$5.95 U.S.

+ TWELVE

two/sixteen™ magazine

Volume 1, Number 5 March–April 1983

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

> "He that publishes a book runs a very great hazzard — since nothing can be more impossible than to compose one that may secure the approbation of every reader." Cervantes
> DON QUIXOTE

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
Contributing Authors	Arnold Fischthal
	David Gash
	Roger Conant
	Joe Lyddon
	Robert VanNatta
	Jim Kloosterman
Typesetting by	Innovative Ink Lancaster, PA
Printed by	Innovative Ink Lancaster, 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) 397-3364
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..

Hard disk horrors

In spite of previous comments about the future of hard disks (the general tenor being that hard disk prices — per byte — should come down dramatically in the near future), I decided about January 15 to buy one anyway. There were two reasons for this: (1) users who need a hard disk right away should be able to refer to a review of at least one type, and (2) maintenance of our own mailing lists on floppy disks (even double sided) was becoming unmanageable.

We could have opted for the Radio Shack unit, except that we needed larger capacity along with a faster and easier backup method. (Since our large data bases would be updated daily, incremental backup could be almost as lengthy as full backup; using floppies for backup would produce an impractically large inventory of backup disks to be administered).

After scanning a few micro magazines and making several phone calls, I selected the Corvus because of its capacity (20 million bytes), ruggedness, history of reliability with CP/M systems, and "Mirror" backup, which allows full backup on a video cassette in about 35 minutes, or about 500,000 bytes per minute.

Thus begins a series of unhappy experiences which underline the urgent need for standardization and consistency in the micro world.

A call to the Corvus sales department in San Jose resulted in referral to a distributor in Brooklyn, New York. The distributor was very cooperative and volunteered to bring the unit to Lancaster (it's about a three hour drive — this distributor apparently makes such personal deliveries regularly) and install it on the following Sunday morning. In order to meet him, I took an all night flight back from the CP/M '83 conference in San Francisco, but TWA was several hours late and I missed him by a few minutes.

The unit was then shipped, and we received it a few days later. Somehow, the installation manual was omitted, but the steps to physically connect the Corvus to the Model 16 seemed obvious. I installed the board and connected the units, then called the distributor to confirm that everything was correct before powering up. We then "walked through" the software phases of the installation, but without success.

Since the Corvus was a new model, the distributor suspected that I might have received the wrong software, and Corvus was called upon to supply a copy of the latest software to make sure we had the correct version. Due to a shipping mixup, the software was delayed for a week. Then it was shipped via Federal Express to arrive the next morning (Saturday). Due to a misunderstanding, Federal Express missed the delivery, and a call to their 800 number produced the information that the package could be picked up at their office 47 miles away. On arriving at the Federal Express office, the dispatcher informed me that the truck was still in Lancaster, and arranged for me to meet the driver there. A one-hundred mile drive for nothing!

As you can guess, the new software did no more than the old software did. The distributor, now suspecting that we had received a faulty unit (although he had tested it, there was a remote possibility of shipping damage), shipped us a complete second unit.

Several days of testing both units, including numerous modifications to the Corvus link program (CLINK2.ASM) under the direction of one of Corvus' principal engineers, produced limited success (files of up to 16K could be transferred to the Corvus, but larger ones would lock up the system).

It was suggested that the two 68000 boards be removed from the 16 so that it would behave like a II. Voila! Large files could now be transferred to the Corvus. Victory was shortlived. When attempts were made to transfer the data back to floppies, the floppies' directories were overwritten and destroyed.

Finally, the Corvus engineer obtained a Model 16 from elsewhere in his organization. Attempts to successfully install on this machine were also unsuccessful. Further investigation revealed that Model II "ports" which were assigned (by Radio Shack) to perform certain functions for use with non-Shack hard disks have been compromised in the Model 16!

So several weeks were lost and many miseries suffered simply because what had appeared to be a standard was not. Contrary to implications, a 16 is not just a II with more boards in it.

During the period from about twenty years ago to about ten years ago, IBM regularly changed hardware and software interfaces between their computers and peripherals in apparent attempts to thwart the attachment of competitors' devices. This activity contributed to the bringing of the long and costly anti-trust suit against IBM. It also produced the absurd result that competitive hardware could eventually communicate more effectively with IBM gear than other IBM products could (since the competitors were more flexible). IBM seems to have learned that this practice is counterproductive, as it appears to have abated.

Radio Shack has implemented many unusual, if not unique, practices in its hardware and software. Some believe that this has been for the purpose of discouraging (a la 1960s IBM) use of otherwise superior competitive hardware and software products in conjunction with Radio Shack computers. Another explanation (which I prefer to accept) is that the microcomputer business has grown so rapidly that there has not been time for sensible standards to develop, and that Radio Shack (and many others) have simply chosen to decide on the most expedient method available at the time. The many *internal* inconsistencies of TRSDOS (and most other "parochial" operating systems) reinforce this notion.

Tandy's introduction of both UNIX and CP/M indicates that they intend to join the microcomputing mainstream. This is good news, as Tandy has the proven ability to produce the best business microcomputers available. As they consolidate their mainstream position, painful incidents like the "hard disk horrors" will, I hope, no longer occur.

In the meantime, however, this does not mean that a user is necessarily better off to subvert genuine business needs to the notion that all purchases should be from a single manufacturer, be it Tandy, Apple, IBM, or whoever. I suggest that for the best long term results (1) users should *insist absolutely* that software and peripherals be as close to their needs as possible and (2) that such products should be as "standard" as possible.

Users groups

We've recently learned of a Model II, 12, or 16 (only) users group in the Chicago area. It publishes a monthly newsletter and holds a monthly meeting on the second Thursday of each month at Hans Bavarian Lodge in Wheeling, Illinois. It has about 25 members so far, ranging from package-only users to consultants. Interested readers should contact Carlos Hidalgo, 311 Longview Road, Waukegan, Illinois 60087, (312) 623-9661.

As we learn of them, other users groups which specialize in II/12/16 matters will be listed here.

4 Terrific Tools... from World Software Resource

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

It was my pleasure to review four utility programs for the TRS-80 Models II/12/16 from World Software Resource (referred to as WSR throughout this review) of Grinnell, Iowa. The utilities, VARLIST, CREF, RESEQ, and MENU are all machine-language programs, executable from TRSDOS READY, or as a command from BASIC (i.e., SYSTEM "VARLIST"). They are primarily programmer's tools, although MENU could be used by any user with a passing familiarity with computers.

Let's look at the utilities, in order of ascending complexity.

VARLIST — Simple But Useful

VARLIST's function is to list the variables used in a program. While on the surface that may seem a simple task, and a not-too-helpful one at that, read on. Most utilities require that any program to be used as input be saved in ASCII form, which wastes both time and disk space. VARLIST operates, as do all WSR utilities which use a BASIC program as input, on a *compressed* (tokenized) program file rather than an ASCII file.

VARLIST lists the variables, arrays, functions, and USR routines used in a program, and can handle up to 1000 distinct variable names. It does not do a complete cross-reference (see CREF, below), but it does include a feature most cross-referencers (including WSR's) don't — it lists each variable's full name, not just the first two characters. Variables beginning with the same two characters are listed together, allowing the programmer to see unintentional "collisions." Also, a warning message is given when a DEFxxx statement is found following an already-referenced variable.

VARLIST's output is alphabetized and may be routed to the screen, line printer, or a disk file. Variables are shown with their type characters, whether the type is explicitly or implicitly defined in the program itself. Operating speed is about nine statements per second, including print time.

CREF — Cross Referencing And More

CREF is the full cross-reference utility. Although CREF does quite a bit more than VARLIST, it should not be assumed that it makes VARLIST completely useless — quite the opposite, in fact. VARLIST is at its best when used as a compliment to CREF. CREF uses and prints only the first two characters of each variable name, so using VARLIST's output along with CREF's will give the programmer a complete picture of his program's variable structure and usage.

CREF's function is quite comprehensive: it lists the variables, arrays, functions, and USR routines used in a program, what lines of the program they are used in, and some very helpful information about their usage. Output is to a (required) printer and may not be re-routed. Unlike VARLIST, the number of variables in a program being processed by CREF may be unlimited (CREF uses a scratch disk file for intermediate storage, while VARLIST uses an internal array). The output lists the variable name, followed by the line numbers of all lines containing references to that variable. If a single line contains more than one reference to the same variable, that line number is listed once for each reference.

Now for the good part: a variable name or line number may be followed by a "use flag." This is a one-letter code explaining how the variable was used in that particular reference, or in what manner the line itself was referenced. Space does not allow an explanation of all the flags, but the following examples will give you the general idea. A line number followed by "S" means it is called from a GOSUB or ON x GOSUB statement; an "E" indicates it is an ON ERROR GOTO reference; an "L" tells you it was found following ERL and a comparison operator. A variable name followed by "M" indicates a possible modifying reference, such as LET, LSET, or SWAP; a "D" denotes a defining reference, as would be found in a DIM or FIELD statement.

CREF, like VARLIST, will warn you of a DEFxxx statement following a previously-defined variable and contains other warning messages, such as "Line xxx has line reference yyy that is >65529." (That number, of course, is the largest line number allowed in BASIC.) Another message is "Above item might not be used correctly." There are several possible reasons for this, usually traceable to a minor programming error, such as failing to DIM an array. CREF's processing speed is about three statements per second, including print time.

RESEQ — You're Gonna Like This One
RESEQ is not — repeat, NOT — "just another renumber utility." Renumbering is only one of its many features. Properly stated, RESEQ's primary functions are to renumber, reorder, or duplicate blocks of BASIC code, or any combination thereof. Up to thirty blocks may be specified to RESEQ in one operation, so total restructuring of a program with a minimum of effort is quite possible. In fact, RESEQ could easily replace a text editor to create a new program from an existing one containing chunks of usable code. It is actually that versatile.

RESEQ's block specification syntax is simple and logical:

A—B = C + D

where A is the first line number of the block to be operated on, B is the last line number of that block, C is the desired line number of the first line in the new block, and D is the increment by which the lines in the new block are to be numbered. Example: if a block of code numbered 520 through 650 exists, and we wish to renumber it beginning at 1000 in increments of 100, the specification would look like this:

520—650 = 1000 + 100

You may omit any portion of the specification except the "=". A defaults to 0, B defaults to 65529, C defaults to 1000, and D defaults to 10. RESEQ checks for bad specifications and provides you with several messages clearly explaining why a specification was rejected.

In addition, there are a couple of really nifty features in RESEQ which greatly expand its capabilities. One of these is the "shift" function. To use this, the increment (D in he specification) is replaced with the letter "S". During renumbering, this causes the existing increments between line numbers to be maintained.

Let me use a real situation to describe this feature. When writing a disk file maintenance program, I like the "add record" section to begin at line 1000, the "change record" section at line 2000, and the "delete record" section at 3000. If I have those routines in a program, but in sections 4000, 5000, and 6000, I can create the proper blocks in one command, and still maintain my desired sectioning. Say the lines looked like this:

```
4000 This is
4030 the "add"
4570 a record"
4940 section.
5000 This is
5470 "change
5700 a record."
6000 And this is
6250 "delete
6940 a record."
```

A RESEQ specification of 4000—6940 = 1000 + S will return:

```
1000 This is
1030 the "add
1570 a record"
1940 section
2000 This is
2470 "change
2700 a record."
3000 And this is
3250 "delete
3940 a record.
```

Another useful variation is the "copy" function. This is simplicity itself — just put a "C" before any normal block specification, and the block will be renumbered and copied, leaving the original block intact.
Example:
the specification C 100—199 = 2000 (the increment has been omitted and will default to 10) will duplicate lines 100 through 199 at lines 2000, 2010, etc. Copy specifications may overlap move specifications and other copy specifications in any way (move specifications may not overlap other move specifications, however).

A final variation is the "extend" specification. This specification, invoked with a leading "X" allows you to resequence several blocks of lines together into a single destination block beginning with a given line number and incrementing uniformly throughout. Example: we want blocks currently at 100 through 299, 300 through 349, and 10 through 50 to be concatenated in that order, and reinserted into the program beginning at line 1000, in increments of 100. The specifications:

100—299 = 1000 + 100
(move the first block)
X 300—349
(append the next block)
X 10—50
(append the last block)

will accomplish this. The resultant lines will be numbered 1000, 1100, 1200, etc.

Also, "copy" specifications and "extend" specifications may be combined, such as XC 100-199. Obviously, if the programmer is not careful with some of the more powerful commands, he can really foul up his code, right? Wrong! As a safeguard against this, RESEQ renames your original program to "PROGNAME/RSQ" and creates the resequenced file under the original program name, ostensibly "PROGNAME/BAS" . Alternatively, you may specify a name for the output file, say, "PROGNAME/TST" and your original will be unaltered in name or form.

As if this weren't enough, RESEQ even prints a line number conversion table as it runs (if you wish). If your specifications will change the order of the lines in the program, as well as their numbers, RESEQ prints two tables, one in the old order, and one in the new order. Lines which have been created by a copy specification are marked with a "C" in the table. And, during its operation, RESEQ occasionally prints messages like "Reading old line numbers," or "Assigning new line numbers," just to let you know what's going on. Its operation speed is about four statements per second without a line number table, about 1.5 statements per second with a table.

My only complaint with this utility is its input program size limitation — 550 distinct line numbers. Of course, to be fair, writing programs is my job, and I routinely write programs longer than 550 lines. For a part-time programmer, this might not be a limitation at all.

MENU — Versatile and Powerful

On to the last utility, MENU. This program is an altogether different animal from the above three programs, and really isn't a "utility" at all in the sense of operating on some existing program. The primary intended usage of MENU is as a master menu of a BASIC multi-program system, although it is useful as a simple disk menu of non-related programs as well. Its features are varied; it can create, edit, or delete any files on disk or select programs from a previously created menu. Menus may be "locked" by the programmer (denying editing privileges to the user); a menu item may be another (sub-) menu, which may itself contain submenus, etc.

Like WSR's other programs, MENU may be invoked from TRSDOS or from BASIC. The display begins with two lines at the top of the screen which identify the program, specify the drive the menu came from and the location (line number) of the cursor, and give a brief description of what MENU will allow you to do at the moment. The remaining 22 lines show the first part of the menu, read from the master menu file, called MENU/DAT, created with the editing feature (described below). A menu may be up to 200 lines long, and may be displayed a page at a time. The arrow keys move the cursor up and down the lines of the menu, or you may select a menu line by simply typing its number. Following the number with ENTER causes the program on that line to be loaded and executed.

Menu entries are created through the editing feature of MENU. Without going into excruciating detail of the editing process, suffice it to say that you may, with relative ease, add, change, or delete lines in the menu file. You can kill the entire menu file, or you can lock or unlock the menu file. A menu entry looks something like this:

```
5     PROGRAM/BAS
This is a description of the program
```

The number is the line number of the menu entry; this is followed by the name of the program; the right-hand portion is a comment field, usually used for a description of the program, and ignored by MENU.

The "program name" portion of the menu entry is more than just the name of the program to be run; it is used to tell MENU exactly what to do when that line is selected, and can be fairly complex. A "program name" may be any one of five basic types of items: (1) An ordinary file name, such as another machine language program; (2) A BASIC program name, followed by the number, in parentheses, of concurrently open files allowed by the program, such as INVENTRY/BAS(2) — this number must be specified, even if zero; (3) A sub-menu to be loaded and displayed; (4) Blank (used with a description field; this is useful for a "title" line for the menu), or (5) A TRSDOS command.

In cases (1) and (2), MENU can pass information to the program it calls through a file it creates called MENU/CHN. Of course, the called program must be capable of reading that file and acting upon the information contained in it. Certain special characters tell MENU some specifics about the entry. For example: PRINT%ABC/DAT requests MENU to run a machine language program called

PRINT which will print (or perform some operation on) a file name read from MENU/CHN — in this case, ABC/DAT. #FREE tells MENU that the entry is a TRSDOS command — when completed, control returns to MENU; *EDIT informs MENU that it should load and display a sub-menu called EDIT.

As a master menu for a system of related programs, MENU is ideal. Each program called by MENU can chain back to MENU (with SYSTEM"MENU") when it is finished, allowing the user to select another program in the system. The F2 key exits MENU and returns control to TRSDOS or BASIC, depending on where it was invoked. In addition to MENU, two other versions, MENUP and MENUC, are included for automatic startup and use as a main menu file other than MENU/DAT, respectively.

Documentation — Good News For A Change

The good news is that the documentation provided with all four packages is clear, coherent, and complete — a radical change from most "user's manuals." This is especially true in light of the fact that the authors of the programs also wrote the documentation. Usually this results in a collection of sparse, incomplete sentences and jargon only understandable (maybe) to another programmer. In this case, the authors (Dale Worley, an MIT doctoral candidate, and David Renaud, a college graduate with over seven years of professional programming/writing experience) have produced documentation simple enough for the novice user to understand and use, yet comprehensive enough to satisfy the curiosity of the professional programmer interested in exactly how the programs do what they do. It is evident that the authors put a lot of thought into the design of both the programs and the accompanying manuals.

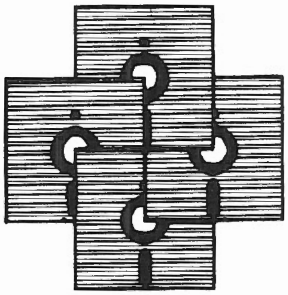
Summary

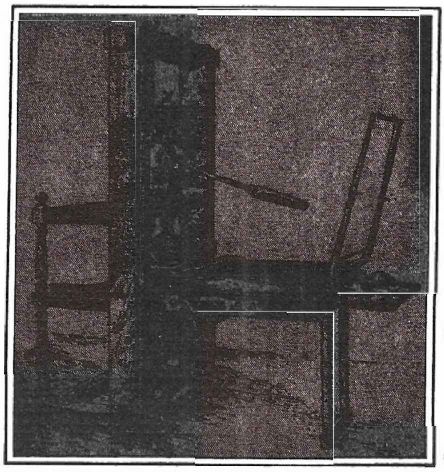
For the professional or part-time programmer, all four packages can be very useful. Personally, I wouldn't buy VARLIST without its companion utility, CREF, and vice versa. However, RESEQ and MENU are individually useful packages. In any event, $130.00 for all four is a reasonable price for utilities as good as these — all in all, very useful goodies to add to your programmer's tool box.

