability to read and write double sided double density disks, and while this version is also compatible with their single sided version, their single sided version is not compatible with their double sided version. This, in our opinion, is one of the major advantages of the ATON implementation, since you could have as wild a set-up as follows, and function with ATON, but with none of the others.

For example, picture a Model II (single sided disk built in) with two additional single sided drives and one double sided drive added on. You could put your ATON single sided disk in drive A, a double density Lifeboat disk in B, a double (extended) density P&T disk in C, and a double sided double density ATON disk in D and read and write all of them, with no problems at all. The only thing you cannot do is read and write a P&T double sided disk. Or you can put a single sided disk in the double sided drive, again with no problems.

Reliability

One of the areas of great importance, under the heading *Drive support* in Mr. VanNatta's review, was omitted — reliability. From about the time Lifeboat came out with version 2.23 on, it was generally recognized that for whatever reason, P&T's implementation had a slight edge in reliability on the Radio Shack machines because it generated fewer "BDOS Errors" (disk I/O errors). However, Lifeboat's version was more "user friendly", and with their version 2.24a, the reliability gap narrowed. When Radio Shack identified and solved the disk controller problem, and notified the implementors of CP/M of the software method to solve the problem, P&T implemented the change immediately, with Lifeboat dragging a bit, but finally coming up with version 2.25a.

Unfortunately, Lifeboat did a few other modifications, such as implementing the video board RAM, messing up their printer driver, fixing the BIOS so it did not know which side of the disk it should read, and so on, which made version 2.25a a version not to be recommended. This became even more apparent when Radio Shack switched brands of drives in the expansion bay. Lifeboat 2.25a couldn't read them. Our contact with Lifeboat at that time indicated that they were about to jump away from CP/M and onto the IBM PC, MS-DOS bandwagon, and really had little intention of supporting CP/M on the Radio Shack machines in the future. They also did not seem inclined to do anything about the Radio Shack hard disk, either, and unlike the previous reviewer, we were anxious to have CP/M on Radio Shack hard disks simply because our users would have someone down the street to go to with problems, instead of shipping a drive to California or wherever.

We then ordered P&T CP/M and began to move our files and programs to their format from Lifeboat, by first going to single density and then to their double density. This, by the way, is why we dropped TRSDOS (or at least one of the reasons). We had spent a great deal of time moving files from 1.1 to 1.2, and when 2.0 came out, and was also not compatible, we said, "Okay, if we've got to do it again, let's move to CP/M." Which we did. Now, here we were again . . . . To give ourselves a second source, we also ordered ATON CP/M, plus an extra memory board for the Model II, just to see what would happen.

When we discovered that ATON CP/M would read both Lifeboat and P&T disks, we suddenly became very interested in ATON. We had just gone through a gigantic problem with moving one of our users from floppies to a Cameo/P&T hard disk combination, by having to transfer Lifeboat files to P&T single density, etc., and here was a CP/M that could read and write both! We then began haunting both P&T and ATON for a Radio Shack hard disk implementation, and ATON beat P&T by about two weeks. Since we don't collect hard disks (we do have nine computers), we have stuck with ATON on the hard disk. By the way, they have also come out with the 12 meg version, which is also compatible with the 8 meg drive, while we are still waiting for P&T's 12 meg.

After the hard disk excitement died down, we realized we had not received our memory board for the Model II, which finally did show up. We installed it, and wow! But more on this later.

Error Trapping

Lifeboat really does not trap errors, as noted in the review. P&T does, and gives you an error code, which you must look up in the manual. ATON gives you a highlighted error message, and waits for you to solve the problem. If you correct the problem, you merely hit the <ENTER> key and continue. If you cannot solve the problem, you hit the <BREAK> key and exit to

CP/M. The problem can be (but seldom is) disk related, or printer related (no paper, broken ribbon, etc.), etc. Since Lifeboat will let you change disks during the "error pause", you can make a mess of things, and with P&T, I don't like having to look up error messages in a manual. Score one for ATON here.

One Drive Support

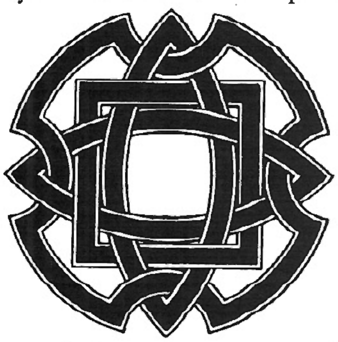
With 8" disk drives now available for less than $200, having only one drive is ludicrous. We can understand not wanting to pay Radio Shack $1,150 for an add-on drive. Fine. Buy some $200 drives, build or buy some power supplies, and put an extra drive on your system for an amount less than a six months' supply of cigarettes costs.

The hard disk utility by ATON, is called FILEX.COM, and you do not need a hard disk to use it, and yes, it does work, which makes it a bargain (free), since something similar is advertised elsewhere for $150. ATON, I believe, is now including FILEX.COM with non-hard disk systems. FILEX offers a number of advantages over PIP.COM, which won't copy a file longer than 512K, and does *not* do an actual verify of data. PIP merely verifies that it can read what it wrote, not that what it wrote is the same as what it read. FILEX compares what it read into memory with what it wrote. If you really want to compare, buy Z80 FORTH from Laboratory Microsystems ($50—a real bargain!) and use FCOPY.COM. That program reads and writes a file, then re-reads the source and destination to do a byte by byte compare. We digress.

FILEX will let you read a full hard disk onto floppies; it tells you when the floppy is full, and when to change disks, which it then numbers internally to keep track of the mess and to make sure you put the thing back together properly, if you ever need to restore your back-ups to the hard disk.

There are a couple of tricks to using FILEX, of which you should be aware. First, it apparently keeps track of things on the basis of "block" size, so you should not, for example, use FILEX as a substitute for PIP when copying files from the hard disk to a floppy, for use on the floppy. For the same reason, use caution when dumpng files from the hard disk to floppies, then re-initializing the hard disk. ATON allows you to change the block size when initializing a hard disk, so you could have files on the hard disk handled in 4K blocks, save them to floppies using FILEX, re-initialize the hard disk to track files in 8K blocks, read the files back to the hard disk (using FILEX), and wonder what happened. We are not sure this is a bug; just don't try to copy "apples" from the disk, and then try to put "oranges" back.

FILEX gives you a way to copy files longer than a single floppy to floppies for backup purposes, but you still have a backup problem, and that is speed. Any backup to floppies from hard disks is slow compared to the hard disk. It's still less expensive than tape or hard disk to hard disk backups, but you will have to determine what you need as far as speed. For example, a 1.2 meg file, filling one double sided floppy, will take about 18 minutes with verify, or 12 minutes without verify. From the floppy back to the hard disk takes about 9 minutes. A tape backup system will pull a 10 meg hard disk in about 9 minutes.



By the way, with ATON, we have run with four logical hard disks (one drive thinking it's four drives), plus 4 floppy disks, two double sided and two single sided. It's fun to read or write to drive H and G, and so on!

Terminal Features

Lifeboat's support of the control codes for the Lear-Siegler Terminals is useful, again because of the wide use of that particular terminal. The P&T version seems close to the ADDS 25, which is also a popular terminal. The main advantage of the SOROC terminal emulation by ATON (and by P&T in some applications) is the ability to highlight text when using applications, such as WordStar. ATON's implementation of the "bell" code will be useful on the Model 12, since Radio Shack put a bell on that machine. We wish they had done the same on the II and 16. If you don't like the "blinking" of the screen with ATON when a "bell code" is encountered, then you can turn it off by following the instructions in the manual and/or in the SYSDEF program furnished by ATON. The formatting of characters into rows and columns on the screen (in this case, 80 columns and 24 or 25 rows) is a function of the machine (not necessarily the hardware, since the contents of an EPROM can be the culprit), not of ATON "squeezing the regular 24 lines (rows) together". The 25th line has always been there; no one has bothered using it until Chuck Wilde at ATON did so. We like it. If you don't, you can turn it off with the ATON SYSDEF program.

You can also patch WordStar to use the video board under ATON if you follow the directions in the WordStar Installation Manual.

The Keyboard

Mr. VanNatta's comments on the ATON implementation of the keyboard table seemed misleading. First, "finding the (keyboard) jump table" is not "too hard", since the whole thing is part of the menu-driven functions of the SYSDEF program included with ATON's implementation. Also, the subject is very well documented in the manual. In any case, the value of the table is fantastic for our users in the field, in that they can move commonly used alpha character functions (Y and N replies to prompts wanting a "yes" or "no" answer) over to some of the keys near the numeric keypad, if most of the surrounding data input was numeric. They can also re-define the characters generated by the arrow keys, if they prefer to use the arrow keys with VisiCalc, SuperCalc, WordStar, WordMaster, etc. In other words, this is a valuable feature! The primary ATON "revision" was to define the backspace key as a DELete, or rubout, which it indeed should be. However, those of us using Radio Shack machines are used to the backspace key generating a control "H", so simply change the definition from "08 7F" to "08 08". The whole task, from A> prompt to A> prompt takes about 30 seconds.

Documentation

For once, we agree! As to the comments on CP/M 3.0, P&T has already publicly announced that they will not implement 3.0, and Chuck Wilde indicated as much to me the other day on the telephone. The reason given by P&T is the same reason as might be given for not finding MP/M implemented on the Model II/16. CP/M 3.0 and MP/M require that a portion of RAM at the top (of 64K) remain constant, while the portion at the bottom of RAM (or below the constant) be "bank

selectable". On the Radio Shack machines, the lower RAM is constant and the upper RAM can be switched, just the opposite of what is needed (anyone looked at the circuit diagrams of the Model 12 lately?). With ATON, who already knows how to handle this bank switching situation, the advantages to be gained by going on from ATON LEVEL2 to CP/M 3.0 may not be great enough to justify the cost, both in development and to the user.

Extra Utilities

We agree fully with the comments regarding P&T's RESIZER substitute for SYSGEN and MOVCPM; we have complained to P&T on the subject. It's enough of a problem to get someone to backup a disk, but to then require him to dig out the master disk and run RESIZER to put a system on the backup is silly (we like to have our users re-format disks before backing up). As noted, the Lifeboat COPY program will back up a P&T disk, including the system. So will ATON's COPY program.

We consider the utilities supplied by both ATON and Lifeboat to be more "user friendly" (easier to use, if you don't read documentation) than those furnished with P&T. For example, the P&T FORMAT program does not verify the format. Separate programs are furnished to accomplish that function (two steps, again!). On the other hand, the Lifeboat COPY program claims to "verify", but in practice, does not really do so. There is no compare of the data written with the source data, either in memory or from the source disk. The "verify" merely indicates the data written can be read — not that it bears any similarity to the source data. This has caused us some very strange problems. As mentioned in the previous review, ATON does not do a selective copy, but it is also faster, so...

WordStar

two/sixteen's reference at the end of the review to turn to page 38 just was not enough, after the remarks regarding ATON CP/M and WordStar. The video board RAM reference is true enough, but not a big hassle. We have used both, and have already pointed out that the situation can be changed, if you wish. And doing so is no more hassle than implementing many serial printers to WordStar/Radio Shack/Lifeboat combinations.

We use parallel printers, including the Daisy Wheel II, partially because we can get service (which we have not needed), partially because we know enough to cut lines 26 and 28 in the printer cable so have no problems with re-booting, BDOS errors, and other problems when the printers are turned on and off while the computer is in use, and partially because our feeling about the RS-232C "standard" is about the same as our feelings about "standard" 5-1/4" disks. Our users like to buy hardware and software, plug the plugs in, turn the thing on, and have it work. Try that with a serial printer on a Radio Shack machine, or on a great many other micros. Some people spend nearly a year getting a particular combination working.

The comment regarding the keyboard buffer is also true when running ATON CP/M Level I. With all of its features, there is no room for the buffer. However, install LEVEL2, and you'll have all of the buffer you want!

Since the reviewer says so, we will assume WordStar works fine on 2.25d; it does not on 2.25a, and is a little sloppy on 2.24a. Until ATON, we preferred P&T with WordStar.

We do all of our documentation (usually more than 100 pages each), and have written a book (well over 200 pages, single spaced), all on WordStar, all using ATON CP/M. No problems. With earlier versions of Lifeboat (prior to 2.25c) we had to learn how to recover WordStar files from RAM, since "BDOS Errors" and "outer space" occasionally caused problems. Version 2.25c does not like some block moves with WordStar, hanging the system up forever with certain moves. We cannot comment on 2.25d, and why bother, they only changed one byte (the other was to change the letter "c" to "d" for the version number).

ATON & LEVEL2

Unlike Mr. VanNatta, we purchased the 64K RAM board for the Model II when Radio Shack lowered the price to $399.00. We also purchased ATON LEVEL2, and again, Wow!

Before changing the jumpers, we opened our machine, unplugged the RAM board already installed, and compared it to the new board. There were some minor changes, but the new board was obviously configured in the same way, so the new board was plugged into the same slot as used by the old board, and the system booted with TRSDOS. Memory tests were run, and all seemed fine, so we unplugged the new board, replaced the

CLASSIFIED

old board and re-jumpered the new board according to ATON specs. We plugged the board into one of the empty slots, pulled the interlock on the machine (we have an old one), and booted using ATON CP/M. At the A> prompt, we typed "LEVEL2", hit <ENTER>, and the system loaded a program, then displayed a memory map showing what memory banks were active. A new kind of prompt, A}, also appeared which is used by ATON to indicate that LEVEL2 is active.

Since we do not recommend new versions or features to our users until we have thoroughly used the product or update ourselves, testing was then started. A programmer can used banked memory a number of ways, but since our users are nationwide, and since our applications programs run on nearly any CP/M based machine, all we were interested in was the direct effect of the additional memory on the use of our applications on the Radio Shack machines, without modifying any of our software.

One of the most notable differences in this case was speed. If the user does not specifically reserve any of the additional banks of RAM, then ATON uses the whole thing for buffers and for their CP/M BIOS, which gives you the first major advantage of LEVEL2 on the Radio Shack machines using CP/M. The various implementations of CP/M on Radio Shack generally leave the user with a "TPA" (transient program area) of between 48K and about 54K-56K. TPA is the amount of RAM available to the user as program and data space, so, for example, if you load in MicroSoft BASIC version have 23K to about 32K of RAM available after BASIC is in memory. With LEVEL2 running, you will have a TPA of 62K, which means about 39K after BASIC is in. You will also have a full page of RAM (256 bytes) in high RAM. All this extra space occurs because ATON banks their BIOS onto the new RAM, giving you the space that used to be taken by the BIOS.

As we mentioned, the balance of the new RAM is then used for I/O buffers, unless otherwise specified by the user. ATON puts all of the disk directories into the new RAM, so no disk accesses of the directory are required, after each on-line drive is accessed the first time. This also speeds things up a bit.

While 64K does not really represent that much additional RAM, when you are talking about a "ram disk" application, you can see some of the advantages and effects of a "ram disk", especially when running text editors, such as WordMaster and WordStar on LEVEL2. It's just plain fast. However, these features alone do not mean too much to the users of our applications programs, so additional testing was necessary. For example, some of our compiled programs run to 50K in length, which on floppy disks, take some time to load into memory.

Which brings up another feature of ATON CP/M. While Lifeboat will allow you to specify the stepping speed to be used on your drives, the speed is fixed for all drives on line. P&T will allow you to specify the speed of each individual on-line drive, and ATON CP/M automatically sets the speed of each drive to the fastest it can be for error-free use. For example, the built-in Shugart on our system really wants 15ms, while the CDCs seem to handle at least 10ms well (which is backwards to what you usually hear). With ATON, the speed of one might be 6ms, another 10ms and so on, all automatically. This in itself speeds things up.

To continue our testing, we checked the amount of time it took to load a number of our applications using different implementations of CP/M. Most of the timings were done on a Model II, although the hard disk times came from a Model 16 (without LEVEL2—more about that later). Here are the results:

Test	2.24a	2.25c	P&T	ATON	ATONLV2	ATONHD
A	13	12	11	8	6.5	4
B	16	19	17	14	7	5
C	16	20	18	15	10	6
D	12	16	14	13	8	6
E	23	24	21	21	11.5	7
TTL	80	91	81	71	43	28

Lifeboat = 2.24a and 2.25c P&T = Pickles & Trout

ATON = "plain" ATON ATONLV2 = ATON LEVEL2 with 64K added RAM

ATONHD = "plain" ATON on RS 8 meg hard disk and Model 16

A	=	program P from RMENU
B	=	program NR from P
C	=	program RR from NR
D	=	NR to RMENU
E	=	Exit to GMENU

Note that a number of files are read and written on entrance and exit from and to these programs.

As you can see, using LEVEL2 with floppy drives causes the programs to load at a speed approaching the hard disk implementation, which now requires a little more comment.

Radio Shack supplies the hard disks with an additional 16K of RAM, which ATON puts to use in a similar manner, as extra RAM is handled on LEVEL2. As a result, the user has a 62K TPA again, and some buffering of I/O.

LEVEL2 will be implemented on the 16 and on the hard disk versions by the time you read this, and we frankly expect fantastic things, since our Model 16 already has 256K. We are also adding another 64K to our Model II.

The above timings were sent to our users in our user's newsletter, and several have opted to go to LEVEL2, others to the hard disk.

LEVEL2 also speeds up the printing of forms, such as monthly statements, labels, etc., again since it buffers the I/O.

Pickles & Trout

We have not had the problem described regarding terminal emulation when using P&T and have had no more difficulty installing various programs on their implementation than on Lifeboat and on ATON. In truth, any of these Radio Shack implementations offers features and ease of use that make other CP/M implementations on many other machines look very primitive.

As to standardization, we agree, but frankly find more problems with non-standardization of double density, of 5-1/4" drives, and of serial port utilization than with terminal emulation. Our programs use functions to allow the user to specify his terminal's control characters and cursor addressing combinations, and, to date, we have found none that we could not handle as desired.

The Problem with Printers

The problem of linefeeds following carriage returns is not exclusively Radio Shack's, and, in fact, does not actually apply to most of the newer Shack printers. As to implementing software on the various versions, with their printer drivers and Shack printers, we have had the most problems with Lifeboat's 2.25a and 2.25c, and no problems with P&T and ATON.

We must agree with the comments regarding the 512 byte sectors used by P&T. We also wonder at their handling of the double sided drives, since they apparently do not handle the two sides as "cylinders" but as separate sides. This is one of the shining advantages of ATON — its ability to handle both P&T and Lifeboat data files (with the single exception of the double sided P&T format).

WORDSTAR USER'S TIP

Robert B. Steinbach
CREATIVE RESOURCES
816 Pebble Lake Court
Sunnyvale, CA 94086

I expect that most WordStar users are already aware that the backspace key works as marked.

However, did you know that the ^B command for reforming paragraphs is built into the <F2> function key on the Model II/12/16 for single key-stroke use?

Also, did you know that the <F1> function key can be used instead of ^A for cursor movement to word-left?

And, did you know that the "down arrow" key works as a DEL for deletion of characters to the left?

Conclusions

After using Lifeboat CP/M for several years, we no longer recommend this implementation to our users. Of the three versions, BDOS (disk I/O) errors seem to occur with considerably greater frequency with Lifeboat than with the other versions. They apparently have not kept up, since at one time they were as reliable, if not more reliable, than other versions. Further, Lifeboat seems to have lost interest in CP/M, and has made no attempt to implement the Radio Shack hard disk or double sided drives.

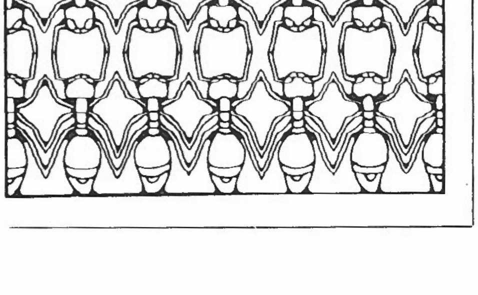
Choosing between P&T and ATON is a bit of a toss-up. P&T has been around longer and is reliable. ATON offers many more "bells and whistles" and also seems reliable. Both companies are very cooperative and easy to deal with and seem to be very interested in their products and our (your) needs.

If you do not want to add the extra memory board to a Model II or 12, or do not care to use the extra memory in a 16, and do not plan on a hard disk, P&T would be a logical choice, since it does offer the keyboard buffer in its standard version, and ATON does not. If you can afford the extra RAM board for the Model II or 12, then ATON is definitely the way to go, if you have any interest in speed.

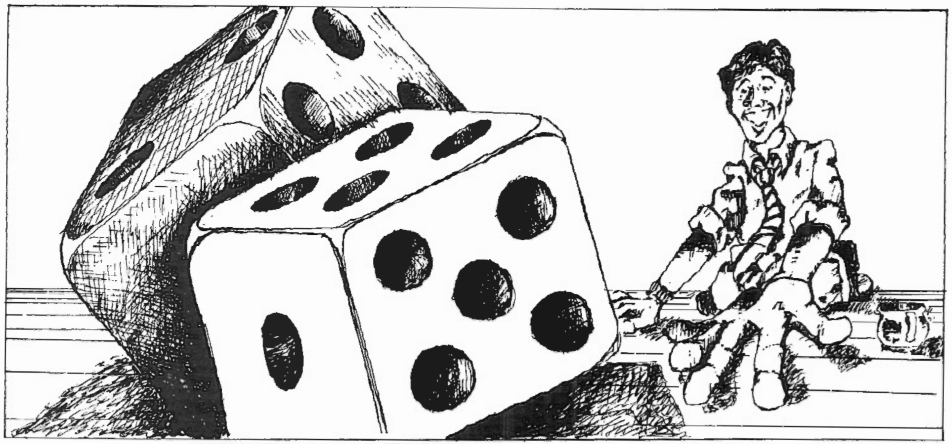
Another example not mentioned earlier: we use PLINK (Phoenix Software) to link long programs that L80 from MicroSoft cannot handle (out of memory). On a Model II with Lifeboat 2.24a, some of our programs take 45 minutes to link, on 2.25c, about 37 minutes, on ATON with the hard disk just 9 minutes, and on the floppies with LEVEL2, just 6 minutes.