Ed. Note: Mr. Charles Sumner of Word Software Resource told ***two/sixteen magazine*** *that since we received the review copies of his company's product, the speed on CREF, the Cross-Reference Utility, has been increased three-fold. The authors felt CREF moved entirely too slowly, so they began checking into the reasons. And they found that the Sort routine was not using all the memory available to it, so they "opened up" the complete memory capability to the sort routine, tripling the speed of CREF.*

VARLIST, MENU, CREF, AND RESEQ
are available from
World Software Resource
(a division of Sumner Enterprises)
Highway 146 North
Grinnell, Iowa 50112
515/236-8406

Prices:
VARLIST—$24.95
CREF—$34.95
RESEQ—$44.95
MENU—$24.95





Printers Compared

Robert P. VanNatta
222 South First Street
St. Helens, Oregon 97051

Looking for a printer? Hundreds of different printers must be floating around the market place. The problem of making an intelligent review of the choices available is almost as difficult as making an intelligent purchase from the choices available. If we try to cover all the printers on the market, we wind up with a table of features. If we try to cover only the newest printers, we miss out answering vital questions such as, "Is it reliable?"

The comments which follow center around five printers: a NEC 5515, a NEC 5530, a TI 810, a C.Itoh F-10, and an OKIDATA 83A. It is my notion that the first question any "Shackie" must resolve is the entry-level question of whether to buy a printer from the Shack or from someone else.

Most likely the correct answer to this question, in turn, revolves around the question of the type of software chosen. In a country that has accepted 4 foot 8 and one half inch rails, Radio Shack is selling TRSDOS with a 5 foot gauge.

One of the major incompatibilities between CP/M and TRSDOS involves the way these operating systems handle printers. The norm for CP/M is to transmit both a carriage return and a line feed character at the end of each line. The norm for TRSDOS is to transmit a carriage return only and expect the printer to automatically advance to the next line.

The reason CP/M behaves the way it does is attributed to the fact that CP/M was developed on a teletype, and this is how teletypes work. The TRSDOS convention is attributed to the fact that the original printer driver for the TRS-80 Model I was put in a ROM chip under very severe space constraints. I suspect that the line feed character routine was simply left out because it wouldn't fit in the chip.

For whatever historical reason, Radio Shack printers have an auto line-feed feature built into them. This feature is usually not switched and usually cannot be disabled. By contrast, most other printers on the market have an auto line-feed switch somewhere. Occasionally, this will be a front panel switch, but more often it will be a DIP switch buried in the innards.

The matter has been further confused by the fact that CP/M versions for the TRS-80 Models II/12/16 have optional filters built into them which permit you to filter out the extra line-feeds. This makes a RS printer usable under CP/M. Conversely, TRSDOS printer drivers are available (maybe, but don't ask me where) which add the required line-feed.

The availability of these patches notwithstanding, it is my opinion, wise or otherwise, that the presence of a switch to toggle the auto-line feed is a very important feature, particularly if the use of CP/M is contemplated. (What are you going to do with that Shack printer if you decide to buy some other brand of computer?) I use CP/M exclusively, as do about half of all Model II/12/16 owners, and it is no accident that I have five non-Shack printers and no Shack printers.

I am of the notion that printers from the Shack ought to be avoided like the plague by CP/M users. Conversely, I am also of the notion that they, perhaps, must be accepted by TRSDOS users. Consider Scripsit, for example. It is a popular and useful program which rivals the best of the word processing programs. It is, however, pathetically dependent on Radio Shack printer hardware.

I don't mean by this to suggest that the Daisy Wheel II is a bad printer. At $1995 it has competitive speed and about half the features of the F-10 Starwriter that sells at around $1400. However, when it was introduced two years ago, it was the first daisywheel to break the $2000 price barrier. My gripe is actually a philosophical one. It just seems to me that if you buy company software that won't work without the company hardware, and company hardware that won't work without the company software, pretty soon the company store is going to own you. A mark of wisdom in the computer rat race is to absolutely refuse to consider the purchase of any product that is not second-sourced. To me, this means you do not buy printers that won't work with competitive computers; you do not buy computers that do not work with competitive printers; and you do not buy software that won't work on a variety of hardware. After all, who will benefit the most if you are locked-in?

C. Itoh F-10 Starwriter

The F-10 Starwriter is a letter-quality daisywheel printer that works at 40 cps. I have had it in use for about three months mostly under control of WordStar.

Setup

The parallel ported version may be easily connected to the Models II/12/16 with the standard Shack parallel printer cable. Configuration is done by means of 18 reasonably accessible DIP switches 'ocated behind the front panel.

For CP/M, out of the block of 8 switches, all switches should be forward (open) except the fourth one. Out of the block of 10, all switches should be forward, except 3, 5, and 8.

Features

Optional configurations include serial port drives and Diablo and Qume printer control emulations. This unit uses standard print wheels and Diablo Hytype II ribbons. It has no cooling fan, and gets quite warm under continuous duty applications. The back of the printer is a heat sink, and it will feel warm to the touch after several hours of non-stop use. This printer was subjected to an abuse test that consisted of running it with a tractor feed nonstop for 36 hours. It performed without error. Fabric ribbons were observed to last about six hours and Multistrike ribbons about 2.5 hours. Production was observed to be 10 to 15 percent less than the NEC 5515.

The tractor and the paper feed method is reliable. It will accept continuous forms either from the back or the bottom. The tractor (optional) may be used either as a bi-directional or uni-directional device. It supports all WordStar print enhancement features (bold-face, double-strike, underline, subscripts, superscripts, and almost any imaginable combination of line and character spacing) except ribbon-color change. It will print 136 columns of Pica type and 163 columns of Elite type. Print spacing is to the nearest 1/120th of an inch horizontally and to the nearest 1/48th of an inch vertically. The paper is limited on the F-10/40 to an original and two copies. The manual indicates that the 55 cps version will support ribbon color changes and from one to five copies. The standard escape control codes are different from either the NEC or the Diablo but are supported by a menu installations procedure from WordStar.

The printer, when not in use, emits a barely audible high-pitched squeal. The noise can only be heard for a few feet, but is painful to the ears.

With respect to print quality, the F-10 should be rated slightly below that of the NEC. Print quality was evaluated by counting the number of misformed or uneven letters generated by a three year old NEC and the near new F-10 while using Multistrike ribbons. The NEC was the clear winner in this test, but one must wonder whether it is a test of printers or ribbons.

The other print quality test was made by comparing the quality and uniformity of printer output when the printers were fitted out with a faded and worn fabric ribbon. Some letters such as an 'm' or 'w' have greater mass than an 'i' or 'r', for example. If each letter is printed with a uniform hammer strike, the result, because of the mass difference, will show some letters are printed with more effective pressure than others. The NEC Spinwriter evaluates every character printed on a scale of 1 to 7 and controls the hammer pressure accordingly. I am not privy to the technical specifications of the F-10 (I have them on the NEC), but if it is supposed to have a variable pressure hammer, it does a bad job of it. As seen on a beat-up ribbon, the F-10 will produce very faded large mass letters and much brighter low mass letters. By comparison, the NEC will produce almost uniformly faded characters.

In fairness, it is time to observe that the F-10 is about $1000 less expensive than the NEC.

NEC 5515 and NEC 5530

The NEC 5500 series is no longer in production; it has been replaced with the 7700 series. So far as I know, the 7700 series is substantially like the 5500 series. On all the Spinwriters, the last two digits of the model numbers serve to identify the personality of the printer. The NEC 5510 is the standard serial ported printer without a keyboard (R/O). The control codes are unique to the NEC. The NEC 5515 is a serial printer that looks exactly like the 5510, and is the same, except that it responds to the control codes of the Diablo 1610/1620. The 5520 and 5525 are similar to the 10 and the 15 except that they have keyboards. The 5530 is parallel ported.

A bit of history is perhaps in order at this point.

The Diablo 1610/1620 used to be the standard of the industry. These printers were originally designed to be terminals for minicomputers and the like. They were large, printed at 40 cps, and their vendors liked to think they were worth big bucks. Then in about 1979, Spinwriters started washing up on the beaches in great numbers. They had a list price of $3000, worked at 55 cps, were rumored to be more reliable that the Diablo, had front panel switches to change the baud rate (it took a soldering iron on the Diablo 1610, I understand) and had a reverse channel connection in the interface.

To state it mildly, if you have only been reading the computer magazines for the last year or so, you may have never heard of a Diablo 1610. Diablo introduced the 630 which is a smaller, lighter printer to respond to the NEC invasion. No one has ever claimed the 630 was better or more durable than the 1610, just less expensive. The 630 was introduced as the competitively priced "little brother" but seems to be Diablo's main entry now that the 1610/1620 and also the 1630/1640 seem to have faded from view.

A unique feature of the Spinwriter is its use of a print thimble instead of a star wheel. The thimble rotates on a horizontal plane and is essentially a starwheel with the fingers bent up to form a cup or thimble shape.

The NEC will print 128 different characters compared to 96 for the Diablo and similar star wheel printers. This is mechanically possible because the NEC thimble has 64 fingers, each of which holds two characters, whereas the starwheel has 96 fingers of one character each.

This permits the use of thimbles with an alternate character set of up to 34 characters. This feature is accessed with a toggle, so its implementation is not too difficult. In fact, if you have a NEC 5510/5520 or a NEC 5530 using NEC drivers and not some odd ball emulation, you can access the alternate character set from WordStar. It is switched on with a QP and off with a PW. You, of course, need a thimble which actually has an alternate character set on it (Some thimbles do and some don't).

Setup

As with any serial printer, the NEC 5515 requires the fabrication of a special cable adapter in order to make it work. This is not the fault of the printer. The unfortunate fact about the standard RS232C serial connectors is that about the only thing standard about them is they have 25 pins (most of which are not used). No one seems to agree as to what is hooked onto those pins. To make matters worse, Radio Shack seemed to think they were building a terminal (as opposed to a computer) when they wired up the serial ports on the back of the Model II. A serial printer is also a terminal. Accordingly, if you plug a serial printer into the back of a Model II/12/16 without an adapter (called a null modem), nothing happens. The logical problem is that you have plugged two terminals into each other.

In order to get a serial printer to work, you essentially need to cross-jack it. The cleanest way to do this is to fabricate a short cable which will plug in the back of the computer and into the printer cable. The current Pickles and Trout users' manual documents the method of making one of these adapter cables and also advises that P & T will sell one for "nominal cost." The only other thing necessary to make the Spinwriter work correctly is to configure the reverse channel to be high when active (inverted).

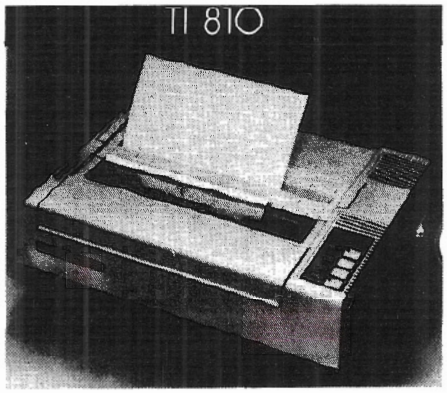
Watch Out For The Handshake

Computers normally do not transmit data at the same speed at which a printer will print it. The transmission speed on a serial printer is called the baud rate. You can, of course, assure the correct operation of a printer by retarding the data transmission rate to something below what the printer will handle. If this is done, the printer will never be overrun, and the result will be correct. This approach is not particularly efficient.

A much better way is to transmit a blast of data to the printer as quickly as possible and then let the computer do something else while it is being digested. The implementation of this concept involves the use of a storage buffer in the printer and some means by which the computer can be informed of the status of that buffer. There are at least three conventions for this communication. Two of them are so-called software solutions which involve introducing a control code into the flow of data. When the control code arrives at the computer, it will shut off the transmission until a further code is sent. The Spinwriter and most other serial printers support both software methods (ETX/ACK and X-ON/X-OFF). CP/M versions for the II/12/16 support both of these industry-standard protocols. The problem with using either is that the arbitrary insertion of control codes in the stream of data can have unforeseen and undesirable side effects. Ordinarily, a control code can be readily recognized and no problem occurs, but, if you are using a program that requires the transmission of escape sequences to the printer, you will find that the arbitrary insertion of a control code into the middle of an escape sequence may generate what is known in polite company as an undefined result.

The Hardware Handshake

The most convenient (and reliable) way to interface a serial printer is to use a hardware handshake called a "reverse-channel connection." A reverse-channel connection is effected by running a separate wire to the computer from the printer. If you have a Radio Shack computer, it will need to be connected to pin 5 (Clear to Send) and will need to carry 12 volts when the printer is ready to receive data. If you have any other computer, the Clear to Send pin will likely by pin 4, and it will need to carry 12 volts when the printer is NOT ready.



As you might guess, most serial printers, including the Spinwriter, are shipped from the factory with the reverse channel configured, so it won't work on a Radio Shack. On the NEC 5515 there is a little blue box containing 8 DIP switches on the PVC board immediately behind the front panel. The fifth switch from the left toggles the reverse channel protocol. That switch needs to be down to get things to work correctly. It is a little difficult to reach, but an unfolded paper clip or a small screwdriver is usually sufficient.

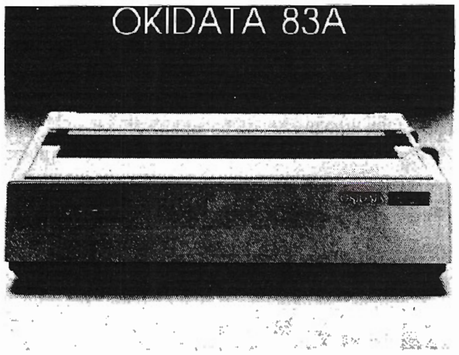
The documentation on the NEC is very good, and if you are willing to read the manual, it should not be hard to locate the reverse channel switch regardless o the model you have.

By contrast, the documentation on the F-10 is hopelessly confused (and confusing). The F-10 manual is a single manual intended to support both serial and parallel versions of both the F10-40 and the F10-55 printers. It seems that whoever translated the manual into English stirred up the pages and left out the chapter headings before sending it to be printed. It is impossible to tell which portions of the manual apply to which version of the printer.

The 5530 Setup

Of all the printers I have configured, the 5530 was the most difficult. It took me about a week of reading the documentation and finally a number of phone calls before I figured out how to set all the DIP switches. The 5530 has 32 DIP switches buried in its innards. They are only accessible by dismantling the printer and pulling the main circuit boards out of it. The boards are large and plug into an even larger mother-board with great difficulty. Amid great fear of disaster, and without the recommended board lifting tool, I pried them out with a screwdriver and made the necessary changes.

I bought this 5530 from another Model II owner who had been unsuccessfully trying to run Scripsit on it. The prior owner's frustration level over its failure to run Scripsit was so high that he actually had a lawsuit pending against the NEC dealer over the printer at the time I acquired it.


OKIDATA 83A

Once the switches are set correctly, the 5530 will work correctly under CP/M in all respects. My only criticism is that the WordStar driver for the 5530 is so complex, it fails to get data to it fast enough to make full use of the 5530's full speed. As a result, the throughput of the 5530 is considerably less than that of the serial version of the same printer and probably even below that of the much less expensive F-10.

It is clear from my experience that the serial versions of the NEC are much to be preferred over the parallel ported versions. Mechanically, both versions are identical. Both will handle up to a seven part form, if necessary, and may be either bottom or back fed.

Both of my NECs are about two and one-half years old, and neither has had serious service problems. In fact, only one has ever had any service. It went to the shop because someone attempted to change a ribbon cartridge without first releasing the latches holding it down. The manual recommends an annual grease job and oil change for the Spinwriter, but I haven't done it.

For what it is worth, NEC advertises a claim of 3500 hours mean-time-between-failure (MTBF) (*Computer System News*, November 22, 1982). This time is in stark contrast to the 300 hour MTBF claimed by Radio Shack on its dot-matrix printers (*Microcomputer News*, November, 1982, pp. 18-19).

About Those Dot Jobbies

A comparison of a TI 810 with an OKI 83A is something of an apples and oranges project. The TI 810 is part of the Texas Instruments 800 series terminals that has been around since 1977. This printer is a 132 column, 150 cps, 9x7 dot-matrix. It exists in numerous configurations, from a stripped version that prints upper case only, to KSR terminals with compressed print and vertical forms control.

The particular version tested is a basic model with the full-ASCII, 95 character set. The application of the TI 810 to microcomputer use seems to be an afterthought. The *two/sixteen* reader survey did not reveal the TI 810 to be very popular as an accessory for microcomputers. If you peek in the back room of your stockbroker, bank, or courthouse, you may well find several of them, however. For example, our local county tax collectors use a TI 810 to print the county property tax statements. They are also more-or-less standard equipment around travel agencies for printing airline tickets.

The TI 810 supports baud rates to 9600, and reverse channel protocol is supported from pin 11. As with the NEC, the reverse channel must be inverted to work with the Model II/12/16 microcomputers. Unfortunately, on the TI 810 there are no DIP switches to invert the reverse channel. Instead, it is a so-called "strappable" option. This means that you must pull the main processor board (board lifters are conveniently built in), hunt down the correct jumper and change it. The whole operation is terrifying to anyone who hasn't had their hands on a computer board before. Actually, however, it takes less than 15 minutes to do the whole thing. If you can count to ten and have at least one thumb and one finger, you ought to be able to do it.

The TI 810 is respected for its durability and reliability. Interestingly, however, it is the only one of the five printers reviewed here that has died in the line of duty. It croaked on about the second day out of the box and required a new print head and driver board. (The warranty fixed it without any backtalk, and there has been no problem since.)