As to the hard disk implementations, our users using Cameo drives with P&T report complete satisfaction, as do our users with Radio Shack drives and ATON. We wish we could say the same for users not running Radio Shack hardware, but we cannot. So far, we have not experienced unpleasant surprises with either P&T or ATON, and if we do, we suspect that either will be of immediate help in solving a problem.

In closing, we switched to CP/M from TRSDOS for several reasons, some of which we have already mentioned. At the time we made the switch, we needed a BASIC compiler, and Radio Shack was apparently heading toward the RM BASIC version, with which we were not compatible. The MicroSoft Compiler was available on CP/M. There is a fantastic library of "public domain" *quality* software available to the CP/M user, and we were interested in that software, and even the earlier implementations of CP/M on the Model II seemed more reliable than TRSDOS, especially in handling files. There are, however, some hassles associated with using a non-manufacturer operating system. These include the drive (hardware) changes made by Radio Shack, the availability of ARCNET and XENIX, and so on. And TRSDOS seems to have become quite reliable, and with some of the SNAPP add-ons, probably pretty useful, but for our immediate purposes, we have no regrets in our choice of CP/M.



Review of



Coffee Break Casino

Arnold Fischthal
Advanced Data Design, Inc
184-08 Tudor Road
Jamaica Estates, NY 11432

For those of you who would like to do something with your business computer during your coffee break and for those gamblers among you who can't make it to the casino this weekend, have I got a software review for you! COFFEE BREAK CASINO.

With this system of programs you can actually be at your choice of six different casino games without leaving your office. You can play

Slot Machine
Roulette
Baccarat
Wheel of Fortune
Keno
Craps

All of these can be played by more than one person at the same time, except for slot machine.

The COFFEE BREAK CASINO disk comes with a well-written, very complete manual describing in detail not only how to play these games on the computer but also the rules for betting if you haven't played any of these games before. For those of you who are used to manuals describing everything except what you need to know when the chips are down (ha, ha), this manual will be a pleasant change.

When the disk is booted up, you will enter a menu-driven system allowing you your choice of game. The system is well error-proofed in that it will only accept those keys having any meaning to the program. When entered, each game has a "help" key which will display on the screen all the legal commands and their meanings for that game. The screen displays for each game are exceptionally well formatted to allow as much as possible on the screen without making the information appear to run together.

As you play each game, you notice another very nice achievement. The games use extensively the limited graphics capabilities of the Model II, and they use them quite well. In Slot Machine, for example, you see three windows with the various objects "spinning." In Wheel of Fortune, you will see a portion of the wheel as it is turning and then begins to slow down. A similar effect is produced on the roulette wheel. There you see a portion of the wheel "spinning" one way and the ball going in the opposite direction.

Unfortunately, since the Model II was not designed for game playing, its moving graphics capabilities don't match those found in other microcomputers. There is an annoying flicker at times in the screen. Although it was obvious the author tried to eliminate this in techniques he used in the software, he could not quite get rid of it all.

Also, for those of you who cannot find any companions to play these games with you, you will have to settle for playing with Slot Machine or for making up imaginary friends and entering them into the gaming situations.

The games follow for the most part "American" casino game rules. Each game will accept and account for all wagering situations, and each game keeps track of every player's winnings or losings; this data can be stored for later continuation of play.

The documentation, error-trapping, and ease of use of this package all rate a 10.

COFFEE BREAK CASINO is available from
Coffee Break Software, P. O. Box 8495,
San Francisco International Airport
San Francisco, CA 94128 415/595-4400
Cost: $39.95

Barbara's Adventures *in* COMPUTERLAND

Barbara S. Albert

This installment of "Barbara's Adventures in Computerland" addresses a topic that should have been the subject of our very first Novice Column. My first venture into Computerland took an in-depth look at the procedure for backing up diskettes and the importance thereof. But before there is a need for backing up disks, there is a need to FORMAT disks, and that is what we're going to look at this time.

The reason I overlooked writing about such an essential topic previously is elementary: I didn't know such a function existed. I thought the disks given to me for doing word processing tasks were straight out of the box they came in except that somehow the operating system got on them, as indicated by some little hieroglyphic-like notes Dick Young made on each one that said usually "CP/M 2.2ed". He writes so small it was very easy to overlook his scrawl and proceed blithely onward.

But time has a way of impressing a few key concepts into even the most oblivious consciousness, and I gradually became aware of the word FORMAT being used by more than just a few people. What was this strange and wonderful process and what did it have to do with me?

On to FORMATTING

We know that the computer's memory is volatile, and if we don't provide a means to preserve the contents of this memory, we'll lose everything when the power is turned off. That's where diskettes come in. A computer diskette is "a flat circular plate with a recording surface of magnetic material on which data can be stored by selective magnetization of the material. Magnetic disks fall into two major categories: hard disks and flexible disks or 'floppies'."

If, however, a diskette is not configured in a precise way, compatible with and acceptable to the DOS (Disk Operating System) of the computer you are using, the contents of your computer's memory will not be "received" by the diskette. And that's where FORMATTING comes in. The simplest definition of FORMAT I found, and one that represents a good place to start, is: "To Format a disk is to prepare that disk to receive data."

When you purchase a floppy disk, its surface is a non-magnetized blank. You can't write on it, and there's nothing to read from it. To make the disk into something that can be used by your computer, you must FORMAT it.

During FORMATTING, the disk drive (as instructed by the DOS) organizes a diskette into working areas called tracks and sectors which are defined and labeled by codes recorded on the magnetic surface.

The way in which a DOS organizes a disk into tracks and sectors is via the FORMAT utility included on the Master Distribution disk of TRSDOS or CP/M or whatever. For those of you with two or more drives to FORMAT a disk, you boot up the Master in Drive 0 (or A), insert the disk to be FORMATTED into Drive 1 (or B) (or whichever), and at the system prompt A> type in "FORMAT." For those of you with only one drive, it's also very easy to FORMAT. Once you have loaded the FORMAT program with the Master disk, remove it from the drive and insert the disk to be FORMATTED.

There's more to FORMATTING than just making a disk your computer can use to read data from and write data to. FORMATTING is also an error detecting process; the FORMAT utility looks for flaws in the recording surface on new diskettes, areas which would not be able to store data, and then "locks out" these areas so they will never be used. The TRSDOS Manual recommends that the FULL verify option be used when FORMATTING a new diskette to give protection against lost data due to a flawed track.

Also, after a period of time, flaws may develop on a disk's surface. The average life of a floppy disk depends on the quality of the disk used, whether the disk drive is in good repair, and whether the work environment is relatively dust and static free — two elements that will cause problems with data storage accuracy. In an article entitled "Floppy Disks Add Versatility" published in the October 1981 issue of *Radio-Electronics*, the following fact was discovered: "The magnetic disk surface (of a floppy) has a life expectancy of 10 to the 6th power loaded passes." So, after your 1,000,000th loaded pass, you should use the FULL verify option when RE-FORMATTING to lock out any flawed tracks.

The FORMATTING process can also be used to erase a disk. Thus far FORMATTING has only been discussed in terms of a new, never-before-used disk, but a used disk can be RE-FORMATTED. When you RE-FORMAT a disk, all the data it contains at the time is lost. You can RE-FORMAT a TRSDOS disk with a CP/M master to create a CP/M disk that will contain not a trace of its former existence.

The Pickles + Trout FORMAT utility is slightly different from the TRSDOS utility. TRSDOS initializes a disk by defining the tracks and sectors to receive data and by writing system information onto the diskette. The P+T CP/M Manual explains that their FORMAT utility writes over any information that is on the diskette "leaving the diskette with all sectors filled with bytes of E5 hex which is interpreted as an empty disk by P+T CP/M." During P+T FORMATTING, the system tracks are written over as well as the data tracks, and if you want a system on your diskette, *the system must be re-written to the system tracks after FORMATTING.*

Tracks and Sectors

So what are these tracks and sectors that FORMATTING so very nicely sets up for us?

Tracks are concentric divisions on the surface of a disk, the area covered by a stationary read/write head with one revolution of the disk.

A sector is an arbitrary division of a track, which in turn contains an arbitrary number of bytes.

Eight inch disks always have 77 tracks on one side, but the number of sectors per track and number of bytes per sector varies from one operating system to another.

In TRSDOS, Track 0 is reserved by the System. Another track, usually 44, is reserved for the directory. In P+T CP/M, the first two tracks (0 and 1) are the system tracks; the directory takes up part of Track 2. How much space it takes depends on the number of directory entries allowed in the system (according to Gary Kildall from Digital Research, the first 16 sectors on Track 2 contain the directory, providing space for 64 file directory entries.)

A single-sided, **single-density** 8″ diskette contains 77 tracks. Each track contains 26 sectors. Each sector is 128 bytes long. The total amount of disk surface available for file data on a single-sided, **single-density** 8″disk is about 241K.

As you would imagine, FORMATTING a disk using the P+T CP/M **double-density** option doubles the amount of data that can be stored in each sector. Each single-sided, double-density 8″ diskette contains 77 tracks, the same configuration as single-density. BUT each track contains 16 sectors, and each sector holds 512 bytes (except Track 0 which has 128). So the total amount of disk surface available for file data on a standard S/S, D/D 8″ disk is more than twice as much as single-density!

And, incidentally, a **double-sided, double-density** P+T CP/M disk contains 1210K bytes of storage space!

TRSDOS FORMATS in only double density. Each TRSDOS disk contains 77 tracks, each track contains 26 sectors, and each sector contains 256 bytes, except for track 0 sectors which contain 128 bytes.

Any Questions?

There you have it—a look at a procedure as necessary and important to successful microcomputing as breathing is to successful living. If you're interested in FORMAT variations between the three versions of CP/M — P+T, Lifeboat, and ATON — see David Otten's article, "A Counter Review of CP/M Variants," elsewhere in this issue.

Correction

An error appeared in "Barbara's Adventures" last time (Vol. 1, #5, the Mar/Apr issue) on page 32. The text should read "300 Baud means 300 bits per second or *30* characters per second," NOT "300 characters per second." This typographical error slipped through and I thank those of you who brought this "gaffe" to my attention.

VISICALC TIP

Robert C. Stockler
Gates, Stockler & Lenz, Inc
4014-C Dutchman's Lane
Louisville, KY 40207

I have adopted an S.O.P. for my VISICALC work that is quite a convenience to me and may be to other neophytes (experienced users have probably been doing something like this for years).

I always put my spreadsheet name (file name, including extension) in Row #1, starting at A1. Then I skip a column or two and put in a label of the date of the last changes or additions, followed a couple more columns to the right by a label indicating the lower right-most coordinate.

Since my machine is 64K and my brain is only 32K, this keeps me from having to think too much about the mechanics of the operations while I'm trying to think about the CONTENT of the materials I'm working on.

Now I don't have to remember what spreadsheet or program I'm working on when it comes time to file it to disk, my hard copy is always identified, and when I want to print-out, I don't have to remember or look to find out what "lower right" should be.

REPRINT SERVICE AVAILABLE

two/sixteen magazine is pleased to announce that as of January 1, 1983 reprint service is available for articles appearing in our publication.

For a fee of $1.50 per article, you will be able to order a reprint of articles published in *two/sixteen*. The only limitation imposed on this offer is that reprints must be ordered within 90 days of the magazine's publication date.

For reprint copies, send your request and check to:

COMPendium Reprint Service
P.O. Box 129
Lincolndale, NY 10540

Feedback

Ralph S. Herring, Technical Support, Pickles & Trout, P. O. Box 1206, Goleta, CA 93116:

Pickles & Trout has several customers who wish to use a WordStar word processing program with a Radio Shack Daisy-wheel II printer. However, these users are having trouble utilizing Wordstar's specialty capabilities, such as underlining, sub- and superscripting, printing boldface, and so on. If any of your readers have discovered a way to implement these features on the daisy-wheel, we would greatly appreciate hearing from them.

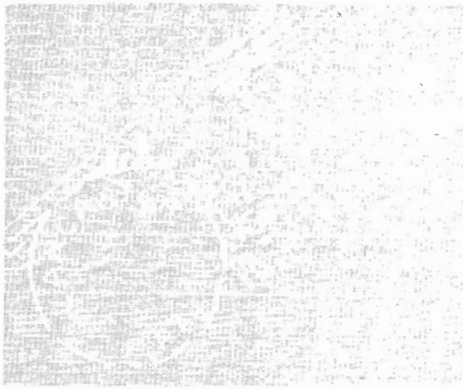
Thanks for your help.

Henning B. Deiter, 3108 Robinson Ave., Austin, TX 78722:

One of the things that annoys me about governments is their habitual use of abbreviations and acronyms that I do not understand. My first reaction to reading *two/sixteen* was much the same.

What (or who) is CP/M? I found much ado about it (or him, or her) in all three of the *two/sixteen magazines*. Who (or what) is EPSON? In which galaxy do I seek the WORDSTAR? Which restaurants serve P+T (Pickles & Trout)? Why does RSJFIITIII16? (Translation: Why does Radio Shack jump from Model II to Model III to Model 16 instead of to Model IV or to Model XVI or to Model II.02.02?) What happened to Models IV, V,...13, 14, XV, etc? Is there a system to all this? Do computer folks do this deliberately to annoy their customers? Is it a sickness, or is it accidental?

Your subscribers who do not happen to have a SPLITNIX can not be expected to know what SPLITNIX is. The opening paragraph of every lengthy article should have an abstract describing the contents in English, preferably Basic English.



This revelation of confusion in the computer world was the most interesting benefit I received from my new subscription. For about a year I have thought it only applied to Radio Shack Manuals. Just learning (but only by reading between the lines) that I could get powerful operating systems for my Model II from people other than Radio Shack made my subscription worthwhile. I had no such idea. I enjoyed the first Alice in Wonderland story, but the second got too deeply into SASASWACAK (shoes and ships and sealing wax and cabbages and kings).

I have some wisdom from experience to contribute: (1) Always make a back-up just before you make a mistake. (2) Never bother to back-up garbage that is saved on paper. (3) If your computer will not respond to any commands, do not turn it off or reset. Call a whiz kid while it is still glowing and he may tell you how to play the Time Prompt and save your work. (4) Do not Fix or Kill the Time Prompt. (5) Except during the Date Prompt, format dates the army way (YYMMDD) so you can sort them.

Peter F. Wells, Box 152, Rindge, NH 03461:

Having read your editorial in the January/February issue, I am still stewing over it a couple of days later and am writing to express my strong disagreement with your viewpoints on CP/M.

I will start with your "final analysis" of the three paramount considerations: (1) standardize, (2) standardize and (3) standardize. My reply is (1) wrong, (2) wrong and (3) wrong. Actually, I will grant that standardization belongs on the list, but only as one of the three and not necessarily the first. The other two are more appropriately user-friendly.

Now back to your first paragraph. The only possible reason that I can conceive of for considering CP/M to be user-friendly is that it is designed along the lines of the operating systems which you are used to from your past experience. Having come into the field more recently, I consider it to be an abortion, or perhaps worthy of being subjected to one.

For a prime example of what I mean by non-user-friendly, I will use the basic PIP command. For starters, what relationship do those three letters bear to what we are trying to accomplish with the command? Yes, I know what they stand for, but to a non-technical person both the letters and the terminology behind them are meaningless and therefore NOT "user-friendly". TRSDOS has CP/M beat hands down with COPY. Now for the rest of the command, i.e., (program name) B:=A:. A large percentage of the population of this country is not mathematically oriented, and to throw something at them which resembles an algebraic formula is a large mistake. Never mind that they had it in high school. That was 15 years ago, and they haven't seen any algebra since; they haven't needed it. Further, using the formula concept puts things in a to-from order, and I would contend that most people will think in a from-to order, adding another element of confusion. Again, TRSDOS wins hands down.

After having spent X minutes getting some poor secretarial type over the shock of th PIP command, all you need, to have her throw a complete fit and go on strike, is to have her practice it and be faced with the return message BDOS error on B: R/O, or worse yet, 0010 0001.

From a personal point of view, I am a technically oriented person and reasonably competent in both BASIC and COBOL. I am actively using both CP/M (P&T version) and TRSDOS. I have even dabbled a little in machine language, and am certainly not frightened by CP/M myself. Nonetheless, I consider the promotion of CP/M to be the modern-day equivalent of the promotion of the QWERTY keyboard, and certainly hope that it dies a natural death at an early age.

No, I am not a "promoter" of TRSDOS either; I recognize at least some of its faults, including its ridiculous rejection of lower-case. If one of your objections is to its wordiness, this is easily handled. The wordiness should be allowed but not required. Let the neophyte type ERASE, and the experienced user type ERA (Kill has an unfortunately too-negative connotation).

The main reason that I have put standardization as low on my priority list as I have is that I have some doubts as to its practicality. To me standardization is more that just syntax, although the standardization of operating system syntax (to a user-friendly version of course, CP/M not included as such) is certainly highly desirable. The big strength of CP/M is that it standardizes on the most popular series of 8 bit processors as well as on the IBM 8" disk format, and therefore provides true transportability.

What is happening now as a result of this standardization? It appears that the 8" disk is on the way out. 5" systems are cheaper by virtue of their smaller size, and now store as much as an 8" disk. With their larger surface, the 8" disk should be able to store far more than the 5" disk, and by virtue of being able to do this in a single module should be more cost-effective on a dollars-per-bit basis than the smaller drives, but this has not happened. Could part of the reason be that in order to do so, the IBM single-density format could not be supported and, from a marketing point of view, no one is willing to fight (obsolete) this standard? I consider the promotion of CP/M for 5" disk systems on the basis of standardization to be somewhat incongruous, as there is no standard data format for the 5" disk and therefore no guarantee in practice of actual transportability. Even if a standard is developed, recognized and supported, how long can it last in the face of technological change?

The same thing holds true for the Z-80 concept; I anticipate that the time will come when 8 bits will die. It will do so on the basis that manufacturing technology for ICs will progress to the point where 8 bits are just not cost-effective compared to 16 bits and up, the market will decrease, and support will cease. Hopefully, a replacement "standard" will emerge, but would you want to guarantee it?

Finally, to many users of computers, standardization is a non-issue. I am specifically thinking of some of the small businesses with whom I deal. They are interested in a computer to perform a specific and somewhat limited set of tasks, and do not care for the predictable future about anything else. Kluge Systems, Inc. could provide them with a unit and the necessary software, not compatible with anything or anyone else, and they would be perfectly happy for several years; after which time technology would have advanced to the point where they would replace their Kluge System with a new and different unit anyway, as it would cost less and provide more than any upgrade could possibly do.

Beyond that, keep up the good work. Controversy is good for us all; it keeps us from becoming complacent.

Alan S. Liebman, Alan S. Liebman & Associates PC, 122 E. 42nd St., New York, NY 10168:

I am a subscriber to *two/sixteen magazine* and after having read the first three copies, I am responding to your invitation to write.

I am an attorney practicing in New York and am the senior member of a firm consisting of six attorneys. We have used Model IIs for approximately two and a half years, using Scripsit, Profile, and programs that operate on CP/M. The programs we use that operate on CP/M are time billing, accounts payable, and payroll.

At the outset, I would like to state that I think your magazine is the first intelligent access to needed information that I have read. In the past years, I have received most of my education about computers from uninformed renegade Radio Shack salesmen, some poorly written soft cover books and numerous magazines which consisted of 90% advertising, only 1% of which related to the Model II.

Overall, having substituted my Model IIs for previously leased Wang systems, I can only state that I am not only pleased, but would be terrified to abandon Radio Shack for another company. The number of horror stories rises in geometric proportion to the number of people one speaks to who use other systems.

Enough of that, I could literally write a book about my past two and one half years' experience. I am writing for a specific purpose; I need information on the following: (1) Does anyone know if you can replace the Model II single-sided drive with two double-sided slim drives, such as those available on the Model 16, or if not, two slim line single-sided drives? (2) Has anybody found a multiplexor that effectively accesses multiple Model IIs to a hard drive? (3) If they have, have they found an operating system that permits them to use simultaneously programs that operate on both TRSDOS and CPM?

Douglas J. Hill, 1197 Cragmont Avenue, Berkeley, CA 94708:

I just received Vol. 1, #4; it's as good as the first three and they were of more use to me than the last two years of *'80 Micro*.

Thornton's article ("More Power to You") causes me to comment as follows: I purchased a Cuesta Systems, Inc. "Datasaver" a few months ago; every time my lights flicker I remember how it used to feel before I got this dandy little box.

My Model II with two Parisitic Engineering "Maxi Disks" keeps on running in brownout and blackout conditions — somewhere between 3-5 minutes — long enough, for sure, to shut down. If you're doing serious work, the reduction in anxiety is more than worth the $695 cost.

Call or Write Your Nearest Snappware Distributor

MICRO-80
284 Goodwood Road
Clarence Park
Adelaide South Australia
Ph-(08) 2117224

DIGI-TEK SYSTEMS
65 Thornridge Circle
Kitchener Ontario N2M-4V9
Ph-(519) 742-8205

STRAWFLOWER ELECTRONICS
50 North Cabrillo
Half Moon Bay
California 94019
Ph-(415) 726-9128

COMPUTER MAGIC
115 Wiltshire Avenue
Louisville, Kentucky 40207
Ph-(502) 893-9334

E-C DATA
Tornevangsvej 88
P.O.B. 116
DK-3460
Birkerod, Denmark
Ph +45/2/81/ 81 91

SYSTEM SOFT
49, Dunvegan Drive
Rise Park
Nottingham, England
NG5 5Dx
Ph-(0602) 275559

CUSTOM COMPUTING
104 Bushwick Rd.
Poughkeepsie, NY 12603
Ph-(914)-471-9318

AEROCOMP
Redbird Airport
Hanger 8
Dallas, Texas 75232
Ph-(214) 339-5104

NOTICE

Snappware Goes On Trial! You Be The Judge!