The Features

The OKI 83A costs half as much, weighs half as much, has more features, and prints three-fourths as fast as the TI 810. The print quality of the TI 810 with its 9x7 patterns is conspicuously inferior to the 9x9 pattern of the OKI. Both use very inexpensive open-spool, half-inch fabric ribbons. The OKI ribbons are listed in lots of a dozen in the November 1982 INMAC catalog for $2.65. The spool is of the 2-inch Underwood type with 12.5 yards of nylon ribbon. It seems to fade after 300 or so pages of printing. The TI ribbons are much longer (40 yards on a 3-1/4 inch spool) and require less frequent changes (perhaps one for every box of paper). INMAC lists them for $4.95 each in dozen lots. By comparison, INMAC sells EPSON ribbons for the MX80 at $9.65 and for the MX100 at $15.70 each in quantity.

The TI will work on continuous forms only (up to six part paper), whereas the OKI will accept four part continuous forms or single sheets (it has a platen). Both can be fed from the rear or from the bottom, but a practical installation almost requires a bottom feed for either. The OKI comes with both a low-speed serial and a parallel port as standard equipment. An optional high-speed serial port has increased buffering and baud rates to 9600. The TI normally comes with only a serial port, although parallel models are available.

If a serial configuration is contemplated, the high speed option is highly recommended for the OKI. I use the TI 810 interfaced with a reverse-channel hardware handshake at 9600 baud, and even though it has only a 256 character buffer, it is the only printer mentioned in this article that permits satisfactory use of the background print spooler of WordStar. Another highly desirable feature of the TI 810 is a pair of form alignment buttons on the control panel. These permit paper alignment adjustments to be made "on the fly" while the printer is working.

The TI 810 and the OKI 83A are not directly competitive printers. The low-end segment of the dot-matrix market is defined as being "under $1000." Reportedly, Epson has about 40 percent of this market and OKI has 15 to 18 percent, and is coming on strong. Considering the fact that the low-end segment is thought to consist of about a million printers for 1982, "this ain't hay."

Meanwhile, back at the farm, Texas Instruments has the 810 firmly in the "under $2000" market segment. Please don't take this to be a criticism of Texas Instruments; they are still out there working. Centronics deserves the criticism. They almost invented the dot-matrix printer and have responded to the increased demand for dot-matrix printers by cancelling their much heralded "quietwriter" project, and announcing that they have lost money for yet another quarter (this makes eight quarters in a row).

If you are inclined to buy greenbar by the pick-up load for your printer, then, in that event, the TI 810 is probably your logical choice. On the other hand, if brute speed and durability are not top priority, the OKI is a pleasant, inexpensive printer with lots of features you will probably never use.

In an effort to keep the TI 810 from fading completely into history, Texas Instruments has recently introduced a LQ (letter quality) option for the Omni 800 family of printers. The option consists of a plug-in conversion board and a substitute high-resolution paper advance motor.

The cost of the LQ option is $750 (more than I paid for my OKI 83A), and you either have to install it yourself or pay extra for installation. The features added are significant, however. First, you get six type faces, which can be combined in the same document and even the same line. The switching is done with an escape sequence. This contrasts with the OKI 83 which allows software switching between standard, compressed, and expanded print only at the end of a line. The high quality print variations are produced by four horizontal passes of the seven-wire printhead with very small vertical motion of the paper between passes. The characters are formed with 13 mil dots placed with an addressability of 240 dots per inch horizontally and 288 dots per inch vertically.

A compressed print mode (14.6 cpi) permits 195 column printing on standard 14.875 inch wide paper. The compressed print option effectively increases the printer output in characters per second to 220 cps from the standard 150 for single pass printing because the compressed print is generated by increased printhead effort, not slower carriage speeds.

By contrast, the OKI 83A appears to vary the character width by adjusting the carriage speed. The OKI approach results in several undesirable side effects, however. The most obvious is the inability to "shift gears" in the middle of a line, and the inability to print more than 132 columns, even in the compressed print mode.

The latter limitation effectively prevents the OKI 83A in the compressed print mode from printing on paper wider than 8-1/2 inches.

Conclusions

On the letter quality side, there is no equal to a NEC 55xx (now the 77xx) series printer. For CP/M, the serial versions are probably a better deal. If price is an issue, the F-10 Starwriter is a little bit less printer for a lot less money.

On the dot-matrix side, the TI 810 is a downloaded mainframe terminal. It is faster and at 55 pounds, certainly makes a better boat anchor. It is likely one of the most durable dot- matrix printers on the market. With the LQ print option the TI 810 has features for everybody, but the principal marketing thrust of the TI 810 is for use as a mainframe and minicomputer terminal, and this is probably where it belongs.

I have plunked down my clams to upgrade the 810 mostly because it is too good a printer to scrap. The 83A is a little less printer for a lot less money. What's more, OKI introduced the ML-92 and ML-93 at the COMDEX. These printers are expected to be priced at 20 percent above the 82 and 83 printers and provide 160 cps performance with 40 cps correspondence-quality printing. This means that they will likely do everything the enhanced TI 810 will do for about half the money.

II + II =

(BUSINESS DYNAMICS)

DOSPLUS II and the TRS-80 model II. The Dynamic Duo.

The TRS-80 model II was created to help you take care of your business. DOSPLUS II was created to help your TRS-80 help you and your business even more.

DOSPLUS II replaces your O.E.M. dos with an ultra-sophisticated operating system that is 5 or more times faster, more reliable and is totally compatible with all standard business oriented software designed for use with the model II. The result is a computer/operating system that far exceeds manufacturer's specs—with more potential and more capacity to be of service to you.

DOSPLUS II operating systems are available for either your floppy or hard disk drives. The hard drive operating system is compatible with Radio Shack, QuCeS, Corvus Micro Mainframe, VR Data and other popular hard drive sub-systems.

DOSPLUS II . . . All the quality and dependability you'd expect from DOSPLUS.

For the dealer nearest you call
1-800-327-8724
Dealer inquiries invited.

$249.95

For product information (305) 983-3390

DOSPLUS DOSPLUS first in quality!
First in the industry!

To order call toll free 1-800-327-8724

POWERSOFT

A Division of Breeze/QSD, Inc.
11500 Stemmons Fwy., Dallas, Texas 75229
(214) 484-2976 Micronet 70130, 203

MICRO SYSTEMS SOFTWARE, INC.
4301-18 Oak Circle.
Boca Raton, FL 33431

7 SIMPLE PROGRAMS

Roger Conant
University of Illinois
Circle Campus
Box 4348
Chicago, IL 60680

The following simple programs — REVERSE, FORWARD, SINGLE, DOUBLE, BELL, PRON, and PROFF — can be keyed in with little effort under DEBUG and produce a useful pay-off for you. These programs, which are identical except for one or two bytes, do the following:

REVERSE turns on Reverse Video
FORWARD turn on Normal Video
SINGLE turns on 80-Character Mode
DOUBLE turns on 40-character Mode (Double-sized text)
BELL rings the bell on your printer if you have one
PRON is the same as "DUAL ON"
PROFF is the same as "DUAL OFF"

Ring The Bell
This easy program, designed to ring the bell on your printer, must be keyed in under DEBUG at 3000 Hex (hereinafter expressed 3000H) and moved like this (type in center column data):

[STEP #]	[TYPE IN]	[COMMENTS]
1.	DEBUG ON	Turns on the debugger, but you don't see anything
2.	DEBUG	You get the display of memory 2800-287F hex

Now you are in DEBUG—**DON'T** press "ENTER" after entries!

[STEP #]	[TYPE IN]	[COMMENTS]
3.	M	You will see "M A=...."
4.	3000	This displays 3000-307F Hex
5.	<F1 Key>	Pressing the F1 key gets you into entry mode
6.	06073E12CFC9	These characters enter the BELL program into memory
7.	<F2 KEY>	Pressing the F2 key exits from "ENTRY" mode of DEBUG
8.	O (The letter O)	Turns DEBUG off

The BELL Program is now stored at 3000-3005H. You have exited DEBUG, so resume pressing "ENTER" after entries.

In response to the TRSDOS READY, type in:

```
DUMP BELL {START=3000 END=3007 RELO=2800 TRA=2800}
```

Now the program is ready to run. Type:

```
BELL
```

and the bell on your printer will ring.

Now Try The Others
REVERSE, FORWARD, SINGLE, DOUBLE, BELL, PRON, PROFF are all programmed in exactly the same way except for two characters in the string entered at Step #6 that differ for each of the programs.

Where the string contains 07 for BELL, you use 1A, 19, 1E, 1F, 07, 0E, 0F respectively for these seven programs. And where the string contains 12 for BELL, you use 08 for all the others. Of course, in the DUMP statement you use the corresponding name. Repositioning the programs to 2800H avoids conflict with BASIC and other programs, so that in BASIC the line

```
SYSTEM "BELL"
```

will ring the printer bell without affecting the BASIC program in any way.

Reverse Program
For example, to create the REVERSE program, you would proceed as follows:

Follow Step #s 1-5 as for the Bell Program above. At Step #6, change the string to read:

```
061A3E08CFC9
```

Proceed through Step #8.

In response to the TRSDOS READY, type in:

```
DUMP REVERSE {START=3000 END=3007 RELO=2800 TRA=2800}
```

Now the program is ready to run. Type:

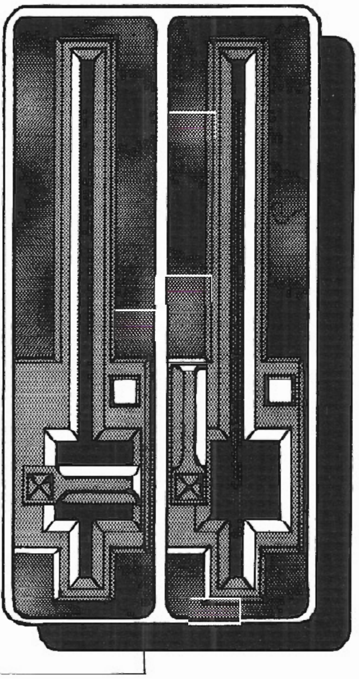
```
REVERSE
```

and your video display will be reversed.

Try out the other small programs and see if they don't prove useful to you.

some random comments on...

THE DISK DRIVE ISSUE



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

Early Model II Drives
The first Model IIs were furnished with Shugart drives as the primary (A or 0) drive; early CDC drives were used in the expansion bay. At least four versions of the CDC units were used during the first year of the Model II, and differences between the first three versions were minor. The fourth version had an entirely different circuit board, known as the "LSI" unit since it used a single large chip to replace most of the chips on the earlier boards.

As early as six or seven months after delivery of the Model II began, Radio Shack issued a service bulletin, instructing their technicians not to even attempt making the earlier units work properly, but instead they were told to change the boards to the LSI version whenever a problem developed. However, with the relatively high turnover of technicians, this bulletin was then — and still is — usually ignored. We know of expansion units, still with the original boards, that have required numerous trips to service over the past three years.

We have been using the oldest CDC drives with complete satisfaction and no problems at all for the past three years. The only difference between our Model II and the early ones issued directly from Radio Shack is that we installed the drives and made the cable connections ourselves. Further we are still using the first "engineering change" disk controller board and will buy another Model II for expansion to hard disk before we will let Radio Shack make any updates to our machine. We do not believe in messing with a good thing.

Why have we been successful in using the CDC drives where Radio Shack has failed? First, there is nothing wrong with the CDC drives. When Radio Shack stopped using their drives, IBM picked up the slack and bought large numbers of CDC drives. Second, linking long, compiled programs using PLINK is a tremendous disk exerciser which the built-in Shugart had difficulty handling. The CDCs have yet to give us a problem. Third — well, let us examine Radio Shack's service, attitude, and implementation of their system of hard- and software.

Booting The System
Radio Shack designed their system so that when you turn on the power, you receive the "Insert Diskette" message. You insert a disk, and the system immediately reads the disk, pulling the operating system into memory. On many microcomputers, the computer is turned on, the disk is inserted, and then the user does a "reset" or tells the system's ROM monitor to "boot" the system by reading the disk that has been inserted into the drive. We are sure Radio Shack picked its method to make the Model II easier to use, but this caused problems in its disk interface to the system. The most obvious of these problems is the A or 0 drive must give a "false ready" to the controller. In other words, the drive is selected and "ready" even when it really isn't.

Disk Changes
Radio Shack also wanted its system to automatically "know" when a disk had been changed, although the advantages offered by this hardware manipulation have never been implemented fully in the operating system software. When a disk is changed in the CP/M operating system, you must tell the system the change has been made via a"warm boot," either with the "Control C" or via a software "reset;" otherwise, the disk is (hopefully) made "read only" or effectively "write protected." For the same reason (destroyed directories), Radio Shack requires your use of the "I" or "System I" when a disk is changed. The "disk change" option in the hardware should be able to tell the software a disk change may have taken place (the door opened, anyhow), and the system should have automatically "reset" the drive. As we said, this feature has apparently never been fully installed.

Drive Termination

Since the A or 0 drive has to be "ready" for the "automatic boot," certain conditions had to be taken care of automatically for the user. In addition, Radio Shack wanted the user to be able to buy an expansion bay, plug it into the connector on the back of the computer and go operational with the add-on without having to open the computer itself.

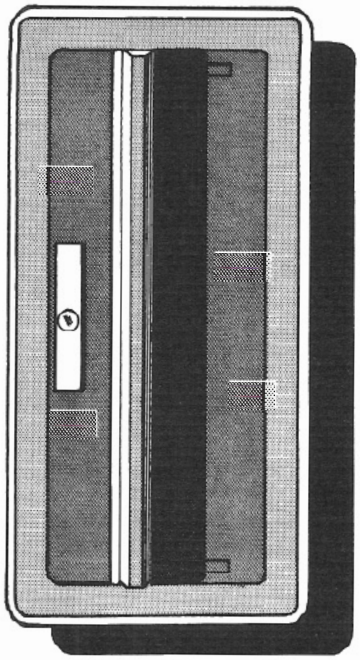
If you have installed disk drives on other systems, you already know that such ease of installation is most unusual in this business. At the very least, you usually have to remove the terminating resistor pack from the A or 0 drive and make sure a pack is in the last drive of the "daisy chain." On the Model I, this was taken care of by selling two different catalog numbered drives. The add-on units did not have the resistor pack, but the original drive was supplied with a pack (and the TRSDOS manual and operating system).

This little fact can cause problems since the disk controller does not like the extra load caused by two sets of terminating resistors. Radio Shack tried to work around this by jumpering the A or 0 drive (which also caused problems with the "terminator" on the Model IIs without expansion drives). One of the early fixes, which often worked, was to bend the pin on the resistor pack in the expansion that terminated the head since it was already terminated on the A or 0 drive. There was also a defect in the controller chip which Radio Shack programmed around, but there are also chips without the defect.

"Jumper" Options

If you've ever looked at the board on your A or 0 drive, you've noticed a large number of jumpers. Many of these jumpers are used to select special options for a particular implementation. If you look at the jumpers on five different Model IIs, you may find five different jumper combinations. The expansion CDC drives also have these jumpers, only on the older boards they are soldered in and look like resistors. It was probably too much to expect the drive vendors, the computer assemblers, and the computer technicians to really be able to figure out the various combinations of jumpers that could be used.

Some of you using Pickles & Trout CP/M on the Model II may notice that everything seems to function very quickly, with very little head loading and unloading. However, if you change one jumper on the A or 0 drive, then the head loading will return. Other Model IIs will have the head loading as usual (on COPY, for example). What happens is that when one head is loaded, all heads are loaded, even if the drive is not selected. Since the head is already loaded, disk I/O is much faster.



Stepping Rates

Then there is the faster "stepping rate" question. Radio Shack has always maintained that the CDC drives had to step slower than the Shugart. Our experience has been the opposite. The Shugart cannot handle the faster rates as well as the CDCs.

Drive Timing

Drive timing against the index sector pulse was probably one of the major causes of problems with the use of the CDC drives in the expansion bay. The specification for the Shugart drive is 200 +/- 100 micro seconds, while the CDC specification is 450 +/- 100 micro seconds. Both types of drives could be within specifications and still be completely different. For example, set the Shugart at the low end, say 125, and the CDC at the high end of its range, or about 525, and you have a timing difference on the drives of 400 micro seconds, which on a computer is "forever." One of the more inventive Radio Shack techs (locally) figured out that you could set the Shugart at 300 (the maximum of its range) and the CDCs at 300 (the middle of their ranges) and everything would be beautiful (and it is!).

Service Bulletins

What is irritating is that Radio Shack, and numerous of their technicians, knew of these little "fixes" three years ago — we and others told them. Nothing happened, or at least, no one read service bulletins — if such were ever issued.

"Critical" CP/M And "Double Density"

Another irritating practice was the one in which Radio Shack blamed CP/M for its problems. They said that the implementations of CP/M were using an "extended" double density, giving 596K of user space per disk, instead of the 490K of TRSDOS which was too "critical" for the hardware to handle. Bull! And now, check the density on the disks when you switch to a Radio Shack hard disk. They are doing the same thing — using the "too critical" density!

This is not to say the CP/M implementation did not have faults — they did. In the summer of '81 Radio Shack furnished the four major vendors of CP/M implementations on the Model II with some patches which would solve some of the same problems Radio Shack was having with TRSDOS (the 2.0a patches). Lifeboat never did get the thing right after that, while Pickles & Trout and ATON did. Cybernetics and FMG were essentially out of the picture.

Back to the "extended density" question. We know of over forty installations using four drive systems and CP/M. Four are having problems; the others are not. The four who have problems cause some themselves. The others are related to poor service.

Door Interlock

For example: the disk drive doors have an "interlock" switch that prevents the drive from being "ready" if the door is open. You will seldom have problems with the A or 0 drive interlock since you seldom switch the operating system disk. But if your application requires different data disks in the expansion drives, you do open and close these drives with some frequency. If a drive suddenly starts giving errors or "not ready" messages, check to see if the rod which is pushed in by the closing of the door is actually closing and keeping the interlock switch closed. It is a simple adjustment, but there are Radio Shack techs out there who will never check it or fix it.

Also, you can have a defective switch. One system was in for service three times in the same month with the same problem, and the technician knew about the interlock switch problem (we told him). Finally, on the third try, he changed the switch and solved the problem.

Also, the nylon rod which activates the switch can become worn with constant use. Again, easy to repair.