Snappware knows your programming time is valuable. That's why we are offering a trial package that will cut your programming time up to 75%!

This unprecedented offer allows you to judge for yourself the value of our software using *your* hardware at *your* convenience. Our five best selling products: EXTENDED BASIC, EXTENDED BUILT IN FUNCTIONS, COLLEGE EDUCATED GARBAGE COLLECTOR, AUTOMAP and AUTOFILE, are all available to you on a trial basis for only $50.00 for the Model II and $35.00 for the Model III. We're convinced that after you see how well our software helps you perform programming tasks, you will purchase them like thousands of others have.

Our trial package consists of a master diskette which may be used to create one working copy. Your purchase price for the trial package will be credited toward the purchase of any software Snappware sells.

If for any reason you are not satisfied, just return the trial diskette and working copy and we will refund your money, no questions asked. With an unconditional guarantee like this, you can't lose.

Call our toll free number 800-543-4628 to put us on trial. We're sure your verdict will be: Snappware saves you time and money.

MODEL II Trial Package	**......$50.00**
MODEL III Trial Package	**......$35.00**

SNAPPWARE
SNAPPWARE
SNAPPWARE
SNAPPWARE

Time saving power at your fingertips.
CALL TOLL FREE:
1-800-543-4628
OHIO RESIDENTS CALL COLLECT: (513) 891-4496
3719 Mantell
Cinti., Ohio 45236

VISA

MODEL II

PEEK POKE

Jim Kloosterman
Post Office Drawer 48
Rogers City, Michigan 49779

I do not know why the decision was made to omit the "PEEK" and "POKE" functions from Model II BASIC. There are occasions when they can be useful. In this article I will explain a little more about the DEFUSRn, USRn, and VARPTR functions while providing a dual set of "PEEK" and "POKE" functions you can use — even if you don't know why they work.

To Begin

Fire up your Model II using a disk you can afford to lose, that is, a "clean one" or one for which you have good back-ups for every file. If you have been following my series and have the file SVC/CMD available (see *two/sixteen + twelve*, Vol. 1, No. 4, pp. 24-28), place a copy of that file on this disk. At TRSDOS Ready, type:

DEBUG ON <ENTER>

If you have the earlier file, type:

SVC/CMD <ENTER>

In any event, next type:

DEBUG <ENTER>

Press:

<X>

to be sure you are in heX addressing mode. Press:

<M>

and DEBUG will ask for an address by displaying A . . . . Type:

EF30

and you will get a display as partially shown in FIGURE ONE.

The ??? bytes indicate we don't care about their value. The other values in the first two lines should be correct as shown if you loaded the earlier SVC/CMD file. The remaining four lines are new commands we will all have to enter. Press:

<F1>

and the cursor will jump to the byte just after EF30. Maneuver it using the four arrow keys. If you didn't have SVC/CMD or if any error shows, enter or correct it now. Then, continue copying the above through the C9 at address EF8A. When finished, re-check very carefully! A few minutes now may save you a frustrating hour later on. When you are sure it's okay, Press:

<F2>

then press

<ESC>

and

<S>

At TRSDOS Ready, type:

DUMP PEKPOK/CMD START=EF37, END=EF8A, RORT=R <ENTER>

At the next TRSDOS Ready, type:

DEBUG OFF <ENTER>

Type:

BUILD PEKPOK <ENTER>

For the first command line type:

PEKPOK/CMD <ENTER> and then <ENTER> again.

For the second command line of our DO file type:

BASIC PEKPOK/BAS -M:61239 <ENTER>

and one more <ENTER>

Finally, type:

BASIC -M:61239 <ENTER>

After BASIC's Ready, type the program in LISTING ONE. Check it carefully, but do not run it yet. After checking, type:

SAVE"PEKPOK/BAS" <ENTER>

Then you can type:

RUN <ENTER>

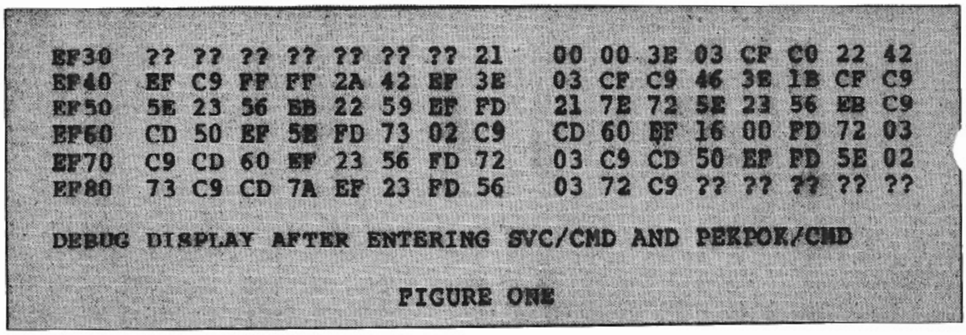
Hopefully, it will work! If it doesn't, carefully check the BASIC listing. If there are no errors there, exit to SYSTEM and turn on and go into the DEBUG program as above, checking the bytes between hex addresses EF50 and EF8A carefully.

Let's Analyze The BASIC Program

Lines 10 and 20 define the variable types, show the program filespec in a handy remark, list the variables used, and DIM the integer array R(0) through R(9). Note that the string variables SW and SB, respectively, set-up for a display upon a white or a black background.

Lines 30 and 40 have four DEFUSRn statements for USR3 through USR6. More on these later but, in order, the machine-language program functions have been called PEEK, PEEKS, POKE and POKES. Line 50 clears the screen, turns-off the cursor and prints the tabbed-over title in black on white background. Then subroutine 200 is called to create a Debug-like, hex format display.

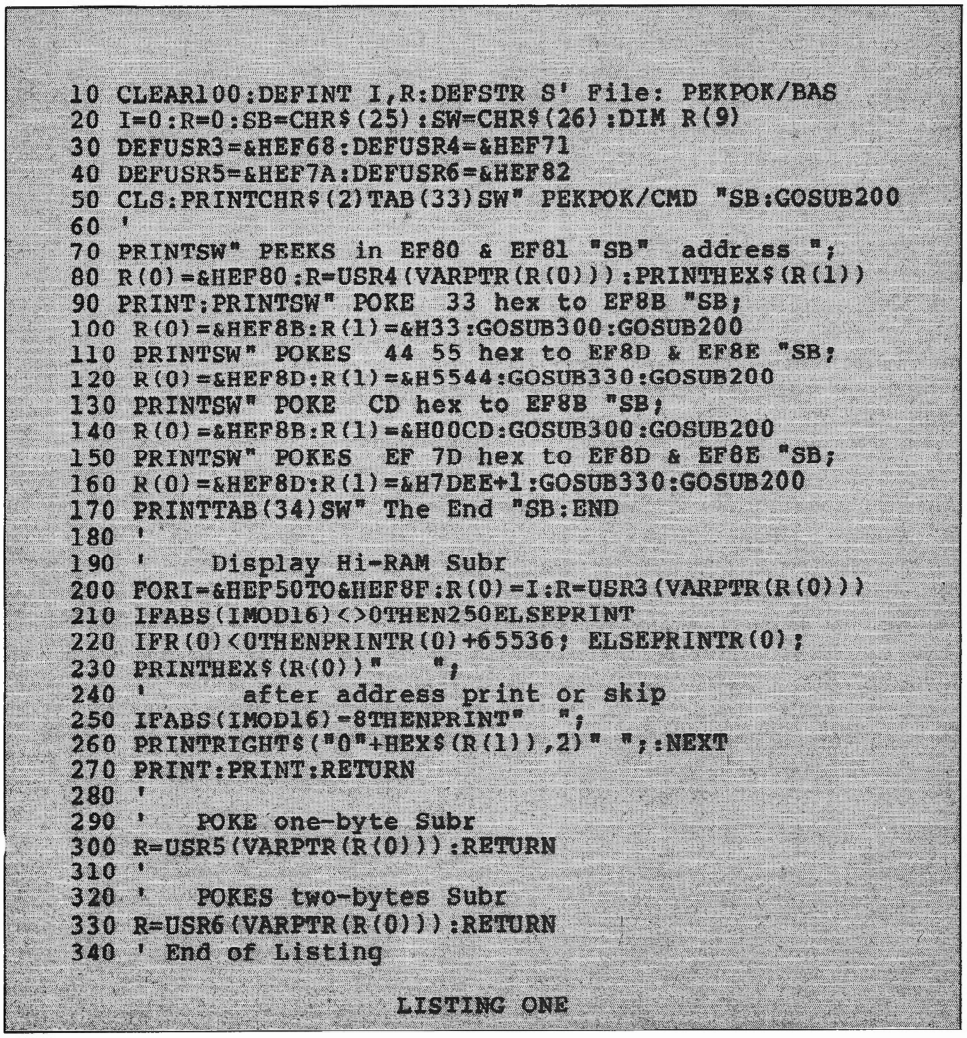
As per Line 70, Line 80 sets R(0) to EF80 hex, then calls the USR4 machine-language program. In contrast to the normal form of X=USRn(Y), all of the USR calls in this program are of the form R=USRn(VARPTR(R(0))). It is unusual but offers practically unlimited volumes of data communication between BASIC and machine language programs, depending upon the size of the integer-array for variable R(I).



```
EF30   ?? ?? ?? ?? ?? ?? ?? 21    00 00 3E 03 CF C0 22 42
EF40   EF C9 FF FF 2A 42 EF 3E    03 CF C9 46 3E 1B CF C9
EF50   5E 23 56 EB 22 59 EF FD    21 7E 72 5E 23 56 EB C9
EF60   CD 50 EF 5E FD 73 02 C9    CD 60 EF 16 00 FD 72 03
EF70   C9 CD 60 EF 23 56 FD 72    03 C9 CD 50 EF FD 5E 02
EF80   73 C9 CD 7A EF 23 FD 56    03 72 C9 ?? ?? ?? ?? ??
```

DEBUG DISPLAY AFTER ENTERING SVC/CMD AND PEKPOK/CMD

FIGURE ONE



```
10 CLEAR100:DEFINT I,R:DEFSTR S' File: PEKPOK/BAS
20 I=0:R=0:SB=CHR$(25):SW=CHR$(26):DIM R(9)
30 DEFUSR3=&HEF68:DEFUSR4=&HEF71
40 DEFUSR5=&HEF7A:DEFUSR6=&HEF82
50 CLS:PRINTCHR$(2)TAB(33)SW" PEKPOK/CMD "SB:GOSUB200
60 '
70 PRINTSW" PEEKS in EF80 & EF81 "SB"  address ";
80 R(0)=&HEF80:R=USR4(VARPTR(R(0))):PRINTHEX$(R(1))
90 PRINT:PRINTSW" POKE  33 hex to EF8B "SB;
100 R(0)=&HEF8B:R(1)=&H33:GOSUB300:GOSUB200
110 PRINTSW" POKES  44 55 hex to EF8D & EF8E "SB;
120 R(0)=&HEF8D:R(1)=&H5544:GOSUB330:GOSUB200
130 PRINTSW" POKE  CD hex to EF8B "SB;
140 R(0)=&HEF8B:R(1)=&H00CD:GOSUB300:GOSUB200
150 PRINTSW" POKES  EF 7D hex to EF8D & EF8E "SB;
160 R(0)=&HEF8D:R(1)=&H7DEE+1:GOSUB330:GOSUB200
170 PRINTTAB(34)SW" The End "SB:END
180 '
190 '    Display Hi-RAM Subr
200 FORI=&HEF50TO&HEF8F:R(0)=I:R=USR3(VARPTR(R(0)))
210 IFABS(IMOD16)<>0THEN250ELSEPRINT
220 IFR(0)<0THENPRINTR(0)+65536; ELSEPRINTR(0);
230 PRINTHEX$(R(0))"    ";
240 '      after address print or skip
250 IFABS(IMOD16)=8THENPRINT"  ";
260 PRINTRIGHT$("0"+HEX$(R(1)),2)" ";:NEXT
270 PRINT:PRINT:RETURN
280 '
290 '    POKE one-byte Subr
300 R=USR5(VARPTR(R(0))):RETURN
310 '
320 '    POKES two-bytes Subr
330 R=USR6(VARPTR(R(0))):RETURN
340 ' End of Listing
```

LISTING ONE

You may recall that executing X = USRn(Y) where X and Y are integer variables, causes the value of Y to be placed in the ASA (Argument Storage Area). When control of the computer is given to the machine-language program, the HL-Register has the address of that ASA. More specifically, HL points to the LSB of the value of Y and the MSB of that value follows in the next successive byte. If the M-L (machine-language) program altered those values in the ASA, upon return to BASIC, Y would be unchanged but X would take the new value from the ASA.

That procedure would work for PEEK if we placed the desired address in Y and then had the value from that address placed in X. However, a maximum of two bytes could be passed in either direction. For a POKE function, we need to pass both a two byte address and one or two bytes of data to be poked from BASIC to the M-L program. Y alone is insufficient to pass more than two bytes! Therefore, I selected a more powerful scheme for both the PEEK and POKE functions for sake of uniformity.

Executing R = USR4(VARPTR(R(0))) places the address yielded by VARPTR(R(0)) in the ASA. The HL-Register will hold the address of the LSB of that address and the MSB will follow. The "value" in the ASA is, in fact, the address of the LSB of the value of R(0) in the array variable table! By knowing that address, each byte of every variable in that array can be explicitly addressed by the M-L program! It can use those values and/or change them as desired. Upon return to BASIC, any new values placed in the array by the M-L program will appear as the BASIC values of those variables. The Interpreter won't even know they've been altered! The variable R simply becomes a "dummy". In fact, upon return to BASIC, the variable R will always be equal to VARPTR(R(0)).

Note that this scheme is a far more general approach than the usual form for USRn. Rather than passing a M-L the value of one variable only and permitting it to pass one value back to BASIC, the USRn(VARPTR(R(0))) scheme passes the M-L program the current location of an entire array of variables, any or all of which it may utilize and/or alter and "pass back" to BASIC!

The four USRn functions operate as follows. USR3 is a PEEK similar to that of the Models I and III. To use it, once you have the M-L programs loaded in hi-RAM with BASIC limited (-M:61239) and DEFUSR3 = &HEF68, you need only set R(0) to the address into which you wish to PEEK. After executing R = USR3(VARPTR(R(0))) the value found in that address will be the LSB of R(1). The MSB of R(1) will be set to zero and the original value in R(0) will be unchanged. The PEEKS M-L program (DEFUSR4 = &HEF71) works the same way, but it returns two bytes in R(1). If R(0) is set to EF00 hex, the contents of RAM address EF00 hex will be placed in the LSB of R(1) and the contents of address EF01 hex will be placed in the MSB of R(1). This is convenient for peeking a two-byte address in one operation rather than a byte at a time and reassembling them.

The POKE M-L program and POKES M-L program (DEFUSR5 = &HEF7A) and DEFUSR6 = &HEF82) operate similarly. In every case, the desired address is placed in R(0). For the one-byte POKE, the value you wish to have poked into that address must be placed in the LSB of R(1). (The MSB of R(1) is ignored and may have any value.) POKES will poke two bytes in one operation. The LSB value of R(1) is poked into the address given by R(0) AND the MSB value of R(1) is poked into the address given by R(0) plus one.

Back to the BASIC program of LISTING ONE. In Line 80, R(0) is set to EF80 hex. Executing R = USR4(VARPTR(R(0))) causes the value in address EF80 to be placed in the LSB of R(1) and the value in address EF81 to be placed in the MSB of R(1). Those two values are displayed — as a single hex address — by the statement PRINTHEX$(R(1)).

As per Line 90, to POKE the single byte 33 hex into address EF8B hex, Line 100 sets R(0) = &HEF8B and R(1) = &H33. Then use of the USR5 function does the job. Note here, R = USR5(VARPTR(R(0))) could have been coded "in-line" as was the USR4 call above but, instead, a subroutine has been set-up at Line 300 for that task. If you are going to use the call more than once, I find the mini- subroutine idea attractive — mostly because I can't seem to type the call correctly more than 50 percent of the time! After the POKE, the Display subroutine of Line 200 is called to show the result.

As per Line 110, Line 120 POKES two bytes, 44 and 55 hex into addresses EF8D and EF8E using the USR6 subrouine function of Line 330. Again, the Display subroutine shows the result. Lines 130 through 160 repeat the POKE and POKES with different values. The program ends in Line 170.

The Display subroutine beginning in Line 200 causes a Debug-like display of the RAM contents from address EF50 through EF8F hex. The FORI, NEXT loop of lines 200 through 260 peeks into each address by setting R(0) to I and calling the USR3 function. At Line 210, if the absolute-value of I MOD 16 is zero, the PRINT causes an "end of line". If R(0) is less than zero, 65536 is added before it is printed; otherwise it is printed without modification. Line 230 prints R(0) in hex.

If the absolute-value of I MOD 16 is not zero, the "end of line" and printing of the decimal and hex values of the address are skipped. At Line 250, if I MOD 16 is equal to 8, the extra spaces for mid-table are printed. Finally, the always two-digit hex value of R(1) is printed by Line 260. If R(1) has only three leading zeroes, it might otherwise print as, say, the single hex digit C. Adding a zero character to the front of the string produced by HEX$(R(1)) insures at least a two hex-digit string. The enclosing RIGHT$ (,2) operator limits the printed string to not more than two hex digits. Thus, exactly two hex digits are printed in all cases for even table-spacing. The quote following adds two spaces between each byte value. (Line 230 adds two more spaces only at mid-table.)

About Those ABS Operators

We are thinking of address EF80 hex as a hi-RAM address 61312 decimal. That is, we like to think of RAM addresses running from 0 through FFFF hex or 65535 decimal. EF80 hex we view as E (14 dec) * 4096 + F (15 dec) * 256 + 8 dec * 16 + 0 dec * 1 or 61312 decimal. But BASIC thinks of all integer variables beginning with a left-most one-bit as NEGATIVE values. It "sees" EF80 hex as -4224 dec (which is 61312—65536). When it calculates -4224 MOD 16 it gets a zero. However, when it calculates EF81 hex, -4223 MOD 16, it gets -1 NOT +1. Likewise, EF88 hex, our 61320 dec, it views as -4216. And -4216 MOD 16 yields -8 and not +8. The ABS operator in Line 210 is not strictly necessary but is a handy reminder that R(1) may be negative. In Line 250, it is essential for proper functioning of the program. Note also that in Line 220, when the decimal value is printed, if R(0) is negative, 65536 is added to it before it is printed to yield "our kind" of addresses rather than BASIC's normal negative numbers.

Summary

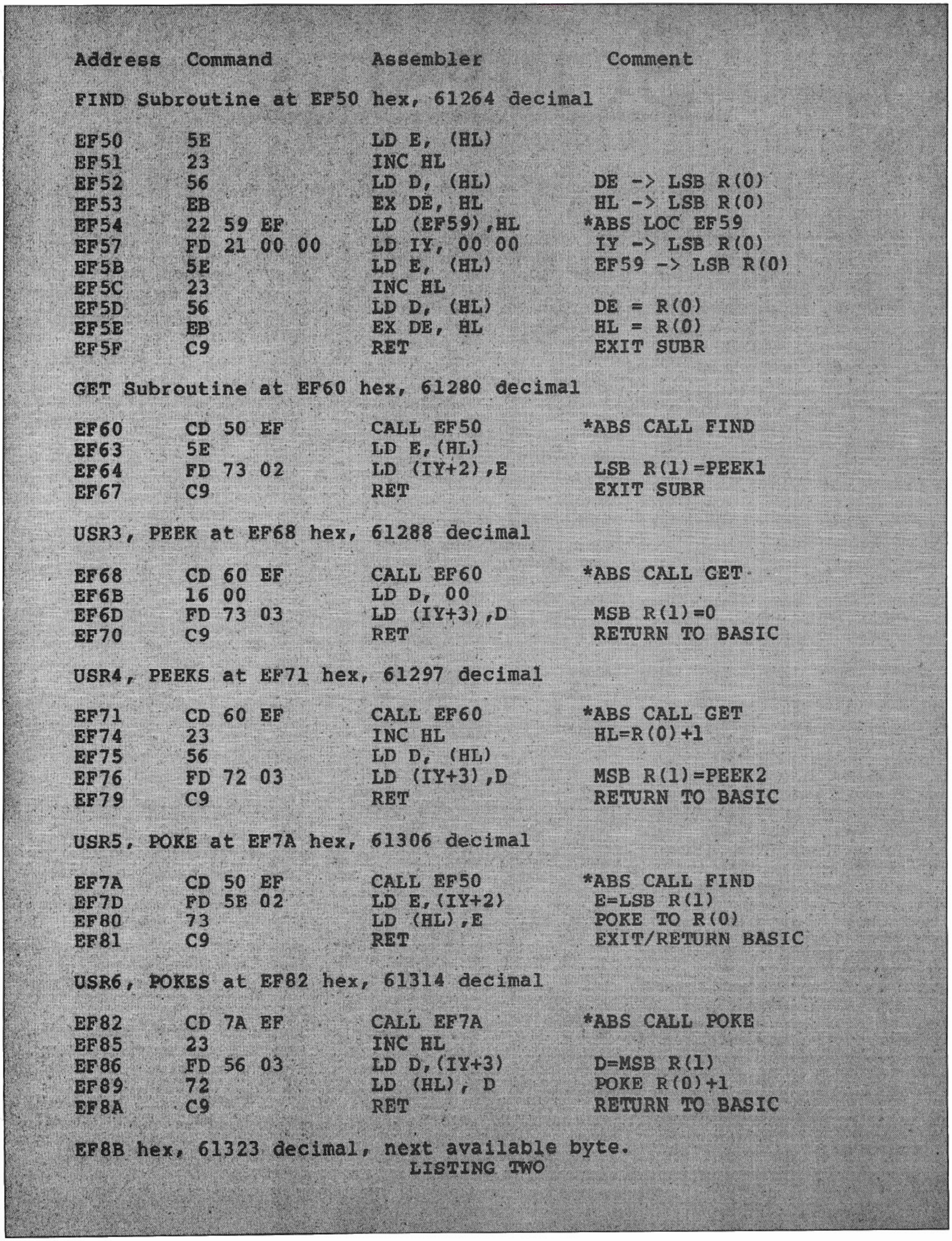
To use the M-L programs for peeking and poking, they must be loaded in hi-RAM beginning at address EF50 hex. When BASIC is loaded, it must be limited by -M:61239. USR functions 3 through 6 must have DEFined addresses as shown in Lines 30 and 40. Finally, there must be an INTEGER array of at least R(0) and R(1) plus some other integer, say R, to serve as a "dummy" variable.