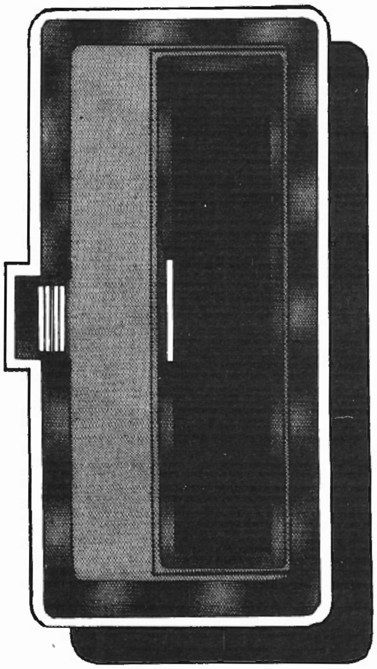
Replacement Pressure Pads

When you had problems with the A or 0 drive, and the technician replaced the head pad, he checked the replacement using a 'scope — right? Wrong! Yet a look at the Shugart manual tells the tech to rotate the new pad in ten degree increments, check each step with a 'scope for the best setting. This is necessary because the head pads are (were) beveled on the Shugart. Note that recent replacement pads are not beveled.

Double Sided Drives

When you bought your expansion drives after January, 1981, you found your favorite version of CP/M no longer ran — or at least you couldn't use the expansion drives. How come? Well, one of the questions was answered in the Sept./Dec. issue of *two/sixteen*; i.e., the Model II has always been able to use double-sided expansion drives. That line, from the connector, has always been active. For some reason, the new TPI drives used in the expansion units are telling Lifeboat 2.25a and later versions that the other side of the disk is available. Pickles & Trout and ATON aren't bothered by the problem. So, from day one, you have been able to use double sided drives on the Model II. In fact, we've just ordered a double sided CDC drive. Also, if you want to implement them (with the software) you can run five inch drives, single and double sided, as well. The hardware is already there.

A comment here. If you use the thinline double sided drives, understand that they use DC motors, which are shut off after 20 seconds of no activity. Then when the drive is again selected, it will return a "not ready" (it isn't). Hopefully, your operating system will try again, and the drive will be ready. Radio Shack, Pickles & Trout, and ATON have taken care of this situation in their software.



Double sided/Double density 8" disk drive sub-systems (including case and power supply) have been available for quite some time, ranging in price from $1,150 to about $1,995 for a two drive sub-system, thin-line or standard. You could do a four drive expansion by buying the cabinet for two standard drives, and the power supply for four drives, and use thin-line drives. Note that the thin-lines are running DC motors, so your 24V supply will have to be able to handle the load.

There is one problem with any of these non-Radio Shack add-on drives: who services them? Also, use of add-on drives, even those fully Shugart-compatible, does require some "interfacing." You may have to open the computer and change a jumper or two.

Buying Add-On Disk Drives

You have a number of choices. You can buy a sub-system, ready to plug in (almost), or you can buy the drives, cabinet and power supply separately. A large number of OEM drives, brand new, turn up at computer flea markets and hamfests at about $150 to $200 with others advertised (some used) at $100 to up to $279 (double sided).

Power supplies are available new for $60 to $130, or you can easily build a dual drive supply for $25 to $30, using brand new Radio Shack parts.

Cabinets are a bigger problem. Generally, they run from $90 to $200, with most designed for the power supply to be enclosed with the drives. We do not recommend this. No matter what anyone says, we still think heat is a major problem, so you might consider leaving the drives bare (not installed in a cabinet). We've been running several this way for three years with less dust accumulation than occurs on the cased drives.

Using brand new drives, you should be able to put together a three drive, single sided expansion for less than $1,500 including cabinet. If you use OEM flea market goodies, you can do it for about $750. Double sided drives, new, would run about $1,600—$1,700 for three, and we have already mentioned the two drive price.

By the way, by adding one chip to the A or 0 drive (the circuit is included in the Technical Manual) you can run a total of 16 drives on the system. When you are using ATON hard disk CP/M (and we assume P & T, as well), the hard disk is one to four logical drives, with the floppies being numbered E through P (4 to 15).

One other note on double sided drives. You will sometimes consider addressing the double sided drive as two single sided drives. This is easy enough for the hardware to handle, but the software gets confused, especially when you "copy" from "drive to drive" on the same physical drive. The system can lose track of where the head(s) is/are on a specific drive(s) since they move together. On a separate C and D pair of drives, the head on one remains where it was when the other drive is being written after a read on the first drive. On a double headed (sided) drive, the heads move as a unit, so when the system returns to the first side, it does not know where the head really is, since it is not where it was (double talk??).

Hard Disks

Many hard disks out there will run on the Models II and 16. But what do you do for service, replacement, or a loaner? How much memory space does the add-on to your operating system take? Some take as much as 2K, some as little as 600 bytes, while the ATON hard disk implementation of CP/M for the Radio Shack hard disk gives you up to 8 to 10K additional memory, since the 16K supplied by the hard disk is used for most of the system. On the other hand, you can also buy double the capacity for half the price (and less!), so flip your own coin.

Turning On Your Computer With The Expansion Drives Off

Our Model II is #94, and we have never really needed the terminator, nor have we ever had a problem on our machine when the computer has been turned on with the expansion drives off. We have wiped out disks doing so on other Model IIs, including ones with serial numbers almost as low as ours. Since the units delivered since the first of '82 do not seem to need the terminator, we wonder if they too still wipe out system disks when the expansion drives are not on.

On our own "computer fest" add-on drives, we have never had the problem. Model Is used to do the same thing, if you turned certain printers on after the computer had been turned on (Centronics 703), but then there are two lines on the printer cables which should not be connected, but are. The printer and disk controller use the same port on the Model II, each using one-half of the byte on the port. There might be some interaction here, too.

Head Cleaning Disks

We do not use or recommend them. If your heads are actually picking up oxide deposits from your disks, then change brands of disks. The cleaning disks can put as much wear on a head in 30 seconds as a disk can in a year. We clean our heads the same way Radio Shack does, with a swab and head cleaning fluid, whenever we have a reason to go into the computer (once every six months?).

Our Only Problems

We have had three problems, none CDC-disk related. After carrying the computer to a couple of conventions, our video acted up, with shrinking and fluttering at the top of the screen. Since Radio Shack had a free "engineering update" on the disk controller board, we took the machine in for its first and only service call. The new controller was put in and the technician worked on the video, ultimately replacing the video board for $120. Back in the office, no disk I/O. Back to Radio Shack. Three hours later, the problem was solved by cleaning the edge card traces with an eraser (by the technician). He had not cleaned the contacts when the board was installed. When the Model II Technical Manual came out, we bought it (Incidentally, we recommend it). Then the video began acting up again. This time, we remembered a note in the Technical Manual about the critical routing of a wire from the video board to the monitor, around the motherboard and card cage. We went in with a wooden rod, moved the wire, and fixed the video problem. The same thing occurs whenever we move the machine, either across the room or across the country. The next to last time we went in, we found the connector itself on the video board was not too secure. Then we discovered that the real culprit all along was the edge card connector going on to the monitor board. The connector has holes for securing it to the board, but the board does not have the mating connections. This connector is relatively "loose" and is under some tension with the wires coming out of it. By making sure the connector is secure, the problems with the video are eliminated.

On two occasions, both Friday evenings, when no service would be available until Monday, the computer, after having been in use all day, went into "outer space" on any attempt to use the disk I/O. The first time, we decided to give it a shot ourselves, after having first checked everything external to the computer and after beginning to believe the error message TRSDOS was giving us. We opened up the computer and pulled the disk controller board and examined it for obvious problems. They were there all right! Down in one corner, we saw several well charred parts. A look at the circuit diagram in the Technical Manual showed that we had lost a zener diode, two capacitors, and probably a resistor. One other diode was checked with a meter and found to be questionable, so we picked up two sets of the parts for $4.84 retail in the local radio Shack store, and replaced them in the computer. It worked!

About six months later, the same symptoms showed up so we again pulled the board. This time, just the two diodes needed replacement. There have been rumors to the effect that there is a defect in the disk controller chip which causes a pulse of voltage higher than specified, and which ultimately wipes out the zener diode. As long as we can keep the machine running by replacing the same inexpensive parts once or twice a year, we are not going to worry about it. Someday, we may take the time to learn more about hardware, and perhaps make an inexpensive modification which will eliminate this little problem completely.

Again, we like the way our machine runs, and we have none of the problems we keep hearing about. We know our machine, since it is such an early version, is a little different, and we want to avoid the "updates" for as long as possible.

Our final problem was with the keyboard. One day the "ESCape" key quit functioning, which is a problem when you are using a text editor. We opened up the keyboard and looked for something obvious but saw nothing out of the ordinary. We then figured out how the keys themselves were installed and removed the ESC key. Still nothing obvious. So, we then removed one of the never-used arrow keys, and tried it in place of the ESC key. It worked! We changed the key caps, reinstalled the keys, and everything is fine. We also picked up a keyboard at a computer fest for $10.00, so we now have a supply of spare key pads.

One Final Comment On Disk Drives

Most people close the doors on their drives when the computer is shut down, but we leave them open. This keeps tension off of any assemblies and keeps the pressure pad away from the head. On double sided drives, this keeps the heads away from each other. If you close the doors to keep dust out, buy a dust cover instead, and leave the doors open.

Help!

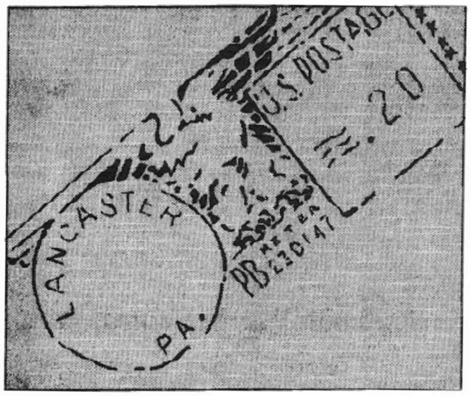
two/sixteen needs qualified Model II, 12 and 16 users to help us on a regular basis. We're especially interested in knowledgeable persons to write articles on topics to be assigned by us and persons who would be willing to review software products—also supplied by us.

We could also use part-time reporters to seek out and research topics that you think will be of interest to a significant portion of our readers.

Interested in writing a column? Part-time columnists will be expected to produce a column at least once every two months (each issue until we become monthly) on some general area of specialization. Examples might be business computing, mathematics, accounting, a specific language, or surveys and comparisons of a particular type of hardware (e.g., printers, hard disks, modems). If you would like to be a columnist, you should suggest your own topic.

Please send resumes to Barbara S. Albert, Editor, **two/sixteen magazine**, 131 E. Orange Street, Lancaster, PA 17603.

Feedback...



From: Gerald Lippey, The Lippey Company, 210 S. Bundy Dr., Los Angeles, CA 90049:

A USEFUL GADGET: We have the early expansion bay with CDC drives. Since there are no activity lights, we had to depend upon our hearing to ascertain whether a head had yet unloaded. It was even more difficult to find out or verify which of our three drives was in operation. That was before we discovered the ACT-100, a set of three lights which simply plugs into the bay connection on the rear of the Model II. It's $39.95 from: The House of Computers, 2021 Range Line, Joplin, MO 64801 417/782-0880.

USE OF FLOPPY DRIVES: Some of your readers may find our approach to the use of floppy drives of interest. We purchased the full set of four drives when our Model II was obtained in early 1980. We did not do this to increase the on-line storage. Rather, there were several other reasons: (1) We wished to be able to operate comfortably with one drive down. (2) Wear and tear could be spread over more units. (3) Operating procedures would be much more convenient if disks did not need to be removed when a file was to be copied, a special program was required, etc. (4) We could adopt the System Disk technique described below.

We have identified one disk (our "System Disk") and one drive (Drive 0) to remain constant across applications. The System Disk contains the operating system, other system software, utilities, and all other programs which are not unique to an application. It is write-protected and always resides in Drive 0. We do not remove this disk or reboot as we proceed from one application to another. Because TRSDOS reads first from the lowest numbered drive and ignores drives with protected disks during writes, it is well suited to this kind of operation. For this and other reasons, although we have CP/M, we rely on TRSDOS for all of our production work.

As an example, consider our data base applications, all of which use Profile+. The Profile run-time programs reside on our System Disk, while use menus, formats, and all other software related to the files in use are on a disk with those data files. Profile programs used to construct data base maps, screen and print formats, and so forth reside on another disk with other development software.

In addition to the obvious storage savings and logistic conveniences of this approach, it improves reliability: There is less danger of clobbering system programs or making errors while changing disks. Also, compared to the method that Tandy would apparently have us use, the all-important Drive 0 gets less wear and tear. And it turns out that a write-protected disk in Drive 0 remains intact when one forgets to turn on the expansion bay!

Our only problem is that SCRIPSIT refuses to lend itself to our procedure. It insists upon Drive 0. If anyone can tell us how to patch and rearrange this creature so that a DOCUMENT/CTL file need not reside on that drive, we would certainly be grateful.

From: Jim Semanson, Parity Systems, Inc., 710 W. Lewiston, Ferndale, MI 48220:

We have just received the first three issues of *two/sixteen*, and are very pleased with the professional and technical content of your publication.

Just a few comments in response to Gary Boatright's Letter to the Editor, in the July-August, 1982 issue concerning whether to convert to COBOL.

We are a professional business software company, and have been using RM-COBOL on the Model II/16 since its introduction in 1980. We have been very pleased with its performance and have developed applications, purchased others and even off-loaded applications from computers as diverse as IBM-370's and TI990's almost without any modifications.

The applications have been transferred from TRSDOS-1.2, to TRSDOS-2.0a, to TRSDOS-II (4.1 H-D), to TRSDOS-16; and from the OASIS operating system to TRSDOS with only a recompilation.

We are presently awaiting the TRSDOS-XENIX operating system to be delivered, so we can take full advantage of the built-in file and record locking functions that COBOL supports so well in a multi-user environment.

RM-COBOL has been in the field for many years now and runs on NCR mainframes and Texas Instruments minicomputers, as well as most all microcomputers. We have also found RM-COBOL to be almost error free and able to use sophisticated data structures and procedures in our normal business programming, without fear of object code failure.

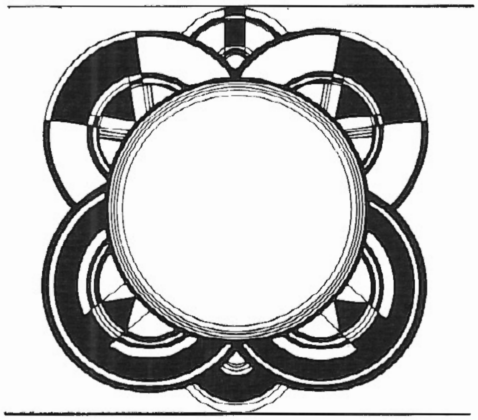
I hope these few comments will be of use to Gary and others who are contemplating using COBOL with future hardware and software upgrades in mind.

Good Luck with your publication.

From: Wayne C. Bucklaew, 700 Etheridge Rd., Chesapeake, VA 23320:

I would like to thank you for putting out a magazine that finally fills the void for the users of the Model II and Model 16. It is refreshing to finally find a periodical devoted to service instead of advertising. I know full well that advertising is the way a magazine stays in business. Future issues will have more, but from what I've seen I think that the advertisements will be tailored to our use.

Now to some of the reasons that I have written. First is to say thank you to Snapp Inc. for sharing with us their DOSFIXes. Some comments about a couple of the DOSFIXes. Number 2 DOSFIX does in fact make the operating system available to the users. Make that 1002 Shack drives, that DOSFIX 09 works on, and it is nice to speed up the access time. One problem with DOSFIX 09 though, is when you transfer from a TRSDOS disk to a SCRIPSIT 2.0 disk remember to Re-boot the system. If you don't, you will probably crash a SCRIPSIT disk. I made that mistake and am having to rewrite this letter because of it. What happens when you don't Re-boot is the operating system is maintained in memory and when the SCRIPSIT program transfers information from memory to the disk it attempts to do so with a head stepping rate faster than the SCRIPSIT program can handle. Next it will lock up the keyboard so that you cannot make any entries. The only way out then is to Re-boot and then you have left the SCRIPSIT file open and you will lose all of your documents on that disk. So be careful to Re-boot between TRSDOS and SCRIPSIT.



Problem, DOSFIX 10 did not work on my backup copy of TRSDOS 2.0 A. In fact, on re-boot it gave me a TK Boot error, so I just started all over again. DOSFIXes 04, 05, and 06 finally allow access to the memory of the computer like I'm used to with other systems. Now that I have total access to memory through Debug, I can finally start to do some of the programming I need to do in order to read other manufacturers' disks.

I would like to thank David R. Canning for showing me how to use the Debug facility, which is not explained very well in the TRSDOS manual. Also his XDIR works great and I like its format.

From: Alfred T. Dori, Independence Life and Accident Insurance Company, 20 Scholl Rd., Pottstown, PA 19464:

I have read your magazine from its inception and have had a hard time understanding much of it. My interest in a publication such as yours is to better learn how to use my Model II in my insurance business. My desire is to better learn how to "drive my car" rather than how to overhaul the engine.

I appreciate that there are a number of owners of Model IIs and 16s that can converse with you at the altitude you are flying, but my experience tells me that there are a bunch of us glider pilots also. Please remember the formula K.I.S.S. from your earlier days in the insurance business.

I am looking for business applications of Profile + and Scripsit, and refinements of them. A review of TransPro and/or like products would be appreciated.

Good luck with your magazine.

From: Sam Jones, Flashy Bookkeeping Service, 1705 Schieffer Ave., Austin, TX 78722:

I hope you have great success with your magazine. It's needed so keep up the good work.

The first thing that's right about my Model II is the price. I always like to remember transistor radios when I purchase any electronic device. In 1959 I paid $84 for a nice radio, and last week I paid $7.98 for a transistor radio that sounds better than the '59 model. The sooner the Japanese get into the computer markets the better off we will all be.

What's wrong with my Model II? Not much. The "A/O" drive has been the only one that has ever given me trouble that required a trip to the repair shop and this month marks the third year that I have used this machine. That is two days out-of-service in roughly 4,140 hours of usage.