ALWAYS load the desired address in R(0). Execute a one-byte PEEK via R = USR3(VARPTR(R(0))) and find the resulting value in the LSB of R(1). Execute USR4 similarly for a two-byte PEEKS and find results in the LSB and MSB of R(1). For a one-byte POKE, load the value to be poked in LSB of R(1), then execute USR5 in a similar fashion. Load both the LSB and MSB of R(1) for a two-byte POKES upon execution of R = USR6(VARPTR(R(0))).

For those who missed the earlier article, there are three M-L programs shown in addresses EF37 through EF4F hex. Use the following definitions:
DEFUSR0 = &HEF37,
DEFUSR1 = &HEF44 and
DEFUSR2 = &HEF4B.
Define an integer variable such as R. Load the programs into hi-RAM as shown and limit BASIC with -M:61239.

To disable the <BREAK> key, execute R = USR0(R). To enable the <BREAK> key, execute R = USR1(R). R may have any value and is unchanged by this usage. Finally, to protect the top 0 through 22 lines of the display from scrolling, set R equal to the desired value between 0 and 22, inclusive. Then execute R = USR2(R)

Finally, a generic assembler listing of the peek and poke M-L programs is shown in LISTING TWO. *ABS indicates absolute-addresses if you wish/need to relocate the programs.



```
Address  Command        Assembler          Comment

FIND Subroutine at EF50 hex, 61264 decimal

EF50     5E             LD E, (HL)
EF51     23             INC HL
EF52     56             LD D, (HL)         DE -> LSB R(0)
EF53     EB             EX DE, HL          HL -> LSB R(0)
EF54     22 59 EF       LD (EF59),HL      *ABS LOC EF59
EF57     FD 21 00 00    LD IY, 00 00       IY -> LSB R(0)
EF5B     5E             LD E, (HL)         EF59 -> LSB R(0)
EF5C     23             INC HL
EF5D     56             LD D, (HL)         DE = R(0)
EF5E     EB             EX DE, HL          HL = R(0)
EF5F     C9             RET                EXIT SUBR

GET Subroutine at EF60 hex, 61280 decimal

EF60     CD 50 EF       CALL EF50         *ABS CALL FIND
EF63     5E             LD E,(HL)
EF64     FD 73 02       LD (IY+2),E        LSB R(1)=PEEK1
EF67     C9             RET                EXIT SUBR

USR3, PEEK at EF68 hex, 61288 decimal

EF68     CD 60 EF       CALL EF60         *ABS CALL GET
EF6B     16 00          LD D, 00
EF6D     FD 73 03       LD (IY+3),D        MSB R(1)=0
EF70     C9             RET                RETURN TO BASIC

USR4, PEEKS at EF71 hex, 61297 decimal

EF71     CD 60 EF       CALL EF60         *ABS CALL GET
EF74     23             INC HL             HL=R(0)+1
EF75     56             LD D, (HL)
EF76     FD 72 03       LD (IY+3),D        MSB R(1)=PEEK2
EF79     C9             RET                RETURN TO BASIC

USR5, POKE at EF7A hex, 61306 decimal

EF7A     CD 50 EF       CALL EF50         *ABS CALL FIND
EF7D     FD 5E 02       LD E,(IY+2)        E=LSB R(1)
EF80     73             LD (HL),E          POKE TO R(0)
EF81     C9             RET                EXIT/RETURN BASIC

USR6, POKES at EF82 hex, 61314 decimal

EF82     CD 7A EF       CALL EF7A         *ABS CALL POKE
EF85     23             INC HL
EF86     FD 56 03       LD D,(IY+3)        D=MSB R(1)
EF89     72             LD (HL), D         POKE R(0)+1
EF8A     C9             RET                RETURN TO BASIC

EF8B hex, 61323 decimal, next available byte.
```

LISTING TWO

SCRIPSIT'S

SUPER-POWERED USER DEFINED SEQUENCE KEYS

Richard T. Rodgers,
Contributing Editor
The Lawyer's Microcomputer®
P.O. Box 729
Lexington, SC 29072

Reprinted With Permission

Even if you have taken Radio Shack's audio-cassette course as a means of introducing yourself to the SCRIPSIT word processing system, the chances are good that you have only the vaguest recollection that this program offers what is unimaginatively called "User Defined Sequence Keys."

And if you have an excellent memory, you may further recall that Model II/12/16 SCRIPSIT 2.0 has twenty of these things (Super-SCRIPSIT has ten); if your memory is plain fantastic, you might remember that these keys are useful for entering single-stroke salutations to your letters, printing repetitive phrases, and other fairly menial text-generating chores.

I used SCRIPSIT for over a year before it finally occurred to me that this feature was the most powerful of all SCRIPSIT's functions. They will do a great deal more that simply sign-off on letters ("Very truly yours, ... etc.) and furthermore, they are considerably easier to use than I first imagined. As far as I am concerned, they are Tandy's best kept secret.

If you have been reading independent product reviews or evaluations of SCRIPSIT, you have probably gathered that the principal objection most reviewers have to this system has nothing whatsoever to do with the program's design; most of the complaints are directed at the limited number of "special function" keys found on the keyboard. As you know, there are only two of them (the <F1> button and the <F2> button), and these keys get a considerable work-out during a typical SCRIPSIT exercise.

As you begin to use this system intensively, you will find its menu-driven design to be something of an impediment to its speed of operation. As an example, I frequently want to <S>wap diskettes after I have finished working on a document. To do this, I must first go back to the directory, <CTRL+Q>, then (from the directory) enter the Utilities Menu, <U>, and (once the Utilities Menu appears), type an <S>.

NOTE: Mr. Rodgers is currently working on a book entitled, *A Heavy Duty User's Guide to SCRIPSIT.*

The keystrokes look like this:

1.) <CTRL+Q> Quit the document
2.) <U> Enter the Utilities Menu
3.) <S> Swap diskettes menu choice

After all this, the screen instructs me to insert a new diskette and press <ENTER>.

Every time I wish to swap diskettes I must go through this *same* exercise.

Let's take this simple "go to Disk Swap status" series of commands as a means of introducing you to the User Defined Sequence Keys:

First, go to your SCRIPSIT Directory and enter the Utilities Menu. Once in the Menu, type <U>, signifying you wish to enter into the User Defined Sequence Key system.

The screen presents you with an interesting configuration with some not-too-well documented User prompts.

Since, for the purpose of this exercise, I have decided to program the "Zero" ("0") key to get me into the Swap Diskettes mode, type a <0> and see if anything appears within the box. If you find text here, then you (or someone else) has already programmed the <0> key to do something for you ... and it is now up to you to decide whether you wish to erase this material and continue this lesson or, alternately, proceed with this exercise using some other (i.e., "vacant") User Defined Key. I will assume that the "0" key will be used for this exercise.

Now is a good time to think about what you are doing and what you want this User Defined Key to do when you have finished programming it. Let's see ... we wanted to quit the document, go to the Utilities Menu, and then enter the Swap Diskette mode. O.K., simply enter the sequence of keystrokes that you would type to accomplish this result. Here goes:

<CTRL+Q> <U> <S>

That doesn't look too difficult, does it? The problem is that we are still in the programming mode for this User Key. Somehow we need to tell this system that has all the information we want to put in this key. So the next question is how do we stop this keystroke-capturing exercise? You do it by typing <CTRL+Z> ... so do that now.

The instructions you have put in the User Defined "Zero" key are:

<CTRL+Q> <U> <S> <CTRL+Z>

You should now be back to the Utilities Menu. From here, press <BREAK> to return to the Directory, and then open any document on your disk.

Now let's try to use this specially programmed "zero" key. Place your cursor anywhere within the document you have opened. Next, type <ESC> <0>. Pretty slick, huh? You should have proceeded directly to the Swap Diskettes mode ... all without typing any keystrokes other than the <ESC> <0>.

Now it's your turn. Another (even shorter) series of commands I frequently use is what I call the "sign-off" routine. This is the way I get out of SCRIPSIT and go all the way back to TRSDOS READY.

Again, assume your cursor is in the middle of a document that you have been working on. Also assume it's time to leave SCRIPSIT and start running another program (such as VISICALC).

How do you get from here to the TRSDOS READY prompt? It takes only four keystrokes to do this so go ahead and program the <SHIFT> + <0> key (the ")" button) to be your specially defined user "bailout" key. Go ahead. Program this key to get you out of a document and to the TRSDOS READY prompt.

Well, so far, we have done two fairly unspectacular things with the <0> key. <CTRL> + <0> takes you immediately to the Swap Diskette mode; <CTRL> + <SHIFT> + <0> key bails you out of SCRIPSIT altogether. What should now be apparent is that these User Defined Keys can be used to command SCRIPSIT to perform a series of instructions which you have hitherto typed manually. These keys, in other words, are not simply limited to generating short bursts of canned text (though they can be used for this purpose also).

To stimulate your thinking, here are some other processes to which these special function keys can be addressed:

Example #1
Repaginating, and then creating a footer page.

```
<CTRL+U>
<R>
<ESC>
<CTRL+G>
<1><F><S><ENTER>
<P><a><g><e><space><{><P><}> <ENTER>
```

Example #2
Assuming you have the optional dictionary module, you will probably hyphenate a document and immediately thereafter repaginate it. To avoid entering these commands repeatedly, try saving three keystrokes in a User Key like this:

```
<CTRL+U>
<H>
<CTRL+U>
<R>
<ESC>
```

Note: This sequence assumes your Disk Defaults for the repagination process are set to the desired standards for the subject document.

Example #3
Envelope creation was always a problem until I realized that I only needed to copy the name and address found in the salutation of the letter, and then transfer it to a properly formatted separate page. To do this with a User Key, first place your cursor at the beginning of the address (over the "M" in "Mr. John Jones", for example) and then

```
<CTRL+D>
<P>
<P>
<P>
<P>
<D>
<CTRL+G>
<A>
<R>
<CTRL+F>
<R> <1> <ENTER>
<CTRL+D>
<B>
<F>
```

Note: This exercise assumes your format line for envelopes is set at "format line recall position #1."

Since this looks more complex than the previous example, let's study the steps taken in this series of commands:

a.) Define the four lines of the address;

b.) Duplicate the defined block;

c.) Add a page to the document (this will be the envelope);

d.) Replace the defined block;

e.) Change the format line to "envelope specifications" previously saved as Format Line #1;

f.) Define all below cursor; and

g.) Format the defined block.

The ending result should be an envelope-formatted name and address located on the final page of your "letter" document. Assuming you use the hand-feed method of passing paper through your printer, you will feed your stationery through the machine until the last page of the letter is typed ... then you will insert an envelope which SCRIPSIT will print as the last "page" of the document.

Example #4
Here's another use. After repaginating a document, you will always want to review each page ending and page beginning just to be certain the repagination process has not severed important blocks of your text.

By way of example, it looks a little strange to have "Very truly yours" at the bottom of one page and then your name (which is supposed to appear below your signature) at the top of the next. While you can take care of this problem before <R>epaginating using the <B>lock-<L>ock command, I usually don't remember to do this until after the problem has already arisen ... so I need a retroactive cure to put the block of text together again. To get a one-keystroke approach to moving these text remnants from the bottom of one page to the top of the next, simply position your cursor at the *beginning* of the text which you wish to move and

<D> <B>
<M>
<N>
<R>

In terms of SCRIPSIT's commands, this series of instructions tells the machine:

a.) <D>efine all text <B>elow cursor;

b.) <M>ove it;

c.) go to the <N>ext page;

d.) <R>eplace the text at the top of the next page.

Example #5

Conversely, to move from the top of one page to the bottom of the previous page, place your cursor at the *end* of the text to be moved, then:

<D> <A>
<up arrow>
<R>

In terms of SCRIPSIT's commands, this series of instructions tells the machine:

a.) <D>efine all text <A>bove the cursor;

b.) <M>ove it;

c.) Go to the end of the previous page;

d.) <R>eplace the text at the end of the previous page.

If you have gotten this far, you have probably thought of several additional functions you would like to program into your User Keys. This list of exercises is by no means exhaustive. My own SCRIPSIT System Disk now has eighteen of its twenty User Defined Keys occupied with pre-programmed instructions to ease the burden of several routine chores we encounter daily. Substantive System disks have pre-defined User Keys of their own to accomplish whatever functions are unique to the system text contained on the disk.

Before you get too deeply into programming User Keys, you need to know two things: First, how to "Edit" the keystroke sequence you have programmed into a certain key; second is how to program a User Key directly from a document (i.e., *without* going back to the Utilities Menu).

Changing what you have already programmed into a User Key is a fairly structured task. SCRIPSIT's typical powers of insert, delete, move, etc. work a different way from the normal keyboard-to-screen techniques you have learned for document creation. Basically, you have three editing commands or choices:

1.) Erase all the contents from the User Key;

2.) Insert one character; and

3.) Delete one character.

Choice #1 is simple. Type a "C" when the User Defined Key contents (located in "the box") are first displayed and they will be instantly erased. Thereafter, your User Defined Key will contain nothing (unless, of course, you add something new to it).

Deleting a keystroke (choice #3) is also fairly typical. Simply place your cursor over the character to be deleted and press <F2>.

Inserting another character (choice #2) is a more tedious two-step process. You must first *open a "space"* in the sequence of characters already programmed into the key; to do this, position your cursor upon the "opening" area and type <F1> which will open one blank space under the cursor. Next, simply type your new character into this newly created blank space.

Note that changing a keystroke by simply typing over it does not work in the usual fashion. If you wish to change a keystroke your must

1.) Delete <F2> the offending character;

2.) Then insert a blank space; and

3.) Then type in the new character in the newly inserted space.

When you are finished modifying your captured keystrokes, move the cursor to the final character in the text (the "reverse video" <CTRL+Z> combination) and then type <CTRL+Z> again.

If you have been entering all of your keystrokes through the Utilities Menu, you have probably found a number of errors in your work product. It's difficult to remember every single keystroke you must strike when you cannot see the effect of each one on your text. As I mentioned before, there are two ways to programming the User Keys, only one of which is through the Utilities Menu. Let's now try the second (and frequently more direct) technique.

To illustrate this second method, we will use the exercise described in Example #4 whereby all the text below your cursor is moved to the beginning of the next page (the post-repagination page-ending adjustment exercise).

Open any "test quality" document on your disk. Go to any page and look at the text at the bottom of it. Let's assume that you wish to move the bottom three lines of this text over to the beginning of the following page. If you were doing this manually (i.e., the "old fashioned" way), you would place your cursor at the beginning of the text to be moved, <D>efine all below, <M>ove, <G>et the <N>ext page, and <R>eplace.

To program a User Defined Key to accomplish this same task, all you do (before typing in the above keystroke sequence) is to "turn-on" the capture keystroke sequence of a User Defined Key. To do this, first type <CTRL+Z>. You will see the User Defined Key Menu suddenly substituted on the screen in place of your text. At this point, type in the number of the key you wish to use hereafter for the capturing of these keystrokes. In this illustration, I will use Key <8> ... so type <8>.

The screen suddenly returns to your previous text, your cursor is exactly where you left it. And now enter the keystroke sequence we previously discussed as follows:

<D> <B>
<M>
<N>
<R>

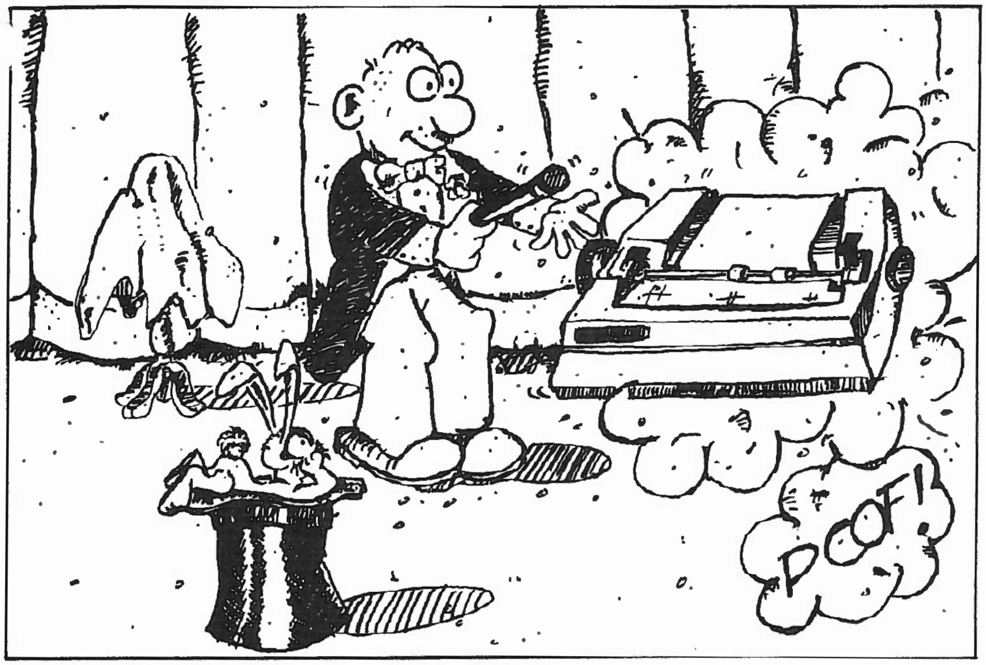
You will note, but for the first command (the <CTRL+Q> entry instruction) and the last command (another <CTRL+Q> instruction), we did nothing special to capture this keystroke sequence. You probably also notice that nothing seemed to happen when you pressed the <CTRL+Q>.

To see if your exercise worked, move your cursor to the third line of the bottom of any page in your test document and then hit <ESC> + <8>. Did this text suddenly get moved to the beginning of the next page? If so, you have successfully programmed the User Defined Key <8> without going through the Utilities Menu.

Just to convince yourself that you did the job right, <Q>uit the document and go back through the directory to the Utilities Menu. Select <U> (the User Defined Key option) and take a look at Key Number <8>. Sure enough, all the key strokes you previously entered while you were working with your document have been captured in this User Key Number <8>'s instruction repertoire. Isn't that easier than doing this the hard way (i.e., directly through the Utilities Menu)?

I hope this article has opened your eyes to the remarkable powers of the User Defined Keys. When I made this discovery, it suddenly dawned on me that the Radio Shack hardware, even without all the "special function" keys other computers may have, is even more powerful than the "ideal" machine which might have twenty of its own "hard-wired" function keys ... at least using *this* technique the user can tell the machine to do exactly what he wants and not what some programmer thought he needed while the program was being written.

Your SCRIPSIT word processor has twenty special function keys which you can program to do whatever tasks you select. Now it's your turn. Go forth and create.



MAGIC (WAND)

for the

DAISYWHEEL II

William John Elliott @ Aramco
Box 4302
Ras Tanura via Dhahran
Saudi Arabia

I have just been introduced to one of those programs you wish someone would create and doubt anyone ever will.

For over a year, I had been using Magic Wand for the production of letters, reports, and other correspondence on a Model II with a Radio Shack Daisywheel II printer. When I first set up this system I expected Magic Wand to use all the available talents of the Daisywheel II in the same way Scriptsit does. I was very disappointed when I discovered that proportional spaced right justified text was beyond the capabilities of the Daisywheel II-Magic Wand combination.

Other disappointments were soon to follow. The printer would not bi-directionally print, and I even had to write a small printer setup program to get it to underline and space lines properly.

Now, I like the Daisywheel II. It has metal where other printers have plastic. In the year we have been operating with it we have had exactly two problems. First, when we set the printer up we got the cable upside down in the Model II. Some are marked "top", some are not. (Just remember, the flat cable goes down, not up.) Second, after about ten months we began to shatter daisywheels. We broke four in a week. We checked with other users and found an answer to our problem. The solution was so simple it had escaped us. We had accidentally moved the Copy Control Lever to accept thick copy (as in multiple carbons). Every time the hammer hit the daisywheel it exceeded its structural limits. The result was broken daisys. The solution was to adjust the Copy Control Lever.

I also like Magic Wand. It is the first word processor I learned, which usually gives a word processor an edge. It uses the arrow keys for cursor moves. Even after learning Wordstar and using it extensively, I think its method of double key calls is clumsy when compared to the Magic Wand single stroke system.

Even though it would be nice to be able to control the printer completely, we chose to continue using the Daisywheel with Magic Wand. As a result, for a year all of our reports and letters were ragged right at either 10 or 12 characters per inch.

Then I saw an advertisement for the TENON SOFTWARE SERVICES (previously MicroProgrammers) Magic Wand-Daisy II interface patch which claimed to be an answer to our wishes. The software is available in several combinations. You can purchase the Daisy II interface patch alone for $75, or a patched version of Magic Wand with the Daisy II interface for $425. TENON also offered other software products including ATON CP/M. We use ATON on our Model 16 and Model II for data processing but were never able to get our version of Magic Wand to operate on ATON CP/M.

Our master Magic Wand disk was sent to TENON and in a reasonably short time we received a copy of the patched Magic Wand suitable for use with ATON CP/M. We also received a disk containing the Daisy II interface patches and utilities.

It should be pointed out that TENON does not support Magic Wand version 1.0 since they use version 1.11. They will gladly update your original master Magic Wand disk as they did ours for $25. When you order the system be sure to specify your CP/M vendor and version.

After making the appropriate backup and active copies of both primary master disks, I built a custom master disk by following the two pages of clear and complete instructions which accompanied the Daisy II interface disk. Starting with an ATON CP/M double density disk containing DDT.COM, SUBMIT.COM, and XSUB.COM, I used PIP to copy PRINT.COM, PRINTDII.HEX, PROP.HEX, MWPATCH.SUB, BID.COM, and DOCUMENT to the CP/M disk. DOCUMENT is a disk file containing the text of the instruction sheets, an excellent feature I would like to see others emulate.

The actual patching is simplicity itself. Type SUBMIT MWPATCH and MWPATCH uses PROP.HEX and PRINTDII.HEX to create a new PRINT.COM modified file while renaming the original file "PRINTO. COM". The two files PRINT.COM and BID.COM can now be PIPed to all disks used for Magic Wand printing.

The time to test the new patched system in actual combat had arrived, and at this point I found the one slight drawback to the application. The instructions tell you that you must run the BID.COM program prior to printing a document. BID.COM is run by typing BID <ENTER> in CP/M mode prior to typing PRINT (filename) <ENTER> to print your text as prepared by Magic Wand EDIT. Since BID was apparently a DAISY II setup program I assumed you only had to run it each time the printer was powered up. I soon discovered that this assumption was incorrect. BID *MUST* be typed each time you run PRINT. This is not a big inconvenience, but it can be annoying when you forget and the whole system hangs up. It is necessary to reboot to get going again. I would have preferred an additional patch in the PRINT program with the appropriate error catching aborts and messages for beginners.

One inelegant method of avoiding the problems associated with forgetting to run BID is to rename it PRINT.COM after renaming PRINT.COM something like MWPRINT.COM. Since the user is accustomed to running PRINT, he will automatically run the printer setup program first. Even leaving the names the same, the inconvenience is quickly accepted and completely overshadowed by the advantages of the interface program and utilities.

BID also allows you to modify the left margin, right margin, lines per page, pitch and direction of printing by typing BID <lm> <rm> <lpp> <p> <d> when you invoke BID. If no changes are made the defaults are 2 135 66 10 0. A zero in any position selects the default. PRINT.COM sets the pitch which can be reset by PRINT.COM or BID.COM. The instructions also include directions for using BID with programs other than Magic Wand in the CP/M environment. It is even compatible with the CP/M CTL-P print feature. You can force proportional spacing from other CP/M programs by setting <p> to a number other than 0, 10, or 12. I tried using BID with a Wordstar file and was able to get proportional spacing and bi-directional printing. I have not been able to get the right justification with proportional spacing with Wordstar, but I haven't given up trying!

Due to a quirk in the way our disk expansion is arranged relative to the Model II, I uncovered a very minor bug in the program. We use drive A (or 0 for TRSDOS users) for the operating system and Magic Wand, and and put data files on drive D (or 3) in the disk expansion. I found that after editing a file and saving it with Magic Wand that the PRINT program (after BID) would indicate an "invalid file name" error. After I rebooted, everything worked fine. The file was not lost, and the name was correct; the program simply would not recognize it. If drives A or B are used as data disks (which is normal for most Model II/16 installations) this problem does not appear. If drives C or D in the disk expansion are used the problem is consistent.

My first printout was a disappointment, but it had nothing to do with the DAISY II interface. It had the appropriate even right margin and proportional spacing, but something was wrong. The characters were bunched and overlapped in places. Then I remembered instruction number one on the instruction sheets concerning the printer. I quote, "Load the Daisy to the top of form and install correct printwheel."

One Madeleine Style proportional spacing daisywheel (CAT. NO. 26-1422) and one printout later I was a happy user. There are very few documents that don't look superb in right justified proportional spaced form. The price of the system was worth that alone.

But there was more.

In addition to the justified proportional spacing, the DAISY II interface allows boldfacing (use command BF4 for proper intensity), underlining, superscripting, subscripting and, of course, bi-directional printing.

I was looking forward to the bi-directional printing as a means of reducing printing time, but due to slight delays between lines, I doubt there is any significant increase in speed. One very beneficial side effect of bi-directional printing is that the wear and tear on both the printer and the printer stand due to the slam-bang action of the returning carriage has been eliminated. The use of incremental vertical control also allows you to print at four lines per inch and reverse feed.

I am sure many users of the DAISY II are not aware of the many characters on the wheel that are not directly accessible to the keyboard user. TENON includes instructions for revising the character table in the 64K Pickles and Trout 2.2e CP/M system using DDT to allow access to these characters. Although I attempted these changes on both ATON and Pickles and Trout 2.2ed CP/M I was unable to achieve the desired results. This could be due to incomplete instructions, incompatible CP/Ms, or this user's misunderstanding. I suspect all three. I have written TENON for further clarification.

In conclusion, I strongly recommend the purchase of the TENON DAISY II interface patch to anyone using Magic Wand with the Daisywheel II printer. Although I contacted TENON from the opposite side of the world, the TENON employees answered my questions in a friendly and prompt manner with complete information and additional suggestions to aid in the use of their product. All in all, it has been a very pleasant experience.

TENON MAGIC WAND-DAISY II INTERFACE PATCH

Available from:
Tenon Software Services, Inc.
1910 Fairview Avenue East,
Seattle, WA 98102
206/324-0116.
Price: $75.00

We're serious about fun!

Software for II/12/16

RIOCHET© This is a fast action game that really tests your reflexes. The object of the game is to get the ball into the goal by deflecting it off the walls before your time is up.

TKSDOS® $25.00

STARTREK™© A large version with lots of graphics of the classic computer game. Takes about two hours to complete. ™ Designates trademark of the PARAMOUNT PICTURES CORPORATION.

TRSDOS® $25.00

STARSHIP/I© (By Richard H. Young, Editor of two /sixteen magazine.) This is a save-the-galaxy type game based on the venerable STARTREK games which have been played on large mainframes for more than ten years. It is specifically designed for the RS Model II/12/16 and features a constant console display with no scrolling. It is a "strategic" game (Opposed to "tactical" games which involve reflex action but little thought). However, STARSHIP requires quick thinking, as all events are timed. STARSHIP requires quick thinking, as all events are timed. STARSHIP comes with a disk instruction file which can be listed on your printer. It can also be listed on the screen at the beginning of a session. In addition, STARSHIP contains a few undocumented "surprises" at advanced levels of play (there are ten levels of difficulty).

TRSDOS® $39.00

GAME PACK I© (by ADD, Inc.) Four games that use enhanced graphics not available from Radio Shack. INTERCEPT and REVERSEM for two players; LUNAR LANDAR and SINK THE BATTLESHIP for one player.

920-0001 TRSDOS® $50.00

GAME PACK II© (by ADD, Inc.) Consists of the four games in GAME PACK I© and two additional games. They are EAT'UM where you control a snake moving around the screen trying to catch its prey and REVERSEM II where it's you against the computer.

TRSDOS® $70.00

GAMES/TRS-80© (by Lance Micklus, Inc.) Contains the following six games:

THE MEAN CHECKERS MACHINE© — The game is played just like regulation Checkers. To win, you must eliminate all of your opponent's pieces from the checkerboard.

STARFLIGHT® — This is a Star Trek type game. The object is to collect data about Omega VI, an unexplored area of the galaxy; to locate and destroy the 20 Klingon Invaders.

CONCENTRATION — The computer will make up the game board. You will try to find the matching prizes. After each correct guess, you will have a chance to guess at the secret number. Guess right, and you win all of your prizes.

TREASURE HUNT — The Lumas caves, located in Lumusville, Vermont, are said to have 20 hidden treasures in them. Few explore the caves because it is said that pirates and dragons live there, and there are deep pits which many have fallen into and died. You, a smart and brave human, and I, an alert computer, will explore the caves and try to find the treasure.

BANKO — The Game is similiar to Black Jack. You draw numbers stopping before the total point value exceeds 11. If you go over 11, then your hand will be zeroed. If you draw 5 times without going over 11, then you'll receive bonus points equal to the value of your hand at the time you end your turn.

DOG STAR ADVENTURE© — The evil General Doom and his Roche Soldiers are ready to launch an attack against the forces of freedom lead by Princess Leya. While traveling to her secret command center. Doom attacks Princess Leya's ship. She is now being held prisoner on one of General Doom's battle cruisers. We must try to save the Princess and the treasury of her freedom fighting force.

TRSDOS® $75.00

ADVENTURES 1-12© (by Scott Adams of Adventure International) By definition, an adventure is a dangerous or risky undertaking: a novel. exciting. or otherwise remarkable event or experience. On your personal computer. Adventure is that and much more. In beginning any Adventure. you will find yourself in a specific location: a forest, on board a small spaceship, outside a fun house, in the briefing room of a nuclear plant, in a desert, etc. The top portion of your video display will tell you where you are and what you can see; the bottom section of the display is devoted to inputting commands to your robot computer and receiving messages that may arise as the result of your orders.

The object of a game is to amass treasure for points or accomplish some other goal such as preventing the destruction of the automated nuclear plant in Mission Impossible. Successfully completing a game, however, is far easier to state than achieve. In many cases you will find a treasure but be unable to take it until you are carrying the right combination of objects you find in the various locations.

#1 ADVENTURELAND — Wander through an enchanted realm and try to recover the 13 lost treasures.

#2 PIRATE ADVENTURE — The lost treasures of Long John Silver lie hidden somewhere — will you be able to recover them?

#3 MISSION IMPOSSIBLE ADVENTURE —In this exciting Adventure, time is of the essence as you race the clock to complete your mission in time —or else the world's first automated nuclear reactor is doomed!

#4 VOODOO CASTLE — The Count has fallen victim to a fiendish curse with you his only possible hope. Will you pull off a rescue. or is he down for the Count for good?

#5 THE COUNT — It begins when you awake in a large brass bed in a castle somewhere in Transylvania. Who are you, what are you doing here and WHY did the postman deliver a bottle of blood?

#6 STRANGE ODYSSEY — At the galaxy's rim, there are rewards aplenty to be harvested from a long-dead alien civilization, Will you be able to recover them and return home?

#7 MYSTERY FUN HOUSE — This adventure puts you into a mystery fun house and challenges you to find your way through and back out of it.

#8 PYRAMID OF DOOM — This is an Adventure that will transport you into a maddening dangerous land of crumbling ruins and trackless desert wastes — into the very PYRAMID OF DOOM!

#9 GHOST TOWN — You must explore a once-thriving mining town in search of the 13 hidden treasures.

#10 SAVAGE ISLAND PART I — A small island in a remote island in a remote ocean holds an awesome secret — will you be able to discover it? This is the beginning of a two-part Adventure, the second half concluding as SAVAGE ISLAND PART 2, Adventure #11.

#11 SAVAGE ISLAND PART II — The suspense begun with Adventure #10 now comes to an explosive conclusion with SAVAGE ISLAND PART II!

#12 GOLDEN VOYAGE — The king lies near death in the royal palace. You have only three days to bring back the elixir needed to rejuvenate him.

TRSDOS® $129.95
CP/M® $129.95

INTERACTIVE FICTION© (by Robert LaFore) The ability to actually interact with characters and situations — to take part in the story-sets INTERACTIVE FICTION™ apart from the competition. the The actions begins when you react to a given situation or event, and then key in your response. You aren't limited to a couple of words; using full sentences is encouraged. You can enter the same situation time and again. and by using a different response. experience new twists of the adventure.

LOCAL CALL FOR DEATH — Meet Detective Sir Colin Drolley, as well as a host of other intriguing rogúes, as you set about the task of solving the perfect crime.

TWO HEADS OF THE COIN — You are one of the world's greatest detectives. Mr. Conway's wife, Georgina, has vanished, and now it's up to you to gather the clues and solve the mystery behind her disappearance!

HIS MAJESTY'S SHIP "IMPETUOUS" —You are at the helm of HMS Impetuous. As captain, your decisions could bring either fame or fortune or utter disaster to you and your crew.

SIX MICRO STORIES is an excellent introduction to the world of Interactive Fiction. They are six short, but complete, stories which involve you, the reader, in a variety of fascinating scenarios. They include THE FATAL ADMISSION, EMPTY WORLD, and four other stories.

TRSDOS® $49.95

TYCOON® (By Barry Ansin) This is a monopoly like game that you can play against the computer.

TRSDOS® $35.00

BIORHYTHMS® This will print on 8½ by 11 paper or display biorhythms charts of an individual starting any date and for as many days as desired. Includes date subroutines, which are used for testings, converting, and calculating time between dates.

TRSDOS® $25.00

AMORTIZE© Financial amortization calculator. Prints high quality amortization schedule on 8½ x 11 paper. Calculates one of the missing values, principal, payment, periods, interest rate, or balloon payment. Prints totals for each year. Includes full screen subroutines.

TRSDOS® $25.00

LISTER© Lists BASIC source programs on 8½ x 11 paper with provisions for separate multi-statement lines. Prints program name, date, time and first comment on each page.

TRSDOS® $25.00

DELUXE PERSONAL FINANCE© (by Lance Micklus, Inc.) This is a sophisticated and unique financial analysis package which is expense oriented. It can be readily customized to suit your personal financial situation.

TRSDOS® $79.95

FOREIGN LANGUAGE VOCABULARY BUILDERS© (by ADD, Inc.) Consists of over 1000 words and phrases that are tested in flash card fashion. You may add over 1000 additional words of your own choosing. Languages currently available are: FRENCH, SPANISH and GERMAN.

TRSDOS® (One language per disk) $45.00

HANDICAPPING© For classifying and ranking thoroughbred flats based on scientific methods. Use of this program will give you the edge and build skills resulting in rewards at the track.

TRSDOS® $100.00

Additional Programs Available
Call or Write For Free Brochure
and More Information.

Rizzo Data Systems Corp.

577 Burlington Rd., P.O. Box 458 • Bridgeton, NJ 08302-0356 • 609/451-7979

Shipping Charge $3.00 Per Order - C.O.D.'s - Add $2.00 • 30 Day Money Back Guarantee On All Programs

Visa, MasterCard, Personal Checks or Money Orders Accepted

A REVIEW OF

DOSPLUS II

A New Alternative

Dr. David A. Gash, President
Creative Micro Systems, Inc
1514 Cedar Ridge Place
Cushing, OK 74023

The task of reviewing DOSPLUS II, an alternative operating system for TRS-80 Model II users currently using TRSDOS, turned out to be more of a job than I had thought. In the first place, DOSPLUS II is somewhat more than just an "operating system." A better characterization might be a "computer management system", due to its comprehensive abilities (more on that later). Secondly, in addition to the DOS, a set of rather powerful enhancements (modestly referred to as "patches") to the BASIC interpreter are included; I knew that, as a programmer, I couldn't fail to examine those. And lastly, the manual, rather than a bunch of Xeroxed, stapled-together sheets resembling notes from one systems programmer to another, is 213 nicely-bound pages of (on the whole) clear, well-written, and useful information — not counting the 127-page technical manual. As you can see, I had my work cut out for me!

Let's look at a few of the system's better features, rather than try to cover every major item.

Operation And Commands

DOSPLUS II includes all the standard features we have become accustomed to under TRSDOS, together with some wonderful extras as well. A few examples of small but significant niceties DOSPLUS II provides:

(a) the date and time questions may be answered in nearly any format, or bypassed completely;
(b) alphabetic character case is ignored in commands and filespecs;
(c) multiple commands may be entered on a single line by separating them with semi-colons;
(d) CP/M-like "wildcards" are allowed in nearly every command;
(e) double-sided disk drives are supported (Model 16 owners, take note!); and
(f) DOSPLUS II's formatting scheme allows about 88K more space per (single-sided) disk than TRSDOS.

The library commands range from the recognizable APPEND to VERIFY, with a few new ones thrown in along the way. Among these are CAT, which shows you just the names of the files on the disk (rather than the detailed information obtained by DIR); CONFIG and SYSTEM, with which you may alter the system's configuration to suit your needs; FILTER, which can intercept and translate any given value into another on its way to or from a system device; LINK, by which two logical devices may be coupled; and ROUTE, which re-directs I/O to a device other than that originally intended.

All this talk of "devices" brings up a very important feature of DOSPLUS II — it is completely device independent. Lest this slip by us unnoticed, consider: it means any device may generally be accessed as though it were a different device; perhaps more importantly, disk files may be accessed as though they were physical devices! Without belaboring the point, let me simply say that a DOS has never, ever attained such flexibility with and independence from its peripherals as may be achieved with DOSPLUS II; this flexibility is useful and usable, not only for the programmer, but for the serious user as well.

Not-So-Basic Enhancements

Rather than write their own BASIC interpreter, the folks at MicroPower, Inc. decided to patch the one supplied with TRSDOS to achieve their objective of a better BASIC. The patch procedure itself is simple, and has been completely automated by a DO file, included on the DOSPLUS II disk. Simply insert your TRSDOS disk into drive 1 and type DO BASIC — that's all there is to it. The patched BASIC interpreter now provides you with some badly-needed goodies. First, you have an additional 4K of user RAM to play with; this area, in high memory, was locked out by TRSDOS to avoid conflicts between itself and BASIC. DOSPLUS II causes no such conflicts and so frees up the memory.

Next, indirect control transfer (named labels) is now supported, a feature that has long been sorely needed in BASIC, and finally, here it is. Indirect addressing lets you refer to the targets of GOTOs and GOSUBs by a name rather than a line number.

```
Example: 10 GOSUB SUBRTN
         20 END
         ...
         ...
         500 NAME SUBRTN
         510 RETURN
```

To achieve this, DOSPLUS II redefines the usage of the BASIC keyword "NAME"; it can no longer be used to rename files on the disk. However, this may still be accomplished by using the SYSTEM"RENAME" command. RENUM, of course, has been altered to ignore the new NAME statement.

For programmers who are tired of typing commands like EDIT, LIST, DELETE, and so on, several "shorthand" commands have been provided. I'll list just a few of them:

; or left arrow	list first program line
/ or right arrow	list last program line
down arrow	list next program line
L", S", and K"	LOAD", SAVE", and KILL"
up arrow	list previous program line
D and E	DELETE and EDIT edit current program line

The other enhancements are in the form of utilities which may be called from BASIC using the SYSTEM command. They are SYSTEM"REF", which does a complete cross-reference on the program currently in memory, SYSTEM"SR" which does a global search-and-replace(!), and SYSTEM"SORT" which does... well, you can figure that one out. Without getting into a discussion of the various parameters and options allowed with these utilities, we can say that they are quite powerful, and very, very handy.

A Few Tests

DOSPLUS II operates very much like TRSDOS in its simpler functions, however, TRSDOS is left in the dust in terms of power and speed. As an example, my copy of TRSDOS 2.0a takes 21 seconds to boot up from door-close to date question; DOSPLUS takes only six seconds to get to the same point.

To further check DOSPLUS II's speed against that of TRSDOS, I performed seven tests designed to measure certain features more-or-less equally. Four of these were executed from BASIC, while the other three were performed at the operating system level. Let's look at the BASIC tests first.

I wrote a short BASIC program to create a random-access disk file of 100 256-byte records. Under TRSDOS BASIC this program took 1:04 to build the file; under DOSPLUS II BASIC, it ran in 23 seconds.

A second program copied each record from the original file to a second, new file using a FOR-NEXT loop of 100 GETs, LSETs, and PUTs. In BASIC under TRSDOS, this took 1:45; under DOSPLUS II, it took 1:15.

The other two tests were SAVEing and LOADing that second (four line) program: TRSDOS BASIC took 11 seconds to SAVE it (try that — it seems like forever), and 5 seconds to LOAD it; DOSPLUS II BASIC took 3 seconds and 2 seconds, respectively.

The first of the DOS tests used the LIST feature of each system to display the file created by the above programs on the screen. Uninterrupted, TRSDOS finished in 1:42; DOSPLUS II was through in 39 seconds.

The remaining tests were two things we all do, all the time: FORMAT a blank disk and BACKUP a relatively full disk. TRSDOS's FORMAT took 2:52; DOSPLUS II's FORMAT only took 1:14 — a significant difference. The times for the BACKUP, performed from drive 0 to drive 1, were even more dramatic. TRSDOS took 8:49, DOSPLUS II took 2:20. In other words, the complete FORMAT/BACKUP took over 11½ minutes under TRSDOS; DOSPLUS II did the same pair of functions in just over 3½ minutes. This comparison becomes even more significant when you consider that, due to its more compact diskette organization, DOSPLUS II actually had to copy over 80K more data than TRSDOS did!

Freebies

My motto (at least, one of them) is, "If it's free, I can use it!" In the case of DOSPLUS II, the things that are free are really useful! Included on the disk are an Editor/Assembler for assembly-language programmers called EDAS, which alone retails at about $150.00; DISKZAP, a disk editor similar to SUPERZAP; a pair of directory examination and repair utilities, appropriately named DIRCHECK and DIRFIX; a graphics editor called DRAW; MAP, which displays the segmentation of a file on the disk surface; OFFSET, which relocates a CMD file to a desired spot in RAM; and a utility to transfer files between TRSDOS and DOSPLUS II disks (since they're not directly compatible), called CONV. How's that for freebies?