My major disappointment has been in finding software. I do not program. I am not particularly interested in learning to program. I am also not interested in making/buying/using a "hot rod" operating system. I am an accountant and my clients pay for regular output. I do not use CP/M that much; my operators find TRSDOS easier to operate. As for using the Model II, I only wish that the RS folks had been farsighted enough to put a "Minus" key nearer to the keypad to accommodate those of us who enter numerical data. The "Break" key could have been more logically placed farther from the "Backspace" key also.

What would I like to see in your magazine? Articles that would be useful to non-programmers, specifically articles by people who have found innovative uses for existing software in application to specific accounting problems. Has anyone found a way to use/adapt/change the many inventory programs that are available that seem to have been written for appliance dealers to use for a smaller grocery store?

From: Robert P. VanNatta, Esq., VanNatta & Petersen, Gray Building, 222 S. First St., St. Helens, OR 97051:

Just received your issue #3 today. You are late , but this issue is much improved. You are starting to do some of the things I had hoped you would do.

I note in the issue demands for an assembly language tutorial. LIFELINES has been running a monthly series on this topic for a year, mostly written by Ward Christensen who has been writing CP/M public domain programs since 1975. It is the best work that I have seen anywhere. It is equal to or superior to the Heathkit home study course (which I also own).

As far as your list of things that you plan on reviewing, it is clear to me that you are on a path very similar to LIFELINES. For example, in the last year LIFELINES has reviewed in depth at least half of the material listed by you on page 2 and 3 (maybe 75%). I have personally written reviews for them on MailMerge and Spellstar. WordStar has been reviewed and at least two articles have been published on installing WordStar on the Model II. They include some patches which apparently you don't know. These articles are copyrighted by LIFELINES. It seems to me that you ought to find a way to publish (legally) some or all of this information.

I hope you know that the Taranto General Ledger is based, I believe, on the Osborne general ledger as published by McGraw Hill. The original version is available in Source (C-BASIC2) from the CP/M users' group for $8.00, postpaid. I am quite familiar with it. I have been using it (the original) for several years. It takes about 40 lines of code to modify the CMUG version to work on a Model II. (The terminal subroutine must be rewritten.)

I note your intention to review MBASIC Compiler. I would note that BASIC and CB-80 (my favorites) are BASICs specifically written for business applications. They contain a 14 point BCD math package which gives them the math accuracy necessary for accounting programs, etc.

On the double sided drive issue, the ATON documentation...tells how to put a double sided drive IN THE MODEL II TERMINAL!

Regarding your inquiry about drives: my 3 drive (latch type) expansion bay has worked for 2.5 years without trouble. Both drives were defective on my Model 16 and were repaired under the warranty.

Of my "built-in" Model II push button drives, two have worked fine. The first one I bought in March of 1980 was defective and was replaced under the warranty. My "homebrewed" expansion bay (which actually came partially from APPARAT, Inc.) has pushbutton (Shugart 801s) drives and has worked fine. I now use it on my 16 with the edge card connector on Drive B disconnected.

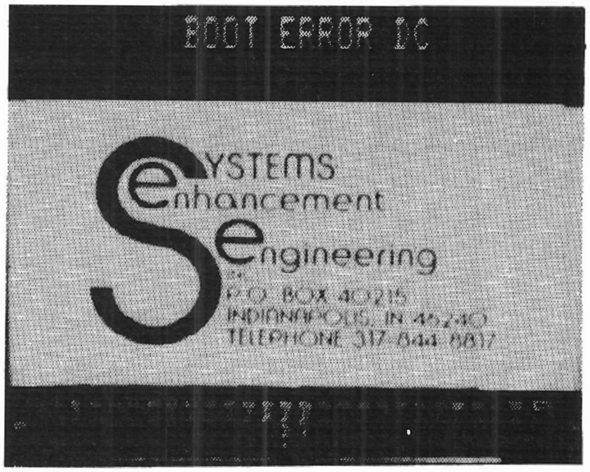
I am convinced that the WordStar patch issue is so important that a comprehensive work on the topic needs to be published forthwith. You sort of insulted the guy who asked for some WordStar patches (Vol. 1, #3, page 16), but I think you are wrong. WordStar is a highly structured program and the patches in the installation routines are fully contemplated by MicroPro. Just because you have to set the switches with DDT because the INSTALL menu does not address the matter doesn't mean that it shouldn't be done.

Osborne general ledger as published by McGraw Hill. The original version is available in Source (C-BASIC2) from the CP/M users' group for $8.00, postpaid. I am quite familiar with it. I have been using it (the original) for several years. It takes about 40 lines of code to modify the CMUG version to work on a Model II. (The terminal subroutine must be rewritten.)

I note your intention to review MBASIC Compiler. I would note that BASIC and CB-80 (my favorites) are BASICs specifically written for business applications. They contain a 14 point BCD math package which gives them the math accuracy necessary for accounting programs, etc.

On the double sided drive issue, the ATON documentation...tells how to put a double sided drive IN THE MODEL II TERMINAL!

Regarding your inquiry about drives: my 3 drive (latch type) expansion bay has worked for 2.5 years without trouble.



Both drives were defective on my Model 16 and were repaired under the warranty.

Of my "built-in" Model II push button drives, two have worked fine. The first one I bought in March of 1980 was defective and was replaced under the warranty. My "homebrewed" expansion bay (which actually came partially from APPARAT, Inc.) has pushbutton (Shugart 801s) drives and has worked fine. I now use it on my 16 with the edge card connector on Drive B disconnected.

I am convinced that the WordStar patch issue is so important that a comprehensive work on the topic needs to be published forthwith. You sort of insulted the guy who asked for some WordStar patches (Vol. 1, #3, page 16), but I think you are wrong. WordStar is a highly structured program and the patches in the installation routines are fully contemplated by MicroPro. Just because you have to set the switches with DDT because the INSTALL menu does not address the matter doesn't mean that it shouldn't be done.

You might tell Barbara that her ideas in the novice column (Vol. 1, #3, page 26) have some fundamental flaws in approach.

Besides my own work, I supervise five other people using WordStar. It is too complicated to expect them to be shuffling around and reorganizing disks. The following procedure works much better:

1.) Prepare a WORKING MASTER of WordStar. This disk will not be used, but will contain the correctly setup operating system and correctly installed version of WordStar and any utilities you might want, such as STAT and PIP, but not the full clutter of system utilities. 2.) Clone this disk a box at a time; i.e., make ten copies. 3.) Hide the working working master. 4.) Dispense the clones to the office staff on an "as needed" basis. 5.) If the staff is doing diverse tasks, urge that they devise a method of dividing their work by subject matter so they will have some organization to their work. 6.) When a disk gets full, urge that it be retired to the bookshelf and that they start with a fresh disk. 7.) When the bookshelf gets full, identify the disks that contain nothing of value and reformat them and start over again. By that time you will probably have a new version of something that you will want to start using anyway.

This approach works quite well if you are working on the single disk, self-contained concept followed by me.

The alternative is to follow that old standby of having a "system disk" and a "data disk" and work them both together. This is more complicated and I believe unnecessary considering the considerable capacity of the Model II/16.

It is the only way to fly with a one drive machine. I have two one drive machines for word processing and have the 4-drive machine available for the occasional multidrive work. The primary assignment of my 4-drive is data processing.

When the disk is very full, the procedure that Barbara outlined won't work anyway because you can't get out of WordStar to make the deletions as she instructed. You must delete from inside WordStar with the control KJ. Also if the directory is full instead of the disk being full, WordStar will simply crash.

If you want to see a genuine bug in WordStar, attempt to mark a column in the Column Move Mode control KN (on) that is about 90 columns wide. HINT: you get there faster than you do with control KX.

From: Stuart R. Hays, Suite 201, 1200 S. Military Highway, Chesapeake, VA 23320:

I hope that your magazine will maintain a positive approach. As lawyers, we found ourselves on the forefront of using the Model II in a law office in Tidewater, Virginia. The outstanding features of Radio Shack at that time were service, contract maintenance, sufficient software to do the job, and very importantly the lowest price. Radio Shack did more for us than its competitors including systems at twice the price.

We weren't programmers, we had no interest in programming, we simply wanted a computer that would efficiently and effectively do word processing, keep our billing records, and the like. We found in careful study that SCRIPSIT is a document oriented word processing program, the majority of the others are page oriented word processing programs. In our judgment SCRIPSIT is the superior word processing system for attorneys if the "Assemble" and "Merge" functions are carefully used in a very systematic approach. We have created a complete system and in comparing our system, other attorneys are constantly amazed at how far superior our simple Radio Shack is compared to their very complex and very expensive systems.

We are using now Profile Plus with the addition of Proforms, Archive, and Prosort. Again we have found this combination to be excellent and superior in our judgment to the other systems that we have previously tried, our friends had tried, or we had written for us. We found that most people in our area spent their time criticizing rather than learning what they had purchased and pushing their purchase to its limits and extremes.

At this time we are using Profile Plus in one form or another for accounts payable, accounts receivable, our client tracking, our time billing, our depreciation scheduling, to compute and print our HUD forms, to do all of our collection accounts, and to maintain our corporate filings for State Corporation Commission. That really is not a bad collection. We have had one program created in BASIC for us to use, and that was a transfer program to allow us to transfer Profile fields from one file to another. With this we will be able to construct a general ledger with Profile, and at the appropriate times during the year make the appropriate moves.

From: Douglas W. Raymond, 23 Martha Road, Orinda, CA 94563:

Thank you for keeping *two/sixteen* going (and in style) against monumental odds.

The September-December issue contains a letter on a subject which has lately been of great concern to me: the dearth of manufacturing software. The letter in question is on page 14, from Vincent Brown. I have written to Mr. Brown to share my problems with him, but there may be other people out there with manufacturing management problems. Readers, feel free to write me directly if you have solutions to this or similar problems, or if you want to keep track of my progress...

I manage Technical Operations at Zehntel Production Services. We produce fixtures, programs, and turnkey application packages for the automatic test equipment industry. The two departments I manage have about 20 engineers and technicians in them who are using ten or so machines on two shifts to process about 30 jobs a month. At any given time there are about 100 jobs in queue. We have no problem getting all our work done, but we would like to be able to do a better job of scheduling the work and the resources.

We would especially like to be able to predict ship dates, as our customers are very interested in knowing when they can count on putting their testers into production. (Our Accounts Receivable people are also interested...) After that, we would like to be able to play "what if...?" with our process model, to see what life would be like if we added another machine, or hired another programmer, or landed an enormous order, etc.

I agree with your stand that most of what is published for manufacturing use is simplistic. I have explored the "sophisticated" project management packages Plan/80 and VisiSchedule (I think they have a common ancestor...) and found them wanting, too. They can handle a small number of complex projects involving arbitrary resources. Manufacturing organizations such as machine shops, on the other hand, process large numbers of fairly simple projects with fixed resources.

I think the scheduling problem is definable: The jobs (hopefully) keep coming, at a variable rate. Each job can be broken into a chain of events, each event requiring a certain class of manpower and a certain type of machine. Our manpower pool is constrained by what we can talk Upper Management into letting us hire or fire, i.e. manpower is an independent variable. Our machine population is likewise a "given," in that it cannot be changed arbitrarily.

All I REALLY need to know is if I accept a rush order in January, do I need to call the customers about delaying the jobs I had planned to ship in February. It would be worth a lot to me to be able to answer this simple question often, easily, and with confidence. If any of you "closet" manufacturing managers out there has worked this problem out, I would like to know about it, and so would Mr. Brown.

From: Sandra Kulczyk, Analytical Development Corporation, P. O. Box 429, Monument, CO 80132:

I have received three copies of *two/sixteen* and have been very pleased with the information.

I have several questions for your staff and readers: is anyone out there using the Model II/16 in an analytical laboratory? Has anyone interfaced instruments to the computer using A/D converters? I am especially interested in a multi-user set up with password protection. Is software available to handle the data from instruments, allow data manipulations, and print formatted reports?

It is very nice to have a magazine devoted to the Model II/16. Keep up the good work.

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

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

The mysteries of DEFUSR$_n$, USR$_n$ *and* VARPTR

Jim Kloosterman
Post Office Drawer 48
Rogers City, Michigan 49779

DEFUSRn, USRn, and VARPTR are probably the most difficult understand functions in BASIC — and often appear in the same program. To really understand them, we must explore some of the details of the internal operations of Interpreter-BASIC.

The Interpreter "reads" your BASIC program, one statement at a time, and "interprets" each one. How? Well, it analyzes that one statement to identify a number of individual tasks, then it selects the proper machine-language routine for each task and activates them in the proper order. When those are completed, it proceeds to "read" your next statement, and so on.

Suppose you were an interpreter! You "read" the statement X1=4 and analyze it. The identified tasks might be:

(1) Has the variable-name X1 occurred before? That is, is there a variable-table entry already for X1? If so, skip to (4);

(2) Determine the type of variable by checking for a suffix character to the variable-name (%,!,#, or $) or, if none, if the letter X has been previously stored in a DEFINT, DEFSNG, DEFDBL, or DEFSTR table of first-letters or, if none, default to single-precision, floating-point type;

(3) Establish a variable-table entry to indicate variable type (integer, single, double or string), the name-characters X1 and the current value of zero (or null) in the appropriate data format (in the case of a string variable, current value of zero will represent the LENgth of the string and two additional bytes will be reserved in the table-entry to show the address where the first-character of the string will be stored (zero for now);

(4) Convert the stated value of 4 decimal to binary in the appropriate data format for the variable type of variable X1;

(5) Store that as the current value in the X1 table-entry.

After all that, the Interpreter is ready to "read" the next statement and deal with it. Sound complicated? That was a gross simplification! The identification of the tasks to be performed and their proper order, as well as detecting syntax-errors, etc., are all steps we just breezed through here but are VERY complex.

The important point is that the BASIC-Interpreter deals with a lot of internal tables; one of them is that variable name table with entries for every variable somewhat as described.

Fire-up your Model II on a TRSDOS BASIC disk you can afford to lose — just in case things go terribly wrong! That is, use a "clean" disk or one for which adequate back-ups exist for EVERY file.

When TRSDOS READY appears, type

DEBUG ON <ENTER>

After the next TRSDOS READY, type

BASIC -M:61239 <ENTER>

At BASIC's Ready, type the program in LISTING 1 (p. 29).

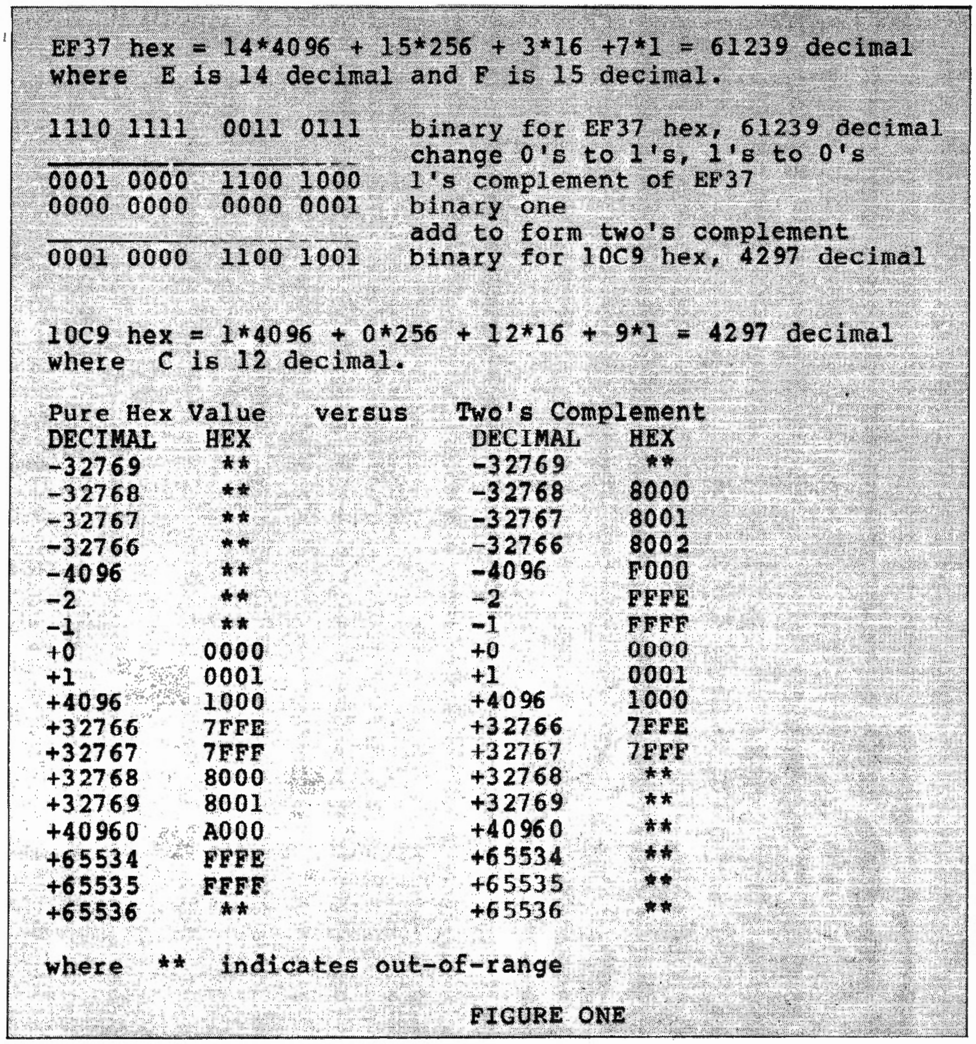
DO NOT RUN YET! Proofread CAREFULLY! To get the addresses I cite below, you must copy the program EXACTLY without even one missing or extra space-character.

After proofreading, type

SAVE"VARPTR/BAS" <ENTER>

And now RUN the program. The first line displayed by the program should read: X = -4297 or 61239 in decimal or EF37 hex. Note that there are two ways of interpreting the hex value EF37. Considered a signed integer-value in two's-complement form, it is -4297 decimal. Considered as a pure hex number, it has the value of 61239 decimal. (See FIGURE 1—next page.)

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



```
EF37 hex = 14*4096 + 15*256 + 3*16 +7*1 = 61239 decimal
where  E is 14 decimal and F is 15 decimal.

1110 1111  0011 0111     binary for EF37 hex, 61239 decimal
                         change 0's to 1's, 1's to 0's
0001 0000  1100 1000     1's complement of EF37
0000 0000  0000 0001     binary one
                         add to form two's complement
0001 0000  1100 1001     binary for 10C9 hex, 4297 decimal


10C9 hex = 1*4096 + 0*256 + 12*16 + 9*1 = 4297 decimal
where  C is 12 decimal.

Pure Hex Value    versus   Two's Complement
DECIMAL    HEX              DECIMAL    HEX
-32769     **               -32769     **
-32768     **               -32768     8000
-32767     **               -32767     8001
-32766     **               -32766     8002
-4096      **               -4096      F000
-2         **               -2         FFFE
-1         **               -1         FFFF
+0         0000             +0         0000
+1         0001             +1         0001
+4096      1000             +4096      1000
+32766     7FFE             +32766     7FFE
+32767     7FFF             +32767     7FFF
+32768     8000             +32768     **
+32769     8001             +32769     **
+40960     A000             +40960     **
+65534     FFFE             +65534     **
+65535     FFFF             +65535     **
+65536     **               +65536     **

where  **  indicates out-of-range
```

FIGURE ONE

Normally, Interpreter-BASIC variables of the integer type are in two's complement form. The maximum range is -32768 decimal or 8000 hex through +32767 decimal or 7FFF hex. The left-most bit serves as a "sign-bit": 0 for positive values and 1 for negative values. Sixteen bits affords exactly 65536 decimal, different binary-patterns. Those are assigned with 32768 decimal, negative values and 32768 decimal, positive values. However, zero is one of the positive values so that limit is only +32767, NOT +32768!

The signed, two's complement scheme works nicely, if a somewhat bit confusingly, most of the time. However, for addresses in RAM, it is more convenient to use pure hex: 0000 to FFFF hex for 64K RAM, which actually has 64*1024 = 65536 decimal bytes. Convert negative two's-complement decimal-values by adding 65536.

Enough theory! Let's see where Interpreter-BASIC has tabled our variables. The display should say "VARPTR(X) = 6DB6" (if your TRSDOS and BASIC work like mine and if you copied the program EXACTLY, otherwise use your values).

That statement SYSTEM"DEBUG" at the end of line 100 has placed you in DEBUG. Press

<X> (the X key)

to be certain you are in hex addressing mode.

Press

<M>

The characters A = will appear, asking you for an address. Type the 6DB6 indicated as the hex value of VARPTR(X). 6DB6 will now appear atop the left-most column. The sixteen hex-character pairs appearing to the right of that are the hex-values in the sixteen bytes of RAM beginning with address 6DB6. The first two will show EF 37 and that is the value of our integer-variable X! Remember, always LSB (Least Significant Byte) FIRST, followed by the MSB (Most Significant Byte).

Address 6DB6 and 6DB7 are where the Interpreter has tabled the value of X — as indicated by the VARPTR(X) function. Simple, huh?

Now press

<ESC>

and then

<D>

The left-most address column will change to equivalent decimal values. Our 6DB6 becomes 28086, but the hex values of the RAM bytes remain unchanged.

Press

<M>

and type

28083

which is the previous decimal-address minus three. Now the display will show the bytes 02 58 00 37 EF ... That 37 EF is still the value of X. The 02 represents the Interpreter's note that this is a Type 2 (integer) variable. The 58 00 indicates it has the name X with no second character. 58 hex, 88 decimal, is the ASCII code for the letter X.

Look in the area away over to the right on the display and you'll see another, smaller "map" of the RAM values just like the hex "map" but this time showing ASCII codes — where the code is a displayable character and a period for all others. See the .X.7. there? That's our X. (The 7 though is just coincidental; it's the ASCII character for hex 37.)

That's how the interpreter "keeps track" of our variable X. Single and double precision, floating-point-form variables are handled similarly — except the value storage takes more bytes and is in mantissa-and-exponent form. Read your Owner's Manual, TRSDOS Section for DEBUG details and in the BASIC Section, for VARPTR details. Integer-ARRAY variables are both simpler and more complex. Our display indicates that the hex value of VARPTR(XD(0)) is 6DD2 and for XD(1) is 6DD4 and for XD(2) is 6DD6, etc. It appears each element of the integer-array is stored separately, end-to-end in sequence from 6DD2 through 6DF0 with two-bytes for each value.

Press

<ESC> and <X>

to get back to hex addressing. Press

<M>

and type

6DD2

Recall that our program assigned the value of I to each of the variables XD(I) from XD(0) through XD(15). Note the displayed RAM: 00 00 01 00 02 00, etc. THOSE are our values, LSB first and two-bytes per variable. A total of 16 values, 32 bytes, the last one beginning at 6DF0 showing 0F 00. How about that!

Now, quick, what's 6DD2 minus 8? It's 6DCA. Count backward from Dty-2: Dty-1, D-teen, Cty-F, Cty-E, Cty-D, Cty-C, Cty-B and Cty-A. Whoa, that's eight! Okay, type 6DCA. The DEBUG display will show

```
02 58 44 23 00 01 10 00   00 00 01 00 02  00 03 00
```

The latter eight values are the same as before, representing current values of variables XD(0) through XD(3). What are the others?

The 02 in 6DCA is the Interpreter's note that these are Type 2 variables (integers). 58 44 are the ASCII codes of the variable name. See the letters XD corresponding away over to the right? My Manual would have me believe the 23 00 is the "total number of elements in the array." That's not quite correct! We know there are sixteen elements in the array, (0) through (15). 23 00 in hex converts to 0*4096 + 0*256 + 2*16 + 3*1 = 35 decimal (LSB first, remember!). Not coincidentally, 35 decimal bytes is exactly the number of bytes following the 23 00 to the array! For a one-dimensional integer-array, the number given is always twice the "total number of elements in the array" plus three. In this case, that is 2*16 + 3 = 35 decimal or 0023 hex.

Next is a one-byte value, 01 here, indicating the number of dimensions in the array (number of subscripts). Finally, 10 00 represents 16 decimal, the depth (subscript-range) of our array.

Now, let's consider a string-variable. A big complication is that string-variables in TRS-80 Interpreter-BASIC may be of ANY length from zero to 255 decimal characters! To list them one after another in a table would mean that room for 255 characters would have to be provided for each one. Or, the whole table would have to be stretched/shrunk as lengths changed. Instead, a table is used for the reference information for each string-variable — but the actual string-values are stored elsewhere! Let's peek!

Our display indicates the hex value for VARPTR(S1) is 6DBB. Type

6DBB

You will see

```
06 89 6C ...
```

That indicates the string is six characters long and stored beginning at address 6C89 (LSB first, as usual!). Type

6C89

You will see

```
42 41 53 49 43 20 ...
```

Look to the right, and you will see those six hex bytes are the ASCII codes for the letters BASIC and one space. The space is a part of the string S1 but the quote-mark is NOT: It has only six characters. In this case the string is in the source-code area of the RAM and part of the stored line 20 of our BASIC program. Type

6DB8

to "back-up" the display by three from the VARPTR address of 6DBB. Now you see

```
03 53 31 06 89 6C ...
```

The 03 indicates a Type 3 string-variable. The 53 31 are the ASCII codes for variable-name, S1, as you can see in the ASCII display away over to the right.

Our program indicates hex VARPTR(S2) is 6DC1 so type

6DBE

to get three ahead of that. You will see

```
03 53 32 0F 95 63
```

Again, the 03 53 32 represent Type 3 and name S2. The 0F indicates the string has fifteen characters. (See why you are limited to 255 characters per string-variable, maximum? There is only one byte of the table allocated to store the length: FF hex or 255 decimal is the maximum number that can be noted there!)

The 95 63 is the address where the string begins, 6395 hex. Type

6395

You'll see

```
73 74 72 69 . . .
```

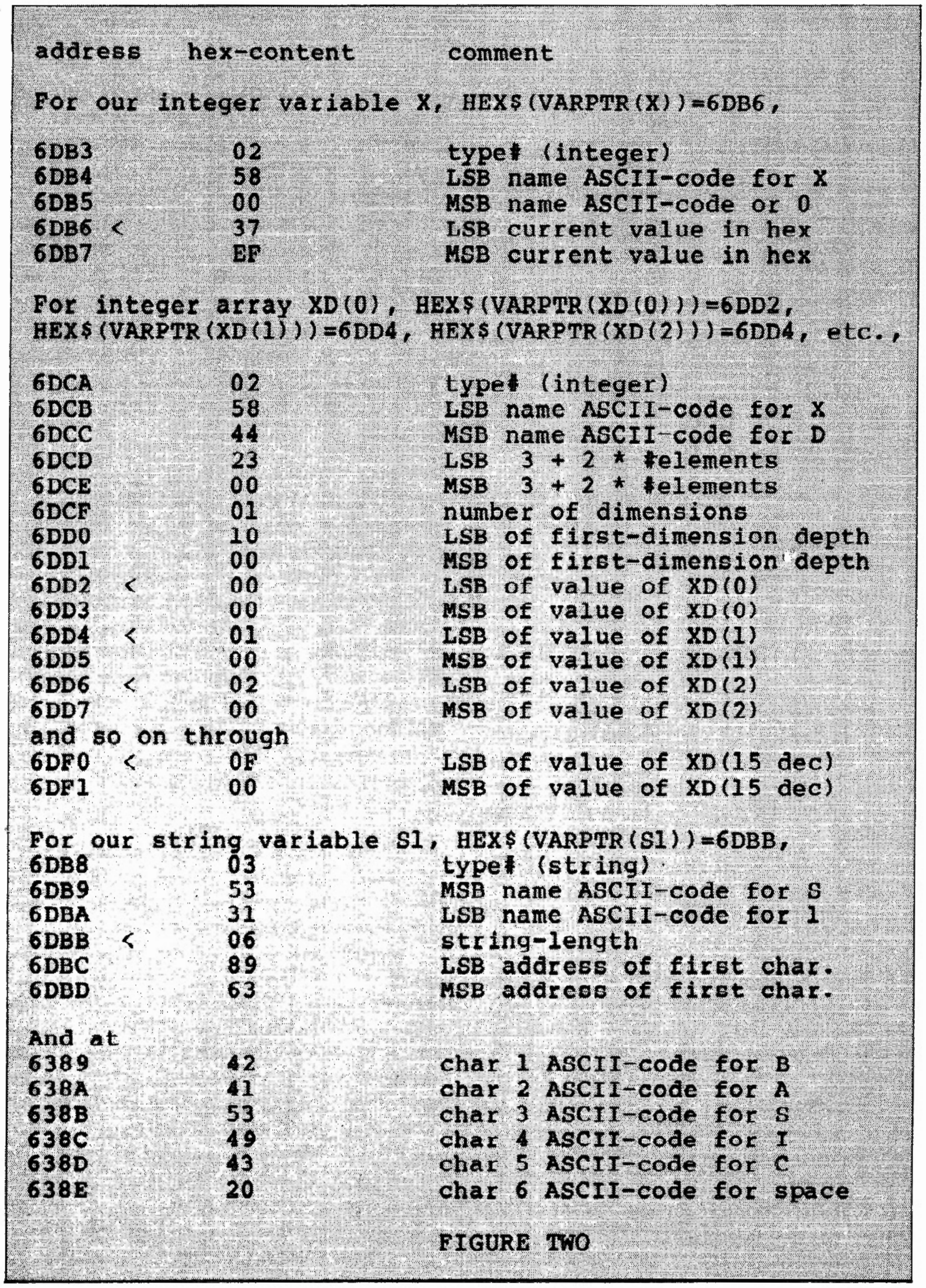
the fifteen ASCII codes of the string-variable per the display at the far right. Again, the string is actually stored in the source-program area.

Finally, type

6DC4

three less then hex VARPTR(S3) at 6DC7. You will see

```
03 53 33 15 22 EF . . .
```



```
address    hex-content        comment

For our integer variable X, HEX$(VARPTR(X))=6DB6,

6DB3            02              type# (integer)
6DB4            58              LSB name ASCII-code for X
6DB5            00              MSB name ASCII-code or 0
6DB6 <          37              LSB current value in hex
6DB7            EF              MSB current value in hex

For integer array XD(0), HEX$(VARPTR(XD(0)))=6DD2,
HEX$(VARPTR(XD(1)))=6DD4, HEX$(VARPTR(XD(2)))=6DD4, etc.,

6DCA            02              type# (integer)
6DCB            58              LSB name ASCII-code for X
6DCC            44              MSB name ASCII-code for D
6DCD            23              LSB  3 + 2 * #elements
6DCE            00              MSB  3 + 2 * #elements
6DCF            01              number of dimensions
6DD0            10              LSB of first-dimension depth
6DD1            00              MSB of first-dimension depth
6DD2  <         00              LSB of value of XD(0)
6DD3            00              MSB of value of XD(0)
6DD4  <         01              LSB of value of XD(1)
6DD5            00              MSB of value of XD(1)
6DD6  <         02              LSB of value of XD(2)
6DD7            00              MSB of value of XD(2)
and so on through
6DF0  <         0F              LSB of value of XD(15 dec)
6DF1            00              MSB of value of XD(15 dec)

For our string variable S1, HEX$(VARPTR(S1))=6DBB,
6DB8            03              type# (string)
6DB9            53              MSB name ASCII-code for S
6DBA            31              LSB name ASCII-code for 1
6DBB  <         06              string-length
6DBC            89              LSB address of first char.
6DBD            63              MSB address of first char.

And at
6389            42              char 1 ASCII-code for B
638A            41              char 2 ASCII-code for A
638B            53              char 3 ASCII-code for S
638C            49              char 4 ASCII-code for I
638D            43              char 5 ASCII-code for C
638E            20              char 6 ASCII-code for space
```

FIGURE TWO

That's Type 3; Name characters S3; Length 15 hex or 21 decimal characters; Beginning at address EF22. Ah! That's in high RAM, the string "scratch" storage area. Why is this string up there when the others were not? Because we did not specifically assign S3 but forced Interpreter-BASIC to "calculate it" by adding the string S2 to the end of string S1 by stating S3 = S1 + S2 in line 30. Type

EF22

and you will see

42 41 53 . . . . etc.,

the ASCII codes for "BASIC string-variable." That is 21 decimal characters long just as the table indicated.

Note one more thing. Type

EF36

The first byte will be 65, the hex ASCII code for the last letter e of our S3 string. Now press

<ESC> and <D>

The address, now in decimal, will be 61238. Remember when we loaded BASIC and indicated -M:61239 to set the high-limit on the Interpreter's use of RAM! That ordered it NOT to use any byte past address 61238 decimal, EF36 hex. And that is exactly what it did. It began building the string "scratch" storage-area backward from there — using just up to but not the byte with address 61239 decimal, EF37 hex!

OKAY! So, in summary, VARPTR (numeric-variable-name) yields a RAM address in an Interpreter-BASIC table where the LSB of the current value of that variable is stored. The other, more significate bytes and the exponent, if any, follow.

In similar fashion, VARPTR (string-variable-name) yields a RAM address in an Interpreter-BASIC table where the one-byte current-value of the length of that string is stored; the next two-bytes indicate the address in RAM where the code for the first character of the string is stored. Character codes are stored sequentially, one-byte-each, for the given length of the string. See FIGURE 2.

For non-array variables, the variable type-number and the ASCII-codes for the variable-name are stored in the three bytes ahead of the address yielded by VARPTR (variable-name). In the case of the first element (subscript 0) of an array variable, the address is preceded by eight or more bytes, giving the variable type-number and root variable-name plus other information about the array. The value-bytes of a numeric array-variable are stored sequentially, beginning with the first element of the first subscript (dimension).

For string array-variables, the three-byte groups of length and address-of-the-first-character-code are stored sequentially beginning with the first element of the first subscript (dimension).

So VARPTR yields a RAM address in an Interpreter-BASIC table you can use with DEBUG to find the current value of the variable and other information about it. WARNING: As you might expect, the current value of any variable may change as the program is being executed by the Interpreter and, as you might not expect, the RAM address of the table entry for some variables may change too! At some point in a program you might find PRINTHEX$(I%(4)) yields value 1ABC and PRINTHEX$(VARPTR(I%(4))) yields address 7CDE. Later, the value of the variable may have been changed to 4321. Still later, the variable may still have the same value, but the address may change to 7CE3. Sometimes, that's confusing!

Interpreter-BASIC uses dynamic variable-space allocations; that is, the specific allocation of RAM for various tables can, and often is, changed during the execution of a single program! The current characters of string-variables are "moved about" most frequently. However, the tabled entries of all arrays are moved too whenever a new, non-array variable is introduced.

The address provided by VARPTR is always accurate at the time it's executed. Do NOT fall into the trap of storing that address and then later assuming it is still correct. Things may have been moved about meanwhile! Make use of the address as soon as possible. DO NOT introduce any new variable-names in between VARPTR and address use. If you need "that same address again" later in your program, use VARPTR again — just in case.

To see this occur, exit DEBUG by pressing

```
<ESC>
```

and then pressing

```
<C>
```

The BASIC Ready printout and flashing cursor will be at the bottom of your screen. Type

```
CLEAR:CLS:DEFINT   I,J,K:   DIM
K(2)   <ENTER>
```

Next type

```
?HEX$(VARPTR(K(0)))   <ENTER>
```

After the hex address is displayed, type

```
I=VARPTR(K(0))   <ENTER>
```

and then type

```
?HEX$(I)   <ENTER>
```

Note that the value from I is NOT the same value displayed before, yet BOTH should be the RAM address of the LSB of the value of variable K(0). Type

```
?HEX$(VARPTR(K(0)))   <ENTER>
```

OOPS! When we did that before, we got one answer; now we get a different one. What is going on? Try typing

```
J=5   <ENTER>
```

Now, repeat once more by typing

```
?HEX$(VARPTR(K(0)))   <ENTER>
```

Still another answer! What is happening is that the entire table for the integer array K(0),K(1),K(2) has been shifted upward in RAM each time a new non-array variable is introduced so that the new table-entry can be added UNDER it. Both the integer variables I and J were newly introduced causing shifts! Each time, VARPTR(K(0)) properly reported the then current address of the LSB of value K(0) in the shifting table of array variables.