DOCUMENTATION

One of my pet peeves is good software accompanied by poor documentation. I am happy to report that DOSPLUS II's documentation is not merely good, it is excellent. It is complete, thorough, and literate. The manual is about the size of the Model II Owner's Manual and bound in a roomy three-ring binder. The professional programmer can dig into the manual and find out everything there is to know about the feature in question; at the same time, the novice user can find out only what he wants to know and go on. Syntactical conventions are explained at the outset and adhered to throughout the text, and examples are numerous and pertinent. To avoid confusion, even a great many "simple" concepts which most of us now take for granted are patiently defined and explained. The text is instructional in nature, and the language used is informal and direct, but not frivolous, complete and often quite lengthy, but not verbose.

Summary

The magnitude of this system was a drawback to an effective review. After using and examining DOSPLUS II, I discovered there is no way to do it justice in the space of a magazine review; many very good features failed to find their way into this article. The speed and flexibility of DOSPLUS II is well worth the price, and the quality of both the software and the documentation is excellent. For the Model II/12/16 programmer or serious user who is currently using TRSDOS, DOSPLUS II is, in my opinion, a sound investment in computing.

DOSPLUS II is available from
Micro Systems Software, Inc.
4301-18 Oak Circle
Boca Raton, FL 33431, and
PowerSoft
11500 Stemmons Freeway, Suite 125
Dallas, TX 75229
Cost $249.95

Printing with TRSDOS 2.0 on the EPSON MX

Paul Margus
54 Crane Road
Mountain Lakes, NJ 07046

Most printers react to not only the conventional set of printable characters but also to a special set of characters called control characters.

Control Characters

Control characters include form feeds (jump to next page), line feeds, tabs, and other useful printer functions. Instead of getting printed, a control character directs the printer to perform the indicated function. One of the distinguishing features of the fancier printers is the variety of control characters to which they respond.

The standard ASCII bit code sets aside hexadecimal characters X'00' through X'1F' (CHR$(0) through CHR$(31) in BASIC) as control characters. Hexadecimal X'20' through X'7F' (CHR$(32) through CHR$(127) in BASIC) are the normal printable characters, including punctuation, upper and lower case, and numerical digits. The remaining 128 characters, running from X'80' through X'FF', are not standardized for printing. Control characters and regular printable characters can be sent to the printer using a normal output statement (e.g. LPRINT in BASIC, or WRITE in FORTRAN).

Until recently, printers were not equipped with the many special features that are now widely available. Even for very elementary functions, the older printers relied on the computer's operating system or the programmer's own software. The TRSDOS 2.0 operating system can accommodate such printers.

The TRSDOS Print Routine

For example, the TRSDOS print routine may or may not send a line feed control character to the printer after each carriage return character, depending on the option specified in the FORMS command. Some printers do this automatically, while others treat carriage return and line feed as separate and distinct. TRSDOS can handle either case. This is analogous to the situation in a conventional typewriter, which may go automatically to the next line whenever the carriage is returned, depending on the setting of the line-spacing lever.

In addition, the TRSDOS print routine keeps track of how many lines have been printed on the current page. In order for the printer to move on to a new page, a program must send it a form feed control character. TRSDOS recognizes the form feed as a special character, and intercepts it instead of passing it on to the printer. TRSDOS sends the printer one carriage return character (with or without automatic line feed) for each line remaining on the page. It "knows" how many lines are left on the page as it maintains a count of the lines already printed. This is ideal for those printers which cannot respond to a form feed.

Similarly, TRSDOS intercepts a tab character. Instead of one tab, the printer receives from 1 to 8 blanks, exactly enough to reach the next programmed tab stop. Under this rather inflexible arrangement, tab stops are set at every eighth column, starting with 9, 17, 25, etc. SCRIPSIT, which operates under TRSDOS, permits tab stops wherever desired, but the printer still receives only blanks, meticulously counted by SCRIPSIT.

How the TRSDOS print routine sends multiple carriage returns or blanks in lieu of the intercepted control character is interesting to note. It loops the required number of times, calling itself each pass through the loop. In normal applications, this is a very unusual technique, and it's certainly hard to follow without documentation!

If a line is too long, TRSDOS sends the printer a carriage return character. This requires the operating system to keep a character count, which is incremented whenever a character is printed. If a backspace character is sent, the character count is also decremented twice, resulting in a net decrement of one (Note that this feature works only if TRSDOS is patched as recommended by Radio Shack. Simply type DO PRTBKSP).

The Radio Shack Daisy Wheel II functions very well under TRSDOS. In addition to a very good selection of printable characters, this printer recognizes control characters for spacing between lines and characters, backspacing, carriage return (with or without automatic line feed), reverse line feed (up one line), and underscoring. But, it ignores many other control characters, including tabs and form feeds. Thus, the Daisy Wheel II, like many other printers, relies heavily on the operating system or other software to tell it what to do. Today's dot matrix printers can produce a variety of typestyles, such as italics, double width, compressed, superscripts, subscripts, or combinations such as double width italics. By sending the appropriate combination of control characters, the programmer can set tab stops anywhere; when the printer receives a tab character, the printer will jump to the next stop. These printers have special graphics characters. In principle, it is possible to print arbitrary shapes or even newspaper-quality photos by programming the firing of each pin in the print head. Best of all, the printer will jump to the top of the next page when it receives a form feed control character since the page length has been set by other control characters.

Thus, when a fancy printer is used with TRSDOS, certain capabilities of the printer are duplicated by the operating system. The TRSDOS print routine is called for each and every character sent to the printer. Each character is screened by TRSDOS, whether it originates from BASIC or any other source. This is usually no problem, but for some operations, TRSDOS gets in the way.

A Solution to an Unexpected Problem
Fortunately, the FORMS command in TRSDOS allows for a "transparent mode". When this option is in effect, every character, whether it's printable or not, is sent to the printer without interference. At least that is what I thought when I started using the Epson MX dot matrix printer. True, the instructions mentioned that the Model I could not transmit the NUL character (hexadecimal double-zero, denoted by X'00', or CHR$(0) in BASIC), unless the programmer poked it directly into a special memory location used by the built-in ROM output routines. However, I got the impression that there was no such problem with Model IIs nor with Model 16s operating in Model II mode.

Unfortunately, my results suggested that our Model 16 was indeed intercepting the important NUL control character. The following sequence of control characters should trigger superscript printing: X'1B' (decimal 27, the ASCII code for "escape"), X'53' (decimal 83, which represents upper-case "S"), X'00' (the NUL character). To print "EPSON", followed by "(TM)" superscripted, Epson's instructions suggest the following sequence in BASIC:

LPRINT "EPSON" CHR$(27) CHR$(83) CHR$(0) "(TM)" or equivalently:

LPRINT "EPSON" CHR$(27) "S" CHR$(0) "(TM)"

Instead, the Epson printed "EPSON", followed by "TM)" *subscripted.* As it turns out, if the third control character is anything other than X'00', the printer goes into the subscripting state. Before it could be sent to the printer, CHR$(0) was intercepted, and the printer interpreted the next available character as a control character; that was the fate of the missing "(".

I ran the FORTRAN equivalent and got identical results, and fared no better using SCRIPSIT. Three radically different approaches were running into exactly the same problem, apparently precluding the use of numerous useful features of the Epson printer.

While frustrating, the situation was not completely hopeless. Unlike the Model I, our Model 16 switches out all built-in ROM outines once the power-up sequence is complete; it subsequently relies on operating system software (whether TRSDOS or CP/M). A Model I hardware problem might well be merely a software problem on the Model II.

This proved to be the case. An examination of the TRSDOS print routine showed that Radio Shack, for some reason, had remained consistent with the design philosophy of the Model I. The NUL character will never be transmitted to the printer, even if the transparent mode is in effect. As written at Radio Shack, TRSDOS is never transparent to the NUL character.

Once the cause of the problem was found, the solution was clear. The following PATCH, entered from TRSDOS READY, did the trick:

```
PATCH SYSRES/SYS A=10FC F=C8 C=00
```

Of course, this PATCH won't become effective until the computer is reset. In addition, it should be done on every system disk for which the feature will be needed. The change will be effective on every backup diskette subsequently produced.

I don't know why Radio Shack has singled out the NUL character in this way. Undoubtedly, all Radio Shack printers get by without it; but other printers, such as the Epson, require it for one or more of their features. If your printer is in the latter category, the PATCH is probably appropriate.

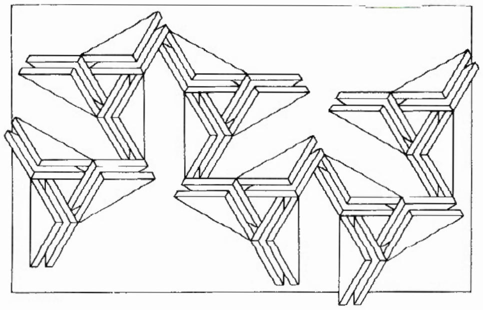
On the other hand, if you expect to use only Radio Shack printers, the PATCH is definitely not necessary. I haven't tested the reaction of any Radio Shack printer to the NUL character. However, the PATCH will have no effect unless the application program actually attempts to send this special character. If your system has more than one printer, you may want to get the most out of each one, in which case, the PATCH will probably do no harm.

Application to SCRIPSIT

In order to use our Epson MX printer with SCRIPSIT, I have established some SCRIPSIT Printer Control Codes, following the procedure outlined on page 66 of the SCRIPSIT Reference Manual. For a list of control characters, check the instructions for your printer.* If you would like to try the experiment suggested below, use the hexadecimal codes shown in Table 1, (see page 40) or the equivalent control characters for your printer. Ultimately, you might want to choose different features.

Note that Control Codes Q, S, V, and X contain the X'00' control character. Without the recommended PATCH, we cannot use the Superscript feature, nor can we turn off Unidirectional Printing, Underlining, or Double Width.

As explained in the SCRIPSIT instructions, you can send the appropriate sequence of control characters to the printer by typing "Control X" (denoted by "x"), followed by the appropriate letter. For example, to turn on the double width feature, type "x", followed by "W". The following demonstrates many of these features. First, open a new SCRIPSIT document, set the margins at 5 and 45, and type the following.



xZStart with NORMAL TYPE. Now, consider using xEEMPHASIZEDxF or xGDOUBLE STRIKExH type, or xRSUBscriptingxT and xSSUPERscriptingxT. xCCOMPRESSED TYPE for contrastxD. xWSwitch to DOUBLE WIDTH, xCand then COMPRESSED DOUBLE WIDTHxDxX. xITry out ITALICSxC and COMPRESSED ITALICSxD. xWThen on to DOUBLE WIDTH ITALICS.xJxX xUUNDERLINE everything in sight. xIxWHow about UNDERLINED DOUBLE WIDTH ITALICS!? xCThen there's COMPRESSED UNDERLINED DOUBLE WIDTH ITALICS!!xDxJxVxW Finally, back to NORMAL.

BackspacexYxYxYxYxYxYxYxYxY/////////

```
xKLine spacing at 7/72 in.
Line spacing at 7/72 in.
Line spacing at 7/72 in.
Line spacing at 7/72 in.
Line spacing at 7/72 in.
xLLine spacing at 1/8 in.
Line spacing at 1/8 in.
Line spacing at 1/8 in.
Line spacing at 1/8 in.
Line spacing at 1/8 in.
xMLine spacing at 1/6 in.
Line spacing at 1/6 in.
Line spacing at 1/6 in.
Line spacing at 1/6 in.
Line spacing at 1/6 in.
```

Now, using the SCRIPSIT print utility, print the document. If you are using an Epson printer and have not made the recommended PATCH, the X'00' control character will never make it to the printer, and the output will look like Figure 1 (see page 40). Otherwise, if everything is working well, it will look like Figure 2 (see page 40).

Of course, the format of this output is not really pleasing, since I didn't allow for different sizes of type. When you use a mixture of typestyles, your output will not always appear as expected. SCRIPSIT's line centering feature may not be useful here. To achieve good results, some trial and error may be necessary.

The "Paper Out" Sensor

In order to prevent accidental printing on the rubber roller, many printers are equipped with a "Paper Out" sensor. On the Epson MX printer, it is located on the left side, near the paper entrance. When the sensor detects the "Paper Out" condition, the Epson MX stops printing, sounds a buzzer for a few seconds, and sends a signal to the computer. As indicated in the instructions for the Radio Shack Daisy Wheel II, this printer has no "Paper Out" sensor, and always signals the computer that it has paper.

Each time a character is to be sent to the printer, the TRSDOS print routine checks the status of the printer for "Paper Out" and other error conditions. If there is a problem, TRSDOS does not transmit the character. Instead, it flags the error. Thus, when the Epson runs out of paper, there are two obstacles to any further printing: TRSDOS refuses to send any more characters, and the printer wouldn't operate even if a character were sent.

Occasionally, we may want to feed single sheets through the printer. Perhaps we are using company stationery. In order to print near the bottom of the page, we must disable the "Paper Out" sensor. As usual, the Epson MX responds to a sequence of control characters. In BASIC, simply type LPRINT CHR$(27) "8" (not CHR$(8), a frequent misprint in the instructions. Appendix B gets it right). This disables the buzzer (but strangely, not the "Paper Out" message, which is still sent to the computer). The printer will now be willing to continue.

To try out this feature, enter the following BASIC program and RUN (Note that CHR$(56) is equivalent to "8").

```
10 LPRINT CHR$(27) CHR$(56)   :'Ignore "Paper Out"
15                            :' Sensor.
20 FOR I = 1 TO 70
30 LPRINT I                   :'Print right off th
35                            :' end of the pape
40 NEXT I
```

Because the buzzer on the Epson Printer doesn't sound, we can be fairly sure that line 10 is doing its job. TRSDOS produces

*For the Epson MX Printer, see Appendix B (On the middle of the second page, there are some typographical errors: X'5B', X'5D', X'5E', X'5F', and X'61' should read X'51', X'53', X'54', X'55', and X'57').

the error message. Having received the "Paper Out" message from the Epson, it flags an error, and sends nothing more to the Epson.

According to Epson's instructions, this is another problem unique to Radio Shack Computers. Epson suggests disconnecting pin 12 of the printer cable, preventing any "Paper Out" message from getting to the computer. This is not a very elegant solution, especially if we want to keep our options open. Once again, a Model I hardware problem is a software problem on the Model II. The following PATCH will remedy the problem.

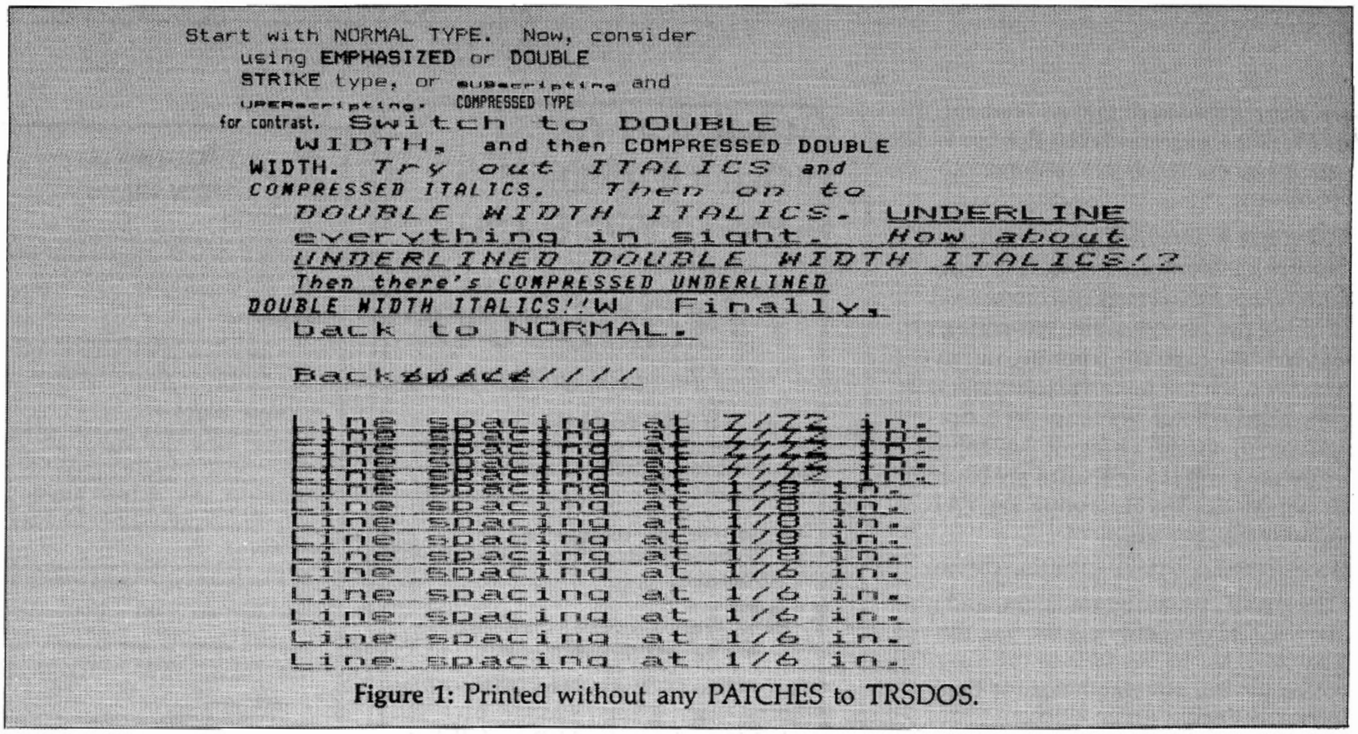
```
PATCH SYSRES/SYS A=1185 F=2012 C=0000
```

RUN the BASIC program again, and note that the PATCH definitely works. You can print as close to the bottom of the page as you please, but watch out for the bare rubber roller.

However, I cannot explain what happens under SCRIPSIT. According to Table 1, xB should send the right characters to the printer. When the printer nears the bottom of a single sheet, SCRIPSIT shows a "Printer not ready" message. Because the printer doesn't sound the buzzer, the problem is somewhere in the software. Perhaps SCRIPSIT should be examined. I would be grateful to hear from anyone who can explain this problem.

Conclusion

If you are using TRSDOS with a non-Radio Shack printer, I hope that this article is useful to you. While certain control characters differ among printers, the principles are always the same. Naturally, if you are getting what you need from your printer, the two PATCHES presented here are not necessary. Also, since they have not been tested under every conceivable situation, I would be interested to know about any of your experiences with them.

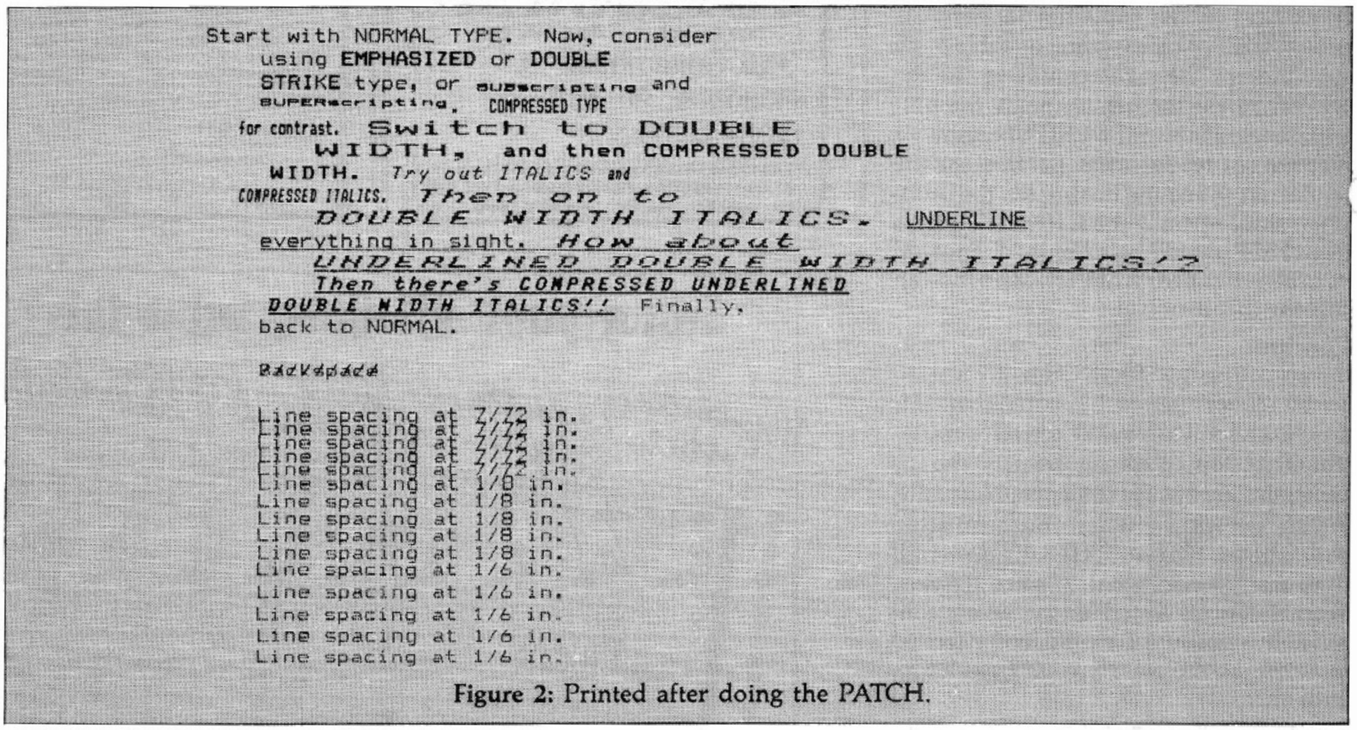


```
Start with NORMAL TYPE.  Now, consider
    using EMPHASIZED or DOUBLE
    STRIKE type, or subscripting and
    UPERscripting.  COMPRESSED TYPE
  for contrast. Switch to DOUBLE
      WIDTH. and then COMPRESSED DOUBLE
    WIDTH. Try out ITALICS and
    COMPRESSED ITALICS.  Then on to
      DOUBLE WIDTH ITALICS. UNDERLINE
      everything in sight. How about
      UNDERLINED DOUBLE WIDTH ITALICS!?
      Then there's COMPRESSED UNDERLINED
    DOUBLE WIDTH ITALICS!!W  Finally,
      back to NORMAL.

      Back~~space~~////

      Line spacing at 7/72 in.
      Line spacing at 7/72 in.
      Line spacing at 7/72 in.
      Line spacing at 7/72 in.
      Line spacing at 7/72 in.
      Line spacing at 1/8 in.
      Line spacing at 1/8 in.
      Line spacing at 1/8 in.
      Line spacing at 1/8 in.
      Line spacing at 1/8 in.
      Line spacing at 1/6 in.
      Line spacing at 1/6 in.
      Line spacing at 1/6 in.
      Line spacing at 1/6 in.
      Line spacing at 1/6 in.
```

Figure 1: Printed without any PATCHES to TRSDOS.



```
Start with NORMAL TYPE.  Now, consider
    using EMPHASIZED or DOUBLE
    STRIKE type, or subscripting and
    SUPERscripting, COMPRESSED TYPE
  for contrast. Switch to DOUBLE
      WIDTH. and then COMPRESSED DOUBLE
    WIDTH. Try out ITALICS and
  COMPRESSED ITALICS. Then on to
      DOUBLE WIDTH ITALICS. UNDERLINE
  everything in sight. How about
      UNDERLINED DOUBLE WIDTH ITALICS!?
      Then there's COMPRESSED UNDERLINED
    DOUBLE WIDTH ITALICS!! Finally,
  back to NORMAL.

  ~~BacKspace~~

  Line spacing at 7/72 in.
  Line spacing at 7/72 in.
  Line spacing at 7/72 in.
  Line spacing at 7/72 in.
  Line spacing at 7/72 in.
  Line spacing at 1/8 in.
  Line spacing at 1/8 in.
  Line spacing at 1/8 in.
  Line spacing at 1/8 in.
  Line spacing at 1/8 in.
  Line spacing at 1/6 in.
  Line spacing at 1/6 in.
  Line spacing at 1/6 in.
  Line spacing at 1/6 in.
  Line spacing at 1/6 in.
```

Figure 2: Printed after doing the PATCH.