In progressively higher RAM addresses, the BASIC-Interpreter stores tables of all non-array-variable data following the array-variable data. For "direct-quote" strings, the characters of the string are left in the program-storage area. All others strings are placed end-to-end with the last character of the first such string at the highest RAM address available to BASIC and building down from there. Certain intermediate strings developed within the process of string manipulations are also stored there. What's more, anytime any string-variable appears on the left side of an equation (unless within a MID$ expression), it is written as a totally new string in that hi-RAM area!

The variable-tables build upward to higher RAM addresses as the scratch strings build downward to lower RAM addresses. The CLEARn statement sets the line between them. If variable-tables reach that, you get an OUT OF MEMORY error. However, if scratch strings threaten to cross the line, the Interpreter takes time out to reorganize the scratch strings area. That can cause the dreaded "computer dead" occasions when your Model II/12/16 responds to nothing you do upon the keyboard, not even <BREAK>, for what may be a frighteningly long time! (RESET will work but ...) Reorganizing is done by checking each stored string. If it is "no longer active", then every string-character below it will be moved up to use the area formerly occupied by that string. When it checks the next string, that one may be okay. But the next may be another "no longer active" string. Again, all of the string-characters below it will be moved up. Obviously, many of the string-characters may be moved many times. That's why this process may take considerable time! If successful in making room by eliminating left-over, "no longer active" strings, the Interpreter will proceed (eventually). If the line is still threatened, the OUT OF STRING SPACE error will be given.

Now you know some of the reasons Interpreter-BASIC is slow relative to machine-language! Even when not "garbage-collecting" to reorganize the string area, it is continually "rooting around" in the various variable-table entries, identifying each variable name it encounters, checking the variable type and retrieving or storing properly formatting values. The Interpreter spends much more time doing that than it does doing "real computation." That is a price we pay for an Interpreter that is conversational and responds instantly to changes in the source program.

There's nothing terrible about it; it's simply a trade-off of execution speed for flexibility, which is great for program check-out, not bad when running small programs, but horrible for routine-production runs of large programs. PLEASE NOTE: These good/bad features are not characteristic of the language of BASIC but only of Radio Shack's version of Interpreter-BASIC software. Other BASIC interpreters might run slower or faster depending upon their goals. Likewise, a compiled BASIC may be slower or faster or much faster, depending upon the compiler design goals. Generally, the closer compiled object-code is to true hex, machine-language code, the faster it will run! Some compilers, such as the TRS-80 BASIC Compiler, produce object code so far from machine-language that a special interpreter is used to execute the COMPILED-object-code! They are not very fast — but may be RAM-use efficient and highly flexible.

Why should we ever care where or how the BASIC-Interpreter stores the values of our variables? Aside from understanding some of the eccentricities, we care because of one very special BASIC function: USRn. Variable-name = USRn(argument) is the format for use where n is a number from 0 through 9 and the variable name and argument must always be of the same type (integer, etc.). Argument may be an expression or a variable-name. What does USRn do that could make it worth all this? Very simply, it allows you to include machine-language programs within a BASIC program!

Why do that? Because machine-language programs can do many things BASIC can't and because they can do those and many other chores much, much faster than Interpreter-BASIC or even a compiled BASIC! Read the details of USRn in your Owner's Manual. Note how USRn is involved with the Interpreter-BASIC tables! For now, just be aware that those registers A, HL and DE are places internal to your Z-80 chip. All machine-language programs have fast access to and can manipulate those registers.

A DEFUSRn function absolutely must precede any use of the USRn function with the same n. (Read up on that too in your Owner's Manual.) The usual format is DEFUSRn = &Hhhhh where n is 0 through 9 and hhhh is a hex value 0000 through FFFF. DEFUSRn establishes some RAM address (in yet another Interpreter-BASIC table) for use by the USRn of the same n. When a USRn function is encountered by the Interpreter, COMPLETE CONTROL of the computer is handed over to a machine-language command beginning at the DEFined address.

For example, if your program includes the statement, DEFUSR5=&HEF12 then any following statement including the USR5 function will cause the Interpreter to hand over control of the computer to a machine-language command at RAM address EF12 hex.

BE WARNED: the command had better be there and what it does and what follows had better make sense! If not, you may need to resort to your RESET switch to stop the internal chaos that may ensue. One tiny error can, and often does, cause a machine-language program to tear-up itself or your program or the BASIC-Interpreter or the various tables or TRSDOS or any combination of them! Often, finding bugs requires reconstructing what went wrong just before the computer crashed.

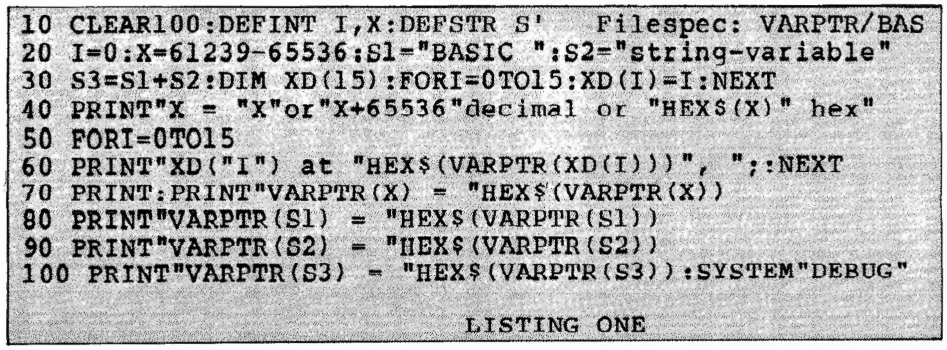
If everything works correctly, the machine-language program will "do its thing" and then return control of the computer to the BASIC-Interpreter. It will go on about its business with the statement following the one involving USRn. Model II BASIC allows as many as ten USRn functions to be active at the same time, each with a different machine-language beginning-address as defined by DEFUSRn. You may use a USRn as often as desired after you have defined the appropriate address once in a DEFUSRn with the same n. You may use more than ten machine-language programs in one BASIC program by redefining some or all of the USRn with new and different DEFUSRn statements — but you may have no more than ten defined at any one time.

In addition to transferring control of the computer to the pre-defined RAM address, the USRn function does some additional, important work! It may pass data to a machine-language program before handing over control and, on regaining control, may accept data passed back from the machine-language program.

Why is that important? Often, certain information needs to be conveyed to the machine-language program to properly do its job. For instance, we may have a machine-language program that can "peek" in any given RAM address and learn the value of the byte there. To use the program, we must pass it the address we wish to peek into. Then, the program must pass back the one-byte value found there. USRn can do just that!

Note: Because of the dynamic (ever-changing) nature of the Interpreter's variable-storage area, no fixed address exists where the machine-language program might look for such a BASIC value. Only the Interpreter knows where BASIC values are stored at any given moment! The Interpreter must pass along the value if the machine-language program is to make use of it.

The statement DEFUSR0 = &HEF37 will set up a table-entry so that the X = USR0(Y) will transfer control of the computer to a machine-language command beginning at RAM address EF37 hex. (Variables X and Y must be of the same type.) When the statement X = USR0(Y) is executed, before control is handed over to address EF37 hex, the Interpreter will first load the Z-80's A-Register with the type-number of variable Y. If Y is a numeric variable, the HL-Register will be loaded with the address of the first byte of the argument storage area (ASA). If Y is a string variable, then the DE-Register will be loaded with the address of length of the string in the variable-name table. Only then will control of the computer be passed to the machine-language program at RAM address EF37 hex. Also, upon regaining control, processing of the USRn will continue (for numeric variables) by placing the then-current value in the ASA in the variable-table as the value of X — before the Interpreter proceeds to the next statement. If the machine-language program did not alter the ASA, then the value of X will be equal to the original value of Y. In any case, the value of 0 will be unchanged.



```
10 CLEAR100:DEFINT I,X:DEFSTR S'   Filespec: VARPTR/BAS
20 I=0:X=61239-65536:S1="BASIC ":S2="string-variable"
30 S3=S1+S2:DIM XD(15):FORI=0TO15:XD(I)=I:NEXT
40 PRINT"X = "X"or"X+65536"decimal or "HEX$(X)" hex"
50 FORI=0TO15
60 PRINT"XD("I") at "HEX$(VARPTR(XD(I)))", ";:NEXT
70 PRINT:PRINT"VARPTR(X) = "HEX$(VARPTR(X))
80 PRINT"VARPTR(S1) = "HEX$(VARPTR(S1))
90 PRINT"VARPTR(S2) = "HEX$(VARPTR(S2))
100 PRINT"VARPTR(S3) = "HEX$(VARPTR(S3)):SYSTEM"DEBUG"
```

LISTING ONE

Regarding the HL Register during USRn, my Owner's Manual states: "When the argument is a number, this register points to the argument storage area (ASA) described later." This is true, but don't confuse the ASA with the variable-table entry described earlier under VARPTR! The ASA is a separate, "temporary value" table in a different area of RAM than the variable-table entries. Only in the case of a string variable does the HL-Register on a USRn actually hold the address of the variable-table entry. That is, only on a string variable will S$=USR9(S$) yield (HL)=K where K=VARPTR(S$). All other variables use the ASA.

To add to the confusion, the ASA holds values in a similar format to that of a variable-table entry for the different types of variables. Let K be the address yielded by VARPTR(A) and ASA be the address placed in the HL-Register on USRn(A). Note that if A is an integer, both K and ASA point to the LSB of the value of variable A. Also, both K+1 and ASA+1 point to the MSB of the value of variable A. However, ASA-1 through ASA-3 DO NOT contain the same information as K-1 through K-3! Furthermore, if A is a single or double precision variable, ASA points to the exponent while ASA-3 or ASA-7 points to the LSB of the mantissa of the value. In contrast, K+3 or K+7 point to the exponent while K always points to the LSB of the mantissa of the value.

The first thing the machine-language program may wish to do is to verify if X is the type of variable expected — by checking the contents of the A-Register. Or, it may simply assume it to be correct. Next, if A is a numeric variable, the contents of the HL-Register may be used to gain access to the current value of the variable X. If X is a string, the DE-Register may be so used. On completion, the machine-language program may or may not leave the same value in the ASA — or the same characters and the same number of characters in the string.

A REVIEW OF SNAPP VI

" The college educated garbage collector"

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

Anyone who has ever written an application for the Model II, 12, or 16 that makes extensive use of strings and string manipulations knows, perhaps sadly, how Radio Shack's version of BASIC handles the chore of string reorganization.

In a program when a large number of strings has been used, a large amount of available RAM must be set aside for "string space." As a string is defined and then replaced, its old value is not cleared from this string space. For example, if A$="CAT" is replaced by A$="DOG" the three bytes occupied by "CAT" in the string space have not been freed up.

Only when all available space has been used up, either by active strings or by the "garbage" left behind, does BASIC stop what it is doing and go through its string reorganization. Unfortunately in the case when a large number of strings must be in use, the time it takes for the reorganization can be extensive.

# strings	R/S BASIC	CEGC/memory	CEGC/disk
125	1.33	.18	
250	5.1	.42	2.36
500	20	.98	3.87
1000	75	2.34	7.40
2000	294	5.40	14.30
4000	1168	12.40	29.10

SNAPP-WARE to the rescue! SNAPP-WARE has developed a product they have dubbed the "College Educated Garbage Collector" (CEGC). This machine language routine in most applications will replace the BASIC string reorganization with one that is immensely faster.

All the user (programmer) has to do is type in the command

SCMD "CEGC"

and the rest is automatic. Whenever string space must be reorganized, the SNAPP routine will be used to accomplish it. The method used is completely transparent to the operator. Rather than relying solely on string space for the reorganization, this routine will use some unused portion of RAM, and when it's done, will return the RAM used to the free pool. Should insufficient RAM be available, then a portion of the BASIC interpreter will be transferred to disk temporarily and the space thus freed used for reorganization.

Above is a table comparing various benchmark times as given in the manual supplied by SNAPP-WARE.

The times are in seconds. The CEGC/disk represents the situation where there is insufficient RAM available and a portion of BASIC must be transferred out to disk and then back again.

With this routine there are some limitations. Depending on the circumstances, there is an upper limit to the number of active strings that can be defined. This number, in any case, will never be less than 3900 or more than 6400. Whenever the maximum is exceeded — or for any other reason the CEGC decides its routine cannot be used — then control for the reorganization will return to the resident BASIC routine.

The machine langauge routine is user relocatable and can reside above &HF000 so that no user accessible space is lost.

This reviewer has had occasion to write programs that use extensive string manipulations and has found that time delays of 15-30 seconds were reduced to .5-2.0 seconds, making the reorganization almost transparent to the operator.

The performance of this product lived up to all the claims made for it. The documentation was excellent, and its implimentation in any application program was quite simple.

The service provided by SNAPPWARE with regard to questions on their products is excellent.

SNAPP-VI for the
Model II/12/16—$100.00
Call to order: 800/543-4628
(Ohio residents call collect: 513/891-4496)

SNAPPWARE
3719 Mantell
Cincinnati, Ohio 45236

BARBARA'S ADVENTURES IN COMPUTERLAND

Barbara Albert

I don't have to tell those observant few of you out there who may have noticed that I didn't write anything last issue. For those of you who didn't notice, you probably don't care anyway. Either way, I have several reasons for failing to provide my customary, albeit naive, observations on the computer scene.

The main reason is that I didn't have anything to write about. For once, I survived several weeks without precipitating a crisis situation. These situations have always provided the impetus for the Novice Column subject matter in the past. I managed to bumble and fumble along in my own way quite satisfactorily, thank you.

The second reason for creating a gap in my previously incisive coverage is that I'm not so sure I qualify as a novice anymore. When do you draw the line between a novice and a someone who knows their way around in a more-or-less capable manner? Of course, I would never go so far as to call myself an expert or a "seasoned veteran," but I've learned a few things during my months here. I can safely say that the mistakes I make now are STUPID mistakes rather than the mistakes I made before out of IGNORANCE!

For example, I no longer turn on my Model II without turning on the expansion drive first. In fact, the first thing I do in the morning as I walk in the door is turn on the expansion drive. Of course, now I have one of those nifty SeeBee (Systems Enhancement Engineering Boot Error Eliminator) gadgets (see Dick Young's Editorial in the Jan/Feb. issue for more on this device) which protect dumdums like me from blowing disks. But guess what. I've discovered a way to ruin disks in the expansion drive, even with an allegedly idiot-proof SeeBee installed.

One day I was working with a disk in Drive 0 and the expansion drive was turned off, thanks to the SeeBee. Then I decided I had to have a program contained on a second disk, so I inserted the disk in Drive 1, signalled the drive and nothing happened. I realized the drive was not turned on, so I turned it on WITH THE DISK IN THE DRIVE. So you see, nothing is safe when the uninformed are on the loose. Needless to say, that only happened once. But once is enough.

Back to the question of whether or not I am still a novice: Webster's Unabridged says a novice is: "One new in any business, profession, or calling; a beginner; tyro." Under synonyms for *Novice* a general discussion ensues: "NOVICE and TYRO are often used without distinction. But NOVICE commonly suggests inexperience, especially in something to be done; TYRO suggests rudimentary acquaintance, especially with something to be learned; as a NOVICE in golf, or, the veriest TYRO knows better."

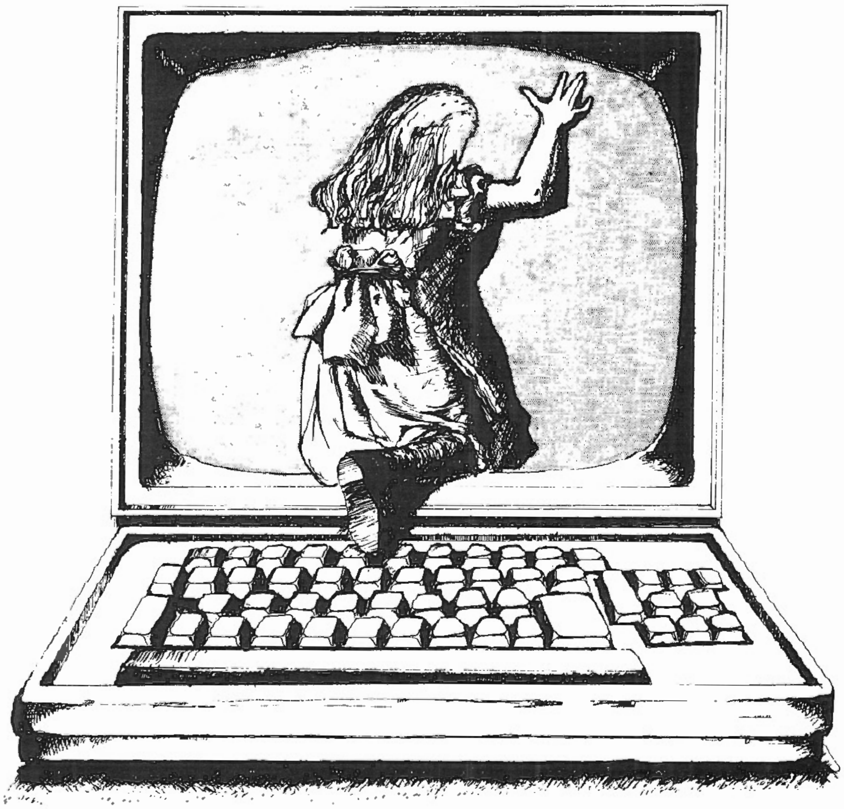
I'm still confused as to my precise status; according to the above, I'm neither fish nor fowl. What is the middle ground? How about *competent*? "COMPETENT: Answering to all requirements; adequate; sufficient, suitable; capable; qualified; fit. Synonym: See ABLE." "ABLE: "Having sufficient power, skill, or resources of any kind to accomplish an object; competent; qualified; capable; Having intellectual qualifications, or strong mental powers; showing ability or skill; talented; clever; powerful..... When used attributely, COMPETENT (implies) the possession of special (often technical) qualifications.

So there you have it. Obviously, through the trials and tribulations of my hours spent in front of the Model II, staring into an often uncooperative, down-right hostile CRT, I've become a competent, able tyro: I have an adequate acquaintance with something to be learned.

I suppose I really shouldn't sell myself too short, or you won't believe my reviews of word processing software and the various enhancements we use to go with it. I do understand how these things work and use them all the time in the execution of my appointed duties.