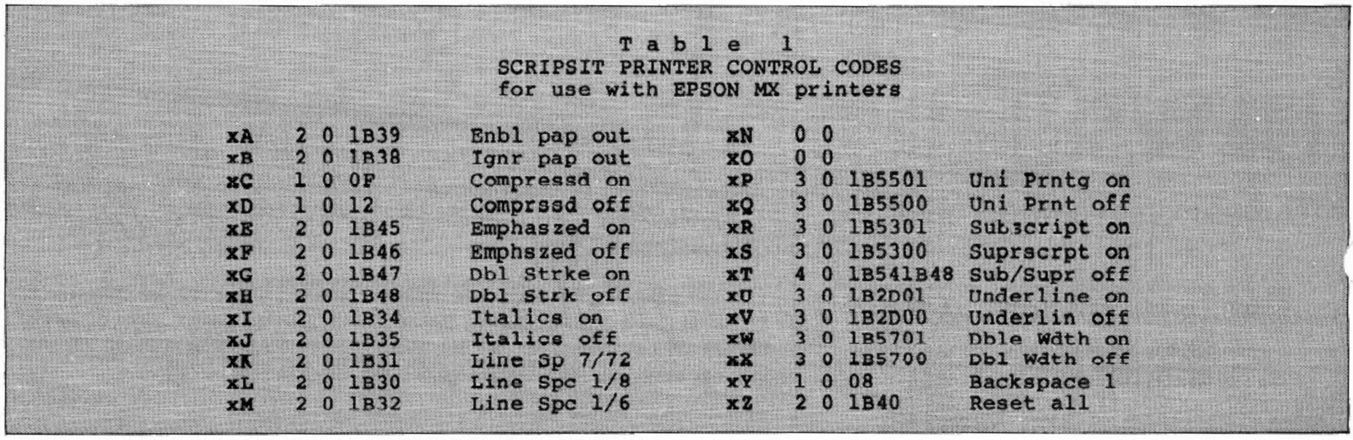


Table 1
SCRIPSIT PRINTER CONTROL CODES
for use with EPSON MX printers

xA	2	0	1B39	Enbl pap out	xN	0	0		
xB	2	0	1B38	Ignr pap out	xO	0	0		
xC	1	0	0F	Compressd on	xP	3	0	1B5501	Uni Prntg on
xD	1	0	12	Comprssd off	xQ	3	0	1B5500	Uni Prnt off
xE	2	0	1B45	Emphaszed on	xR	3	0	1B5301	Subscript on
xF	2	0	1B46	Emphszed off	xS	3	0	1B5300	Suprscrpt on
xG	2	0	1B47	Dbl Strke on	xT	4	0	1B541B48	Sub/Supr off
xH	2	0	1B48	Dbl Strk off	xU	3	0	1B2D01	Underline on
xI	2	0	1B34	Italics on	xV	3	0	1B2D00	Underlin off
xJ	2	0	1B35	Italics off	xW	3	0	1B5701	Dble Wdth on
xK	2	0	1B31	Line Sp 7/72	xX	3	0	1B5700	Dbl Wdth off
xL	2	0	1B30	Line Spc 1/8	xY	1	0	08	Backspace 1
xM	2	0	1B32	Line Spc 1/6	xZ	2	0	1B40	Reset all

FROM TRSDOS TO CP/M AND BACK AGAIN

Roger Conant
1108 SEO
University of Illinois @ Chicago
Box 4348
Chicago ILL 60680

In the inaugural issue of *two/sixteen* (May/June 1982, p. 17), a note was published, asking what do you do if you want to transfer information from CP/M to TRSDOS. Indeed, there is a similar problem in transfers from TRSDOS to CP/M, unless you are fortunate enough to have the program TRS2CPM.COM to do it for you.

It turns out that you don't need anyone's program to do these transfers. You can do them yourself using only TERMINAL on the TRSDOS side and DDT and SAVE on the CP/M side. The transfer is a little tricky, though, and therefore is not convenient for transfers of many files. But if you do not perform transfers frequently, you can get along with the information in this article.

A Valuable Patch to TERMINAL

The TRSDOS Manual suggests that you create a personalized version of TERMINAL and shows you how to do it. This is *required* for the transfer methods to be described.

Type

```
TERMINAL
```

to load the terminal program. In response to the menu, type

```
<S>
```

When asked, enter this TRSDOS command:

```
DUMP TERM {START=3000 END=4219}
```

which saves everything up to the RAM buffer used by TERMINAL.

Exit the TERMINAL program by typing <Q>

Then perform this patch on your file TERM:

```
PATCH TERM A=3061 F=39 C=24
```

This patch defeats the part of TERM which clears user memory just before quitting. Clearing memory was a bad idea anyway since often executing a System command from within TERM (using the S command) dumps you out into TRSDOS READY, and any data stored in the RAM buffer is then lost. With this patch installed the information stays there, although if you enter TERM by typing TERM <enter> at TRSDOS READY and then D (for "display RAM buffer"), the buffer will appear to be empty. The reason is that when TERM is reloaded, the pointer (at 3774 hex) to the end of the RAM buffer is re-set to the start of the buffer.

So, if you have data in the buffer but find yourself dumped out of TERM, type the following (let "*" stand for <enter>):

DEBUG ON*	remember, * = <enter>
DEBUG*	
M 3774	the contents of cells 3774 and 3775 give the address (inverted) just past the end of the buffer. Write them down. Say they are "YZWX".
<Escape key>	get out of memory mode of DEBUG
S	leave DEBUG
TERM*	load TERM and re-enter DEBUG
M 3774	note that the contents are not what you wrote down (unless the buffer was originally empty)
<F1 key>	jump up to the 3774 location
YZWX	where these are the contents you wrote down
<F2 key>	save the contents
C	and run the TERM program

Now you should be back in TERM with the data in the RAM buffer once more available to you. You can display it with the D command, save it to disk with the C command, etc. The procedure above simply reveals, before it is too late, where the end of the buffer is, and then puts this information into the right place in TERM.

Moving Material from CP/M to TRSDOS

If you can recover RAM information left there by TERM, you can also recover RAM information left there by CP/M, and that is the idea. To transfer things from CP/M to TRSDOS, you just load them into the RAM area corresponding to the buffer of TERM, leave CP/M, boot up TRSDOS, and enter TERM by the procedure above, sticking the proper address into locations 3774-5. Then you can display the information, save it into a disk file, etc. using the facilities of TERM.

From here on, "*" will stand for "Enter" and <...> will stand for whatever key is named inside the brackets.

Suppose you are in CP/M and have some text or data in a file called CPMSTUFF. To get this material loaded into RAM you type

DDT*	invoke DDT, the debugger
F421A,A000,0*	clear memory from 4100 to, say A000—at any rate, sufficiently to hold the data but not enough to destroy CP/M
ICPMSTUFF*	tell DDT that the input file is CPMSTUFF
R421A*	read the data in, starting at 421A hex

Hex 421A is the start of the RAM buffer of TERM. DDT will give you the appropriate ending address, as "next" or something to that effect, e.g., "JKLM". Unfortunately, some (but not all) versions of CP/M bring in data in 256 byte chunks so that the ending address JKLM may be beyond the end of the data. If yours does, you have to find the actual end yourself by scanning RAM.

Subtract 100 from JKLM to get the address of the last chunk and then type

Dnewaddress*	for example if JKLM=551A type D541A*

and note from this display the address which is one cell beyond the end of the data. (You may have to type D* to see the tail end of the page.) Call this address JABC.

Leave CP/M by removing the disk, and put in the TRSDOS disk and push "reset" to reboot. After the initialization, type

```
DEBUG ON*
TERM*
M 3774
<F1 key>
BCJA                the address you found above, but inverted
<F2 key>
C
```

Now you are in TERM, and you will find the CPMSTUFF information in the RAM buffer. You can display or store it as usual.

Moving Material from TRSDOS to CP/M
Some versions of CP/M (e.g., Pickles & Trout) have a built-in program, TRS2CPM, which performs this transfer for you. If yours does not, the following procedure works at least for Lifeboat CP/M (but is untested in others).

TERM, the terminal program, is the key again. In your TERM program, use the G command to get your file loaded into the RAM buffer, then the Q command to quit. Then type

```
DEBUG ON*
DEBUG*
M 3774              and note the inverted address there—"LMJK",
                    say; the true address is JKLM
<escape>
O                   to turn off DEBUG.
```

Now remove the TRSDOS disk, insert the CP/M disk, boot up, and type

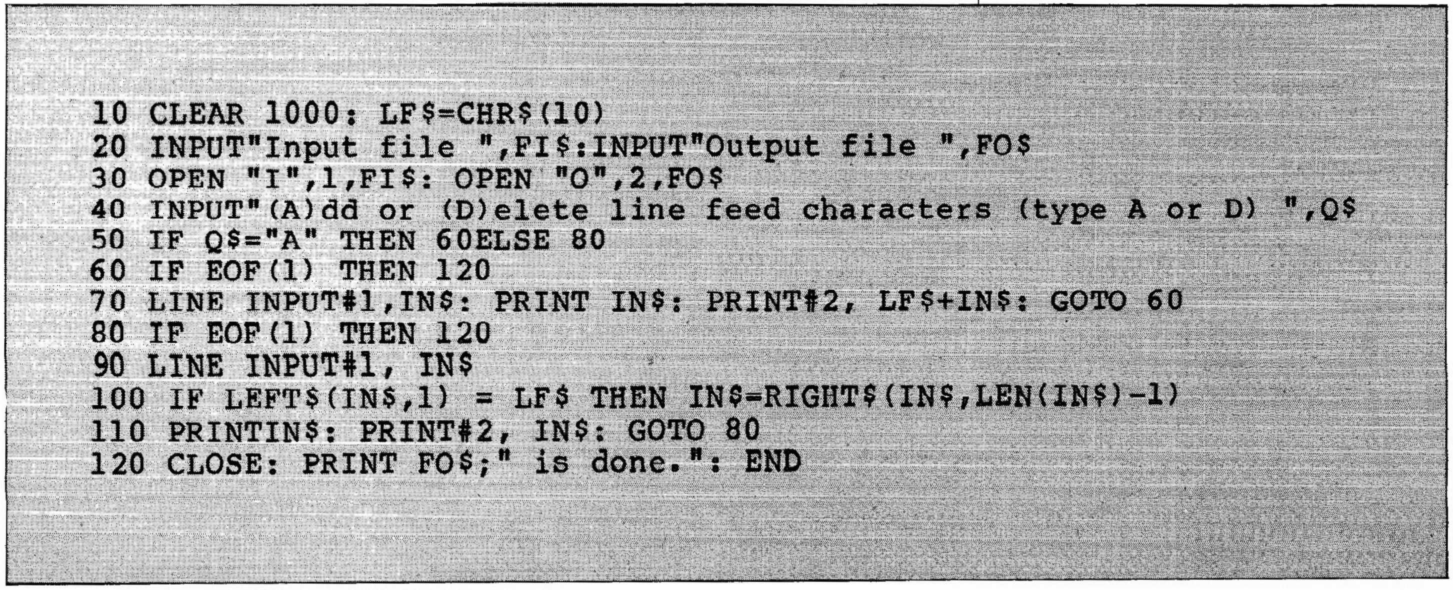
```
DDT*
D421A,JKLM*         this step is optional—it displays the data in
                    the RAM buffer. JKLM is the address you got
                    from DEBUG
M421A,JKLM,0100*
                    moves the file lower in memory in
                    preparation for saving it
G0*                 gee-zero exits DDT
SAVE n filename*    where "filename" is the name to be
                    assigned to the CP/M file
```

In the last command, n is the number of 256-byte chunks ("pages") to be saved. To calculate this number, subtract 421A hex from JKLM hex, take the first two digits, add 1, and convert to decimal.

For example: suppose DEBUG shows 5F7E as LMJK, indicating an address JKLM = 7E5F. Subtract 421A to get 3C45, the length of the file in hex. Take the first two digits and add 1 to get 3D, the number of pages to be saved. Converting to decimal, the correct command is "SAVE 61 filename*".

A Minor TRSDOS-CP/M Incompatibility and its Solution
If you move text files from CP/M to TRSDOS and then display them in TRSDOS, they will appear to have extra line feeds at the end of every line. Worse, if you move text files from TRSDOS to CP/M and then display them in CP/M, they will appear to have NO line feeds, so that every line prints on top of every other. This is because the TRSDOS convention is to put 0D hex ("carriage return") at the end of lines, and let the printer or screen display do the line feed automatically, while the CP/M convention is to put 0D0A hex ("carriage return, line feed") at the end and NOT have the display do any line feeds unless requested. Some TRS2CPM programs take care of this problem for you when you are transferring into CP/M. If yours does not, and in any case if you are transferring into TRSDOS, this incompatibility can be removed, on the TRSDOS side, by a simple BASIC program which either inserts 0A hex on departing files or removes it on arriving files, as needed.

Here is a program to do it:



```
10 CLEAR 1000: LF$=CHR$(10)
20 INPUT"Input file ",FI$:INPUT"Output file ",FO$
30 OPEN "I",1,FI$: OPEN "O",2,FO$
40 INPUT"(A)dd or (D)elete line feed characters (type A or D) ",Q$
50 IF Q$="A" THEN 60ELSE 80
60 IF EOF(1) THEN 120
70 LINE INPUT#1,IN$: PRINT IN$: PRINT#2, LF$+IN$: GOTO 60
80 IF EOF(1) THEN 120
90 LINE INPUT#1, IN$
100 IF LEFT$(IN$,1) = LF$ THEN IN$=RIGHT$(IN$,LEN(IN$)-1)
110 PRINTIN$: PRINT#2, IN$: GOTO 80
120 CLOSE: PRINT FO$;" is done.": END
```

KWIX: A KEYWORD INDEX

Dean R. Lambe, Ph.D.©
Route 1, Northlake Drive
Vincent, Ohio 45784

Did you ever need a table of contents or an index for a long report or book-length project? Perhaps you have needed to find one client's name in one document file that might be on one of several disks. Or maybe you have wondered, as we all should, whether you have over-used jargon or slang terms in your business communication. If so, then Skyline Software's KWIX Key Word Index may be the program for you. While several grammar-checking or proofreading programs that run in CP/M operating systems do provide some of these key word sorting features, KWIX appears to be the only program of its kind that runs in TRSDOS and accesses SCRIPSIT document files. If all your word processing is done on Radio Shack software, KWIX is the only game in town.

Functions of KWIX

As the documentation manual states, KWIX produces a sorted listing of significant words in selected SCRIPSIT documents. Two types of sorts are offered in this menu-driven program: Key Word Only, and Key Word in Context. In both cases, once the user specifies an option in the menu, a pre-programmed file of "non-key words" is loaded — common words such as "the," "and," "new," etc—and all remaining words in the chosen SCRIPSIT document(s) are then compiled and sorted in descending alphabetic and/or numeric order. Up to 100 short documents from a single disk may be indexed together, and with the Key Word Only option, files from up to 100 disks may be combined within the same index. The Key Word Only listing is printed all in capital letters and is followed by the document number and page number for each occurrence of that word, as for example:

Key Word	Reference Doc/Pg
MOTHERHOOD	2/4, 2/6, 3/12
APPLE	1/6
PIE	4/1, 5/8

With the Key Word in Context option, on the other hand, the listing is made up of 80-character lines with the key words in the middle and marked with an asterisk:

```
Doc/Pg * Key Word
1/6 ... those who use a fumble-fingered
          * Apple Computer deserve ...
4/1 ... TRS-80 systems are as easy as * pie to use,
        because the ...
5/8 ... the merits of cobbler over fresh * pie are not to
        be over- ...
```

Only with this In Context option may one actually see the key word as used in a particular document, but for those files on the same disk at least, all uses of a given word are shown. Sixteen appearances of the word "program" in the same three-page document should tell the user something about his creativity.

Word counts for both key and non-key words are displayed during the sort, and the operator then may chose to list the entire index or begin the index from a specified number or word. Three final display modes are available: Printer, Print to Screen, and Print to Screen plus creation of an ASCII disk file.

Using KWIX

The 27-page documentation is very clear and straight-forward. The first page is a "quick reference sheet" and the final page is an appendix with details for conversion of an Index or Context file from ASCII file to a SCRIPSIT document. Once KWIX is installed on a TRSDOS disk (the program arrives on a distribution diskette, and must be put on your own TRSDOS disk with a MOVE command), and the manual has been read once, most people will need only those first and last pages for future operations.

To create an index, one inserts the KWIX disk in Drive 0 and one or more (as called for) disks with SCRIPSIT documents in Drive 1. From TRSDOS READY, either a full alphanumeric, alphabetic only, or alphabetic plus alphanumeric (e.g., Z80) listing may be chosen. A listing of numbers only—as with a ZIP code sort, for example—is not available, but since numbers are listed first in the KWIX F(ull) option, a list of such numbers can be created easily. A count of SCRIPSIT documents is given, and the documents are numbered as they will finally appear on the list. In this regard, if one runs more than one SCRIPSIT disk in a given index procedure, s/he will have to keep track of document numbers, for they will no longer correspond to the numbers within the individual SCRIPSIT disk directories (although the individual document names are preserved and printed at the beginning of each index).

Any or all documents may be chosen for indexing through a small number of keystrokes, the sorting is very fast, and printing options include single or double spacing and 80 or 132 column lines. The Index or Context files are stored on the disk in the drive you select (the SCRIPSIT disk usually lacks space for a long index), and these files may of course be recalled at any time. Beware, however, of the fact that each new sort writes over an existing Index or Context file, so if you want to permanently save your index of all letters to Acme Nut Company, rename that file or keep it on another TRSDOS disk.

Conversion of Index and Context files to SCRIPSIT documents is also easy, and necessary if one wishes to produce a true index of contents. The ASCII file is loaded just like a MYWORDS file for the SCRIPSIT spelling check dictionary, and words can then be added, deleted, or modified in the usual fashion.

Limitations of KWIX

There are a number of problems with this key word index program that are either trifles or disasters, depending on your point of view (and your intended use). The word count is displayed so rapidly that one could blink and miss the number of words in the indexed document(s). And when printing the list to screen only, one has to be very quick with the HOLD key to catch all the words. When a full list of both words and numbers is chosen, the numbers appear first and sorting is by initial digits only, thus documents that contain street addresses, ZIP codes, and phone numbers would yield Key Word Only lists like:

```
16
16TH
169
17603
19101
1981
3743856
```

If only alphabetics are listed, on the other hand, the program has a nasty habit of missing the obvious. "Radio" and "Shack" would be listed separately, just as "Dow" and "Jones" would no longer be a familiar team. And with both numbers and words, the selection process drops a number of often-vital punctuation marks. Embedded hyphens, periods, apostrophes, and commas are removed. "John's" becomes "JOHNS" and "often-vital" degenerates to "OFTENVITAL." Given that both "4,999" and "49.99" fall into the same spot, mathematicians and accountants will have second thoughts about KWIX (although, in fairness, the Context option lacks these difficulties). And the record length is limited to 15 characters, so the writer who is fond of "antidisestablishmentarianism" will come up a bit short—several bits actually. In all cases the user is forewarned, however, and conversion to a SCRIPSIT document provides a ready repair of fractured words and syntax.

I also wish that there were some access to the non-key words file, for a listing of personal pronouns is impossible. If you worry about over-use of "I" and "me," KWIX cannot help you. More importantly, since "new" is considered non-key, your index will never tell you that you have written "New York" and "New Jersey" sixteen times, unless you are quick to spot the odd appearance of "YORK" and "JERSEY" in the printed list.

Once Over KWIX-ly

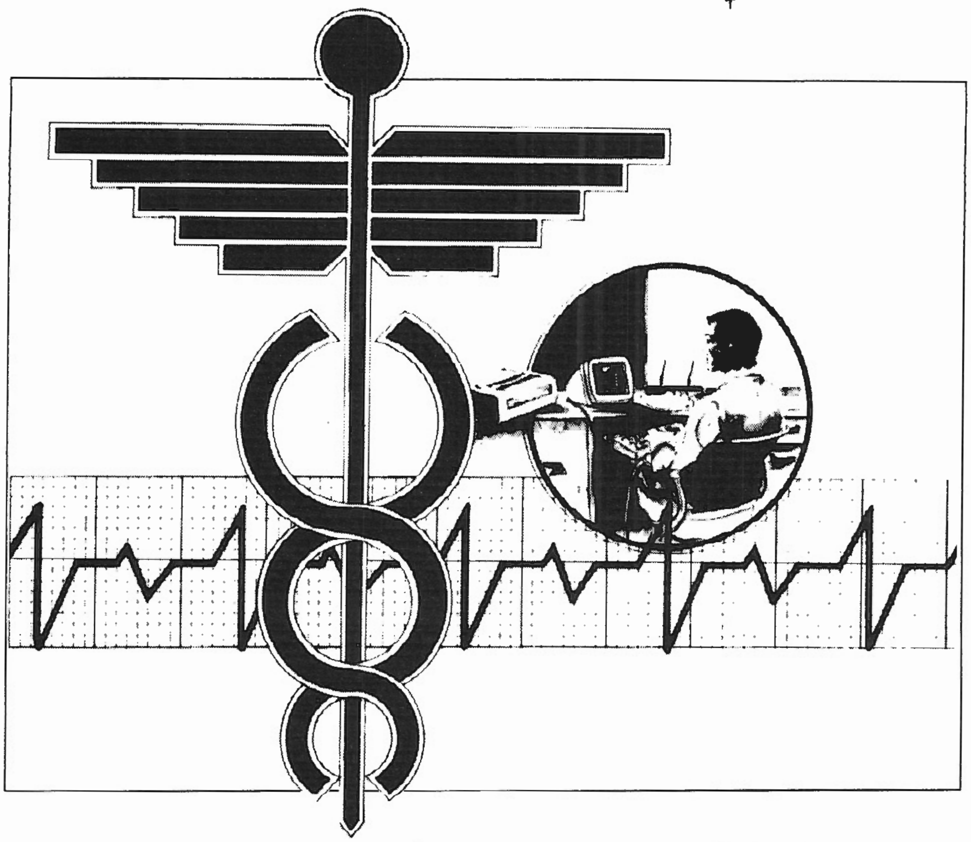
Despite the above limitations, and although considerable additional work would be needed to transform a KWIX Index into an actual ready-for-the-typesetter book index, both the Index and Context functions are fast and useful. The quite reasonable price makes it even more attractive and it is a worthy addition to the TRSDOS-user's library.

As further indication that password protection of SCRIPSIT document files is a joke, if you still needed confirmation, KWIX sails past personal passwords as if they did not exist (which is more than can be said for SCRIPSIT itself).

While my copy of KWIX arrived with a many-legged bug, a call to the developer, Steve Kundzala, received quick attention and a list of 12 patches arrived within three days. I would assume that all disks sold after March 15, 1983 are bug-free, and now anyone can sort their laundry lists to their heart's content.

KWIX ver 1.2 for TRS-80 Model II/12/16 with two disk drives:
Cost $40.
Available by mail from:
Skyline Software
Suite 2411-S
3705 South George Mason Drive
Falls Church, VA 22041

There's not the least thing can be said
or done, but people will talk
and find fault.
—*Cervantes,*
(Don Quixote)



THE MEDICAL MICRO

"The Multi-User Dilemma"

Robert Dale Langer, M.D.
The Center for Health
16766 Bernardo Center Drive
Suite 111-A
San Diego, CA 92128

A recent editorial side-bar in *Info-World* purported to document a remarkable occurrence in microcomputing. A new system, replete with CPU, monitor, printer and cables, plus the usual documentation, was purchased by an inexperienced end-user. This computer novice proceeded to unpack and set-up the system according to the documentation provided, and, about 30 minutes later, powered on the system. Then . . . it worked! There's a moral in there somewhere.

****

I have often been accused of expecting too much from my machines — and perhaps this is true. The fact is that in a typical medical or other busy professional office, there is a mind-boggling amount of information of many different kinds to be managed — a significant percentage of which is at least loosely linked together. For example, an individual patient needs to be billed for services rendered, needs to be scheduled for a return appointment and/or a reminder at a certain interval, needs to have clinical information entered on a chart, needs to have certain treatments or medications ordered and noted, and may have a rare condition which either requires further reading or consultation, and/or which may serve as a useful case for a planned research project or clinical presentation. It seems to me that my computer systems should provide the means to help order such chaos.

Microcomputers are typically seen as providing solutions to individual problems. The business executive may view his computer with its electronic spreadsheet primarily as a forecasting tool, while the attorney may see his machine primarily as a word-processor. Nevertheless, each may run other jobs on the machine occasionally. This kind of usage pattern has tended to make the terms "micro-computing" and "personal computing" synonymous; they're not.

Machines such as the Model II and the newer members of its family, the Model 16 and the Model 12, which are designed for use in business and professional environments, are a different breed than Apple IIs and TRS-80 Model IIIs. That's hardly news to you, I'm sure, but, in fact, what are the differences and how thoroughly can we exploit them to enable us to make and maintain more complex relationships with our data?

To begin with, so-called business oriented micros like those of the II/16/12 family are designed with faster processors and more memory than most of their less sophisticated counterparts. A second major difference is in the standard mass storage capability, while the third and final major difference is found in the implementation of a standard video display format. Some may argue that higher quality components go into business micros as well, but there seems little evidence to support this claim.

Micros Equal Minis

So what can more RAM, a faster processing speed, increased mass storage, and improved video display characteristics do for us? Theoretically it should take us very close to the capabilities of the traditional minicomputer. In practical terms, one might reasonably assume that these features should permit management of more complex data structures than the garden-variety "personal" microcomputer. And the next logical step should be the capability for organization, storage, and retrieval of the multiple varieties of "loosely-linked" information commonly encountered in professional settings. Realistically, however, this kind of information management requires concurrent access to computer resources by several people who should be updating a shared "data base." My receptionist may be billing the patient and scheduling the next appointment or reminder, while I want to view that or even another patient's clinical record, and my nurse wants to access a treatment plan, as the bookkeeper reviews the accounts receivable, and my partner wishes to find all patients in the practice who are juvenile onset diabetics so that he may pursue a research project!

But a major and very significant difference between micros and minis has to do with the sophisticated memory and peripheral management techniques implicit in minicomputer operating systems which enable multiple tasks for one or several users to be carried on simultaneously. This type of processing is known as "multi-tasking."

While there are some operating systems for micros which purport to offer these functions, their usefulness has been limited by several factors. To begin with, significant degradation of processing speed occurs with most microcomputer chips in multi-user situations which means that operators often experience noticeable delays when using the system. Then it is usually necessary to re-write or otherwise tailor a significant portion of the software which may already be in use to work in a smaller amount of memory and/or with a new operating system. In fact, it is often necessary to start over with entirely new software. And, generally there is less than adequate protection to prevent two users who have simultaneously accessed the same information from updating that information without recognizing the work of the other.

Multi-Processing

A new wrinkle in multi-user systems which has become practical as prices for processor chips have tumbled allocates a separate processor to each user while sharing mass storage and other peripheral devices. This arrangement is termed "multi-processing." Multi-processing systems theoretically should eliminate one of the problems associated with multi-tasking and decrease the significance of another. Specifically, the degradation in processing speed is eliminated (since each user has his own processor), and the necessity of rewriting software should be decreased since in most cases each machine operates as though it were the only one active and has all the usual resources available to it, not the least of which is its own native operating system. However, the problem of simultaneous up-dating of the same information still exists, and, with the present state of the art, some of the expected user memory partition in each computer is actually required by the expanded operating system which manages the shared resources. Both of these problems may necessitate revision of the software, although such revision is often less extensive than that required by a multi-tasking installation.

Host/Terminal Protocol

One very simple solution to the multi-user dilemma for the II/16/12 computers is to establish a host/terminal protocol. In this arrangement the computer is connected to another computer or a "dumb terminal" through an RS-232 port and cable, and a "host" program is loaded on top of the operating system which enables any individual station to access the computer. TRSDOS has such a utility which may be called from the "TRSDOS Ready" prompt once the RS-232 ports have been properly initialized with the SETCOM command. Unfortunately, this routine is customized for communications between two equivalent Tandy computers and will not provide usable formatted screen output on most other computers or terminals. However, appropriate host driver routines may be written, or they may be purchased from reputable software houses such as SNAPPWARE. Using both RS-232 ports on the II/16/12 computers will allow two peripheral workstations plus the main keyboard to send commands to the machine. It is, however, on a first come, first served basis, and only one operation or program can be run at any given time; those at other stations will see the operation being performed by whoever entered a command first, but these would-be operators must wait until the operation in progress (generally a full program routine) is complete to begin using the machine. It is possible for any workstation to "break" the operation in progress prematurely and usurp control; however, if this were to occur with any regularity, no one would ever get any work done! The computer remains, in essence, a single user machine, although it may have up to three workstations attached.

Xenix

The next level of sophistication is that offered by a multi-tasking environment. The Model 16 (or Model 16 upgrade for the II or 12) is designed to offer this capability. Utilizing a faster "16 bit" processor and the Xenix Operating System with up to two "dumb terminals" connected to the RS-232 ports for additional workstations, the Model 16 may very well overcome the problem of processing speed degradation. Xenix, developed by Microsoft, is a derivative of Unix, a system developed at Bell Labs for minicomputer applications and designed to perform the sophisticated memory, mass storage, and peripheral resource management necessary in a multi-tasking environment. Thus, the Model 16 has the potential to function in a manner similar to a minicomputer, with each "dumb terminal," as well as the main computer, acting as if it were a fully independent computer workstation with all system resources and peripherals available to it, and, except in rare circumstances, no waiting time or perceptible slowing of processing speed.

Unfortunately, it appears that software which we have been using on our "8 bit" processors, including the Z-80 "Model II" section of existing Model 16s, will not operate in the Model 16 Xenix environment. Therefore, in switching to the Model 16 under Xenix, established data files and software will either have to be discarded, or, if essentially equivalent software is made available for the "16 bit" processor, the data will have to be converted and transferred to the mass storage format of the new system. Such a conversion might ideally be offered by Tandy for their software, although, to my knowledge, it has not yet been announced and may very well never come to pass. It is highly unlikely that any other software will be convertible in that manner unless sophisticated users take it upon themselves to write conversion routines for which the developers have been gracious enough to supply detailed information regarding file structure. Of course, the option will always exist to continue to run the "8 bit" software on the "8 bit" processor, and this is quite likely what most of us will do—but we then lose the advantages of the "16 bit" multi-user environment. As of the present time, it is also unclear whether or how the Xenix software will protect against the overwriting of the same information by two or more users working on the same file.

Oasis

Oasis is another multi-tasking operating system which can be used on the Tandy II/16/12 computers. Although there is a growing list of applications software for use under this system, there is no cross-compatibility between it and TRSDOS or CP/M. Model II or 12 owners must also add additional RAM to their machines if multiple workstations are desired. While I have seen Oasis demonstrated at several computer shows, I have never actually seen a fully implemented installation in active use. Perhaps a reader who has such an installation would be interested in sharing his or her experience with it.

The other possible solution to the multi-user dilemma is the multi-processing, shared data base arrangement. In this case each operator works at his or her own independent computer which is connected in a cluster arrangement to a shared mass storage device — almost always a "hard disk." Other peripherals such as printers are sometimes also shared but most often are "owned" by the computer to which they are connected. The independent computers are connected through a switch-box or "multiplexor" which acts as a kind of traffic cop to insure that conflicts don't arise between users trying to access the shared resources. In this arrangement processing speed is the same as in a single user system since all processing is done independently by each computer. Also, since the same computer hardware and operating system utilized in a single user environment are used in this configuration, most existing software can be used without change. The only exceptions to such software portability are those programs which cannot operate in a hard disk environment, those which push the limits of available memory (since the networking software needs some space), or those which trespass on the operating system. Most CP/M software meets these portability requirements, as do a wide variety of independent software products written under TRSDOS. Scripsit version 2.0 will not operate with these constraints; however, Profile and Scripsit 1.0, as well as Scripsit HD, apparently will.

Multi-Processing Systems

Several manufacturers and systems houses offer the combination of hardware and software to accomplish such multi-processing with a shared data base. We have recently devoted a considerable amount of time and effort to researching the available systems. Our first contact was with Corvus Systems whose hard disk and multiplexor were represented in their literature as being capable of allowing at least eight independent users with various types of computers using their "native" operating systems to share the Corvus hard disk so long as a separate area of the disk was assigned for each operating system. In other words, all CP/M based computers could access a CP/M area of the disk while all TRS-80 computers could access a TRSDOS area, and all Apples IIs could access an Apple DOS segment. "Semaphores" were also to be available through the multiplexor and associated system software to prevent two or more computers from writing to the same file at the same time.

Upon taking delivery of the Corvus hardware, however, we quickly discovered that only a fraction of these capabilities were actually implemented at the present time. Our intent had been to connect two Model II computers and have a CP/M area on the hard disk for one set of applications and a TRSDOS area for another. However, no support was available for TRSDOS through Corvus — only CP/M was supported by their software. TRSDOS could be used if we wished to purchase CORDOS, a TRSDOS enhancement developed and marketed by an independent vendor, however, if we were to install that system we could not also have a CP/M area on the disk. It also turned out that the "semaphores" were not actually set by the multiplexor software but would be maintained by that software if they were actually set by the application software; in other words, the application software would have to be re-written to use this protection! When, after learning all this, the unit we received also would not power on, we decided that this system was probably not for us, and after some initial resistance, the people at Corvus were reasonable enough to provide us with a refund so that we could obtain other equipment more suited to our needs.

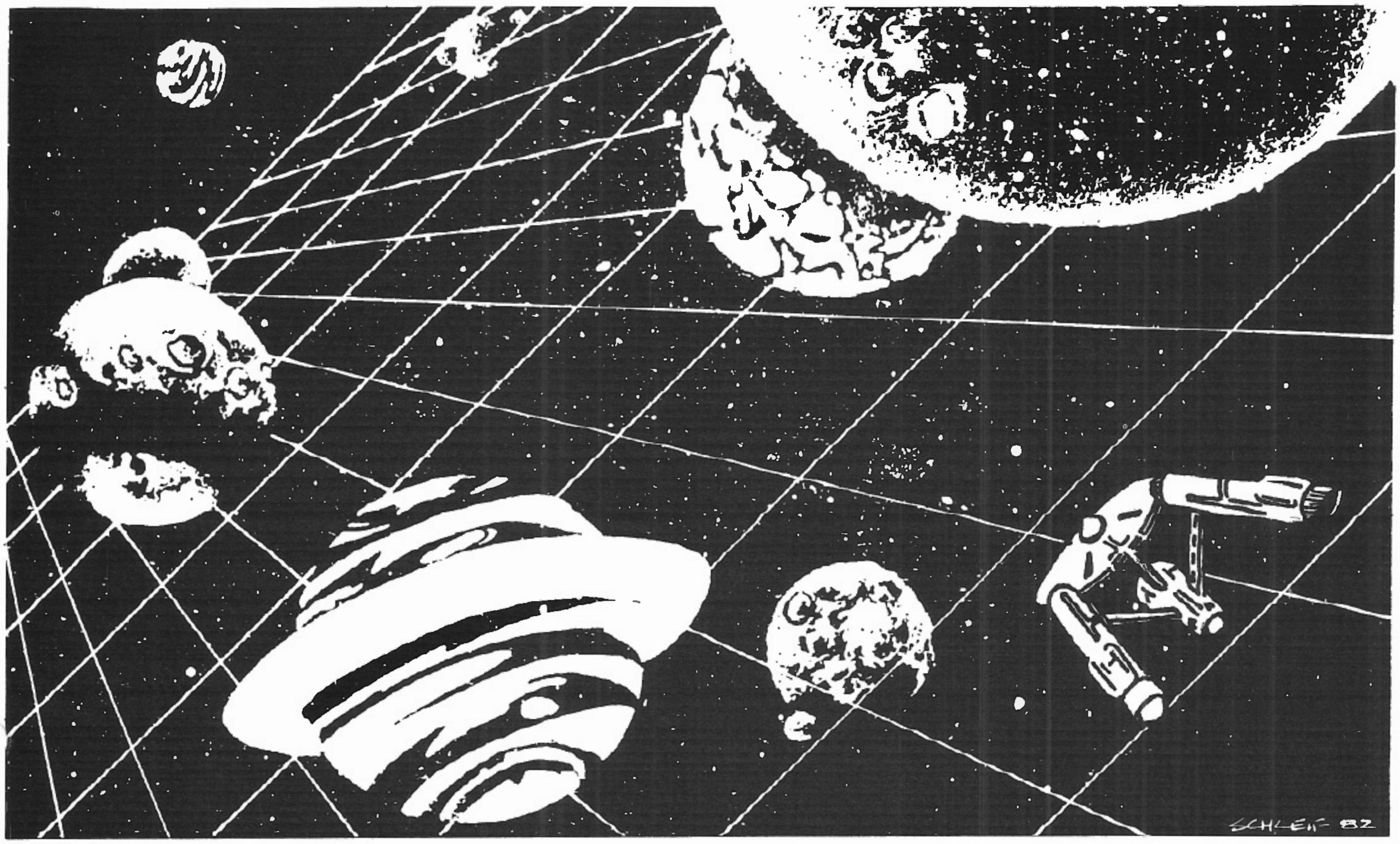
Other suppliers in this market include RACET Computes and Santa Clara Systems. RACET is primarily a software house which has developed an extension to TRSDOS called "Hard/Soft Disk System" (HSDS) which, coupled with their multiplexor, allows up to four users at independent computers to share the same hard disk. ARM, one of the hard disk suppliers to RACET, has been working on software to enable operation under CP/M concurrently with operation under TRSDOS utilizing the same multiplexor and with the same four user capacity. RACET has also developed patches for Scripsit 1.0 and Scripsit HD which allow these programs to be used in their multi-user environment.

We are about to take delivery on the RACET multiplexor system with HSDS for TRSDOS compatibility and the ARM hard disk with CP/M enhancements. We are hopeful that the rather significant time investment we have made in investigating the variety of possible solutions to the multi-user dilemma for the Tandy machines will prove worthwhile.

The rather strange thing about this industry, however, is that the more one understands about the workings of a system, the more questions seem to arise about is ability to do the job. While the newer more sophisticated micros should theoretically be able to support the information management needs associated with concurrent use in a professional environment, we are probably just on the threshold of a practical solution to this problem. And, as with almost everything in the computer world, it is likely that this frontier will not be crossed without some pain.

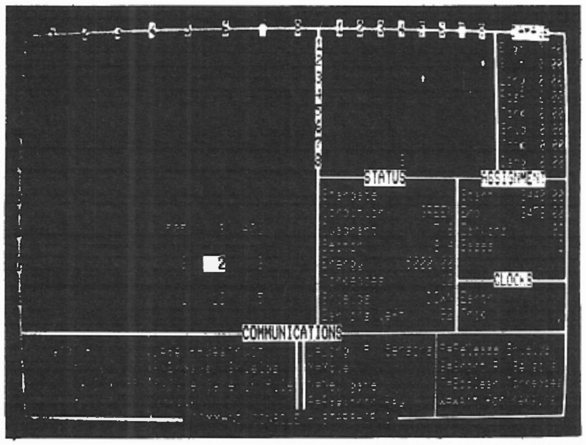
If you have an important point
to make, don't try to be
subtle or clever. Use a pile driver.
Hit the point once. Then come back
and hit it again.
Then hit it a third time—
a tremendous whack.
—Winston Churchill

STARSHIP I



STARSHIP/I (copyright registered) is a save-the-galaxy type game based on the venerable STARTREK games which have been played on large mainframes for more than ten years. It is specifically designed for the RS Model II and features a constant console display with no scrolling. Each game begins in a randomly arranged galaxy of 64 "quadrants," each containing 64 "sectors."

STARSHIP/I is a "strategic" game (as opposed to "tactical" games which involve reflex action but little thought). However, STARSHIP requires *quick* thinking, as all events are timed. Because STARSHIP is written in the RS BASIC interpreter language, users can modify the source code or use sections of the code as models to learn techniques which can be used to produce games of their own design.



starship I

STARSHIP comes with a disk instruction file which can be listed on your printer. It can also be listed on the screen at the beginning of a session. In addition, STARSHIP contains a few undocumented "surprises" at advanced levels of play (there are ten levels of difficulty). These surprises may be analyzed by reviewing the source code, then modified or deactivated if desired.

Games may be saved in files at any point for later play or for "duplicate" games in which several players start with the same layout and position and compete for the highest ending score.

STARSHIP/I was sold for several months at $50 through leading computer magazines. It is now available to **two/sixteen** subscribers at the special price of $30, including air mail (anywhere) in protective Kangaroo mailer.

COMPUTER FURNITURE BY Data-MATE®

INTRODUCTION

DATA-MATE® computer furniture is carefully designed and engineered to provide higher efficiency and productivity. It is far more than simply a support for your computer equipment.

Rugged steel construction is designed for durability. Cable and paper management systems provide efficiency *and* neat, uncluttered appearance. All equipment bays have ample ventilation for heat dissipation. Clean lines blend perfectly with hardware design and function.

Data-MATE® products are in stock for immediate shipment.

Operator comfort is becoming increasingly more important. Features such as height adjustments to reduce operator fatigue and casters for mobility and versatility have been designed into the Data-MATE line. And that means greater efficiency and higher productivity.

Catalog, prices, and further information available from

two/sixteen /magazine (717)397-3364

Excellence in computer furniture technology.