Scripsit Review

That brings me to a topic I must address: my promised review of Scripsit. Once again, those observant few of you need not be told that it is not in this issue as planned. The more I worked with this enigmatic system, the more frustrated I became. And the more frustrated I became, the more I realized that I could not possibly prepare a worthwhile review for publication until I could gather more facts and experience some more experience.



So this is a call to all you Scripsit users out there: HELP! I am quite serious in my intent to put together a valuable and helpful review/tutorial that you will be able to use when deciding if Scripsit is for you or when you need some special trick to get you through a bad situation or when your secretary tells you she can't do a job you told her to do because she can't find it referenced in the manual, etc. Please send all your tips and tidbits, long or short, to me at P.O. Box 1216, Lancaster, PA 17603, and I will assemble them into some sort of cohesive unit. I know I can count on you.

two/sixteen Is "Paper-less"

TRS-80 Microcomputer News got a lot of coverage in the June 1982 issue of *FOLIO: The Magazine for Magazine Management* because they are producing a "paper-less" magazine. Well, I just want to tell you that we are doing the exact same thing. I don't want you to think we aren't right in the forefront of computer technology, because we are. Our Jan/Feb issue was "paper-less," and no doubt this issue will be, too.

What I mean by a "paper-less" magazine is that everything up until the actual paste-up is done by computers — and I do mean everything!

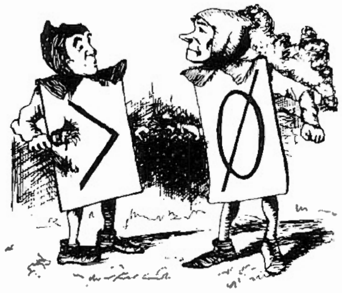
There are some differences between the procedures used by *TRS-80 Microcomputer News* and *two/sixteen*. For example, the *FOLIO* article says: "(Editor Bruce) Elliott prefers (to receive copy on discs* [sic] by telecommunication) so much that he is offering writers a special deal if they send articles on a floppy disc [sic]: They will receive two free discs [sic] in return."

We have been offering an even better "special deal" since May of 1982: You send us an article on disk, and we'll return it to you with a copy of Dick Young's game, STARSHIP/I which sells for $30. (STARSHIP/I is a save-the-galaxy type game based on the STARTREK games that have been played on mainframes for more than ten years.)

Another difference between us and *TRS-80 Microcomputer News* is that they have an in-house typesetting system. Obviously, we are too small for any such venture — not that we wouldn't like to have one, however!

Perhaps the most substantial difference between us and them involves the method by which copy is transferred from the microcomputer to the typesetter. At *TRS-80 Microcomputer News*, "floppies containing the magazine's articles are taken from the editorial TRS-80s and **carried down the hall** (my emphasis) to another TRS-80 that is connected directly to the typesetter." We don't do any "carrying;" we have computers and modems and telephone lines to do our carrying for us.

The articles we receive from our contributing authors are (hopefully!) on floppy disks. We do still receive—and accept— articles that have to be "re-keyboarded." (*Re-keyboarding* is a fancy word for what was known back in simpler times as "typing.") But we do prefer and encourage contributors to send us their articles on diskette. First of all, this method saves us a tremendous amount of time. But secondly, and probably more important to you as readers, this method assures the accuracy and readability of any computer code that may be contained in the article.



All editing work is done directly on the article using word processing. When an article has been edited according to our standards, I have to prepare it for transmission to Innovative Ink, our typesetter. This process is quite simple and involves nothing more than checking the file to make sure that there are no soft hyphens embedded in the text, that the text is not justified, and that no WordStar print commands are left in the file. (The first time we attempted to transmit data, this procedure was not so simple. We converted all files into straight ASCII and removed all carriage returns. This took hours and hours of what we subsequently found out was unnecessary labor. Now we send straight WordStar files.)Then the article is transmitted from my computer to the computer at the typesetter's via modem transfer.

The word *modem* is actually an acronym for MODulator-dEModulator, and what the modulator-demodulator does is quite complex to the electronically-illiterate, such as myself. Essentially, a modem takes the serial digital data from a computer and translates—or modulates—this data into a signal that can be transmitted over telephone lines. A modem can also demodulate data. Demodulation occurs when the modem takes signals received from another computer and translates them back into serial digital informaiton.

This modulation/demodulation is not an observable process. You plug in, connect, turn on, boot-up, dial, and then hope for the best. Whatever the modem does, it does it without allowing you to watch.

We have a Hayes Stack Smartmodem, and it is truly a wonder of modern technology. It is fully compatible with all computers that use the RS-232C terminals and with the Bell telephone system modems, which is lucky for us since our typesetter uses a Dataphone 212A modem from Bell. The Smartmodem is a direct-connect modem as opposed to an acoustic coupler, a real dinosaur that requires the telephone receiver to be fitted into it. With a direct-connect modem, the other lines on a telephone set can be used even when one line is tied up for transmitting data. The Smartmodem also features auto-originate. This means the computer dials its own calls, using either touch-tone or pulse dialing; you can establish data communications without ever touching the telephone. The Smartmodem is capable of giving you a carrier detect signal that indicates when the carrier of another computer is detected; i.e., when your call has been connected. Evidentally, from what I've read, not all modems are able to pass this carrier-connect signal, and thus you have no way of knowing if your data is being transmitted to the destination you have specified or whether it is being sent into thin air. One final feature of the Hayes Smartmodem I want to mention is its audio monitor, better known as a built-in speaker. This little gizmo allows us to listen to the various steps completed by the modem. First, I hear the dial tone, then the number being dialed, then the answer on the other end, and finally the signal for carrier connect.

Without getting into too much technical information here, the Hayes Smartmodem can only transmit and receive at a 300 Baud rate. Baud rate, for the uninformed, is the gauge by which data transfer is measured, and 300 Baud means approximately 300 bits per second, or 300 characters per second. This rate is slow; if you're really with what's-up-to-the minute, you transmit and receive at 1200 Baud. But it costs lots of money to operate at 1200 Baud, and until such time as we are able to forget about such mundane matters as money, we'll continue to send *two/sixteen* articles at 300 baud.

The software we use, which fully supports the Hayes Stack Smartmodem, is the LYNC 2.0 program from Pickles & Trout.

* A *disc* is defined in Webster's Unabridged as "a disk." *Disk*, as used in computer terminology, is an abbreviation for "diskette."

The equipment set-up at Innovative Ink is truly something wondrous to behold. All the components of this impressive system work together to comprise the MCS 8400, Compugraphic Corporation's Modular Composition System. The individual parts of the MCS 8400 are the phototypesetter, the video terminal, the direct-entry keyboard, the hard copy printer, the soft copy preview, the dual disk operation drives, and the Advanced Communications Interface (a fancy name for the modem).

When our copy arrives at the typesetter's, the raw data is stored on disk. Unfortunately, they are limited to using mini-floppies with max storage capacity of 160,000 characters. You can imagine that our magazine takes up more than one of these. When a duplicate copy of our articles is safely tucked away somewhere, they run the data through what is called a translation table. A translation table is custom designed for each of Innovative Ink's customers, and our translation table contains all the little idiosyncracies inherent to WordStar file transmissions. ^S, ^O, and ^B are print codes specific to WordStar, ones that appear repeatedly throughout all our articles. So in our translation table, the Compugraphic system is alerted to change these codes into whatever it is that its system can recognize.

Earlier in this discussion, I mentioned that when we first transmitted copy to Innovative Ink, I took out all hard carriage returns because the carriage returns were appearing in our data as tab returns to the Compugraphics' computer. (Tab returns mean something different to the typesetting computer than carriage returns, and it's important that they aren't used interchangeably.) Now that we know a little more about what is going on, the translation table has been programmed to recognize and painlessly convert the tab returns into carriage returns before any typesetting work begins. In our first endeavors, it took HOURS to remove and change these little buggers.

After all data has been translated, the staff at Innovative Ink begins the actual typesetting process: they imbed certain typesetting codes into the text which tell the typesetting computer what type font to use, what leading to use, what point size to use, how many picas wide and long the columns will be, and any other commands needed to make the copy look like our magazine when it's finished.

The typesetting computer comes with a preview screen, and after the commands have been imbedded in the text, certain buttons are pushed which cause the copy to appear on the preview screen, AS IT WILL LOOK WHEN IT IS LAID OUT ON THE PAGE!! If you don't like the way it looks, you can change it. If the headline is too big, you can make it smaller. If the column is too long, you can make it shorter. In other words, it is completely flexible to your needs. No more frantically rushing copy to the typesetters, waiting for the galleys to be set, proofreading, waiting for finished proofs, and then tedious laying-out and pasting-up the finished proofs. The electronic process is so marvelous compared to the old way, you'll never believe you actually put up with such gross inefficiencies in the past.

After we've decided on the physical layout of the page, the data is sent to the phototypesetter electronically; there is no physical "carrying" done here, either. I don't really understand the actual process that takes place inside the phototypesetter, but the data is printed onto a specially treated, light-sensitive paper via a mercury cathode-ray tube. This paper, which at this stage contains a positive of the negative, is then carried (and I mean that literally; I guess we do have some physical carrying after all) to the hard copy printer where it is developed into the finished product by special chemicals.

When it emerges from this printer, it is the final copy of the article. This copy goes directly from the typesetter's hands into the hands of Tim Schleif, our art director and expert paste-up person. If everything has been done correctly, he can paste the article directly onto the page as it will go to the printer. No cutting! No trying to un-glue columns that were glued on crooked! No trying to cover blurred hand-ruled lines with white-out! This is truly the age of wonders. All Tim needs to do is insert ads and drawings, etc.

We are especially lucky with our situation because Innovative Ink is only about a block and a half away from our office, so I was able to watch over all typesetting and layout operations last time. Fascinating. The Compugraphics typesetting computer functions much like a word processor, so the work they did to our copy was not totally unfamiliar. They are able to move blocks of text around and delete and insert and other word-processing type functions. They are also able to type directly into the copy from their keyboard, so in the unlikely event an error is detected on the screen, we have one last chance to catch it.

In case you noticed any changes in the magazine's cosmetic appearance, this is why. Not only does it put us directly into the state-of-the-art with typesetting, it makes life so much easier for us here on the editorial staff of *two/sixteen*. Chances are that if you ever have occasion to have any typesetting done, you will be given the option to transmit copy electronically. The first time will probably be a little tricky, but in the long run it's worth it. If you want to know more about our set-up, give me a call.

A review of VERSALEDGER II

William Strating, CPA
Small Business Computer Associates
126 East 39th Street
Holland, MI 49423

When I began to write this review, the goal foremost in my mind was to effectively communicate some useful information to you concerning H & E Computronics' accounting package, VERSALEDGER II. To accomplish this goal, it will be helpful for you to understand who I am. To put it very simply, I am an EXPERT!!.

Before microcomputers, it would not have been necessary to tell you what "expert" means. But we ARE dealing with microcomputers, so first we change the spelling from "expert" to "x-spurt". Now all you need to know to get to the "guts" of this review is: how do I define the word in relation to microcomputing? (A.) X = an unknown quantity; (B.) SPURT = a drip under pressure.

I am trying to tell you that my review of VERSALEDGER II is based on the experience I have gained in two + years of using microcomputers for business applications; it is also based on the performance I have witnessed when running other books of account or general ledger type applications packages.

My "Expert" Opinion

I am not going to keep you waiting until the end of the article to find out what I think of VERSALEDGER II.

1. I like it.
2. At $149.95, I think you are getting a bargain.
3. The manual is better than average, but not as good as it could be or should be. If cost is holding down the quality, then I would be willing to pay an additional 15 percent to have more explanatory text and examples added to improve clarity.
4. This application package is no more or no less difficult for an operator to learn than other similar packages I have used which cost up to twice as much. Like everything else, it will take some getting used to before you get rid of the uptight feeling associated with any new procedure.
5. Going back three years to the time when I was putting my books on my first computer, I would have been happy if I had had VERSALEDGER II. It does the job it is supposed to do.
6. I can think of no information you must have from your general ledger not provided for in VERSALEDGER II.
7. After you have decided that VERSALEDGER II is a good program for you, find a day you can set aside to work with the sample data supplied with the package. Shortly thereafter, you will be ready to start running your own data.

Some General Facts And Observations

VERSALEDGER II is a group of interrelated programs that have been "chained together" to provide a means of keeping our books on the computer. Instead of using pens, pencils, calculators, and adding machines to make entries in journals and general ledgers, we can now accomplish the same tasks on the computer with VERSALEDGER II.

The version of VERSALEDGER II I reviewed was run under TRSDOS 2.0a on a single floppy disk drive, Model II, 64k.

The VERSALEDGER II package is written in BASIC, which is important to me and will be to those of you who have developed an understanding of this programming language. You will be able to do some customizing by using the words provided by BASIC to give instructions to microcomputers. The author of this package has been very generous with his use of Remarks in the source code; he is to be commended for this.

On the front cover of the VERSALEDGER II manual, H & E Computronics lists 14 characteristics of the package. One of these says:

"VERSALEDGER II out performs all other competitive systems now available to microcomputer users, at a fraction of the cost (and we offer a 30-day money back guarantee to back up our claim!)." That, my friend, just may not be an overstatement!

This is a good program, with good sample data supplied with it for training purposes and a good manual. The only thing missing is a good attitude on your part, and you are in charge of providing that. I haven't seen any programs at $149.95 easier to learn to operate than this one.

Documentation

The manual has 105 pages in addition to its 51 pages of sample reports (don't misunderstand — this does not mean that you can get 51 different reports!) for a total of 156 pages. There is a considerable amount of white space between the covers, but I would give the manual a grade of B+ or A- relative to others I have seen. Some statements were not too clear to me on first reading, but I am not the brightest person you will ever meet. To sum up my over-all impression of the documentation: you get a lot of "bang for your buck".

The Table of Contents is complete and useful; but why oh why don't manual writers include indexes and/or alphabetically listed operating options, report options, etc.? These features would make locating specific items in the manual so much more pleasant — and easier.

The Issue Of Errors

The eraser found on one end of a pencil and the correction fluid found in nearly every bookkeeper's desk (neither of which are of any use when using a keyboard) is what separates the men from the boys (or women from the girls).

Of course, if you never make mistakes, then I apologize for wasting your time with this information. But if you are guilty of finding erasers and correction fluid useful, you will need to know what to look for when making a decision about a microcomputer program.

I place a very high priority on a business program that acknowledges all of us need to be able to correct mistakes and be forgiven for our errors.

It must not be easy for a programmer to replace the eraser and correction fluid. Also, error trapping must be a costly undertaking. If I paid $500 or more for VERSALEDGER II, I would not be satisfied with the measures taken to help me avoid mistakes and correct them after I had made them. However, for $149.95, I guess they're OK.

Adequate measures have been taken to insure your ability to correct mistakes made during data entry from source documents into your General Ledger. When I was trying to anticipate commonly made mistakes while preparing this review, I didn't find any I couldn't correct in a reasonable amount of time.

A few specific items in this regard may be useful to you in deciding about VERSALEDGER II:

(A.) I had to add some information to a message on one screen to make it clear to me that I had to wait until I went to the next screen before I could change my mind about continuing to print a report.

(B.) If you are using Double-Entry Bookkeeping, there are places where it is easy to make an unbalanced entry.

You will not find a lot of preventative techniques in this program to keep you from making mistakes. You are expected to have read the manual thoroughly, to spend enough time with the sample data to understand how it works, and to have a good understanding of Double-Entry Bookkeeping. All of this is expected before you scrap your present system and go on-line with VERSALEDGER II.

Formatting The General Ledger

The two primary documents every set of books (general ledger) provides are the Balance Sheet and the Income Statement. The quality and usefulness of these two documents is dependent mostly on the skill, knowledge, and experience of the person keeping the books.

In addition, the appearance of the Balance Sheet and Income Statement influences the quality and usefulness of the information. VERSALEDGER II gives you the opportunity to format (determine the appearance of) both documents. Some programs do not permit this.

The first paragraph of the section in the manual explaining the use of Format provides a clue to its value:

"The {Format} option is the heart of VERSALEDGER II's general ledger functions. Understanding this option is the key to successfully using VERSALEDGER II."

On a scale from 1 (easy) to 10 (difficult), you are going to find the Format function about an 8 if you have never formatted any kind of report on a microcomputer before. Even if you have, for sure it is on the hard side of 5. As far as I'm concerned, this rating makes it no different than anything else in the real world.

Supporting Schedules

What are Supporting Schedules? In this context, they are reports (usually in columnar form) used to provide additional detail (support) for information summarized in either the Balance Sheet or the Income Statement. Their purpose is to provide ALL the details for someone who needs to know more than the facts contained in the Balance Sheet and Income Statement. Logically,a business can either include or exclude these Schedules from the financial information furnished to outside parties.

If Supporting Schedules are what you want and expect, you will not find them in VERSALEDGER II.

To borrow a phrase from a former leader of mine, "let me make one thing perfectly clear": the Schedules are not absolutely necessary since you can include all supporting details in the Balance Sheet or Income Statement. However, if you are used to working with Supporting Schedules, VERSALEDGER II could prove to be disappointing.

Source Documents

Let's look at the way in which VERSALEDGER II goes about getting information into the General Ledger and the form its reports take.

For most of us functioning in the business world, source documents are usually one of the following:

1. check stubs
2. checks (remittance advices)
3. receipts we get
4. receipts we give
5. bank deposit records
6. bank statements
7. sales invoices
8. purchase invoices
9. purchase orders
10. and so on

VERSALEDGER II distinguishes between the following types of source documents (everything else is treated the same):

1. check stubs (technically not quite true; I'll explain later)
2. bank deposit slips
3. notices of additions and deductions that the bank has made to your account

Journals

In a manual system of bookkeeping, you will always find journals used to record the information from source documents. You will also often find these journals in computer systems.

You will not, however, find any journals in VERSALEDGER II. From my viewpoint, I prefer to have them and find them useful to my style of keeping a set of books.

I must admit that the journals are not nearly as necessary to a computer system as they are to a manual system. On the other hand, it is not my intention to conduct this review on the merits of what is necessary, but rather to try and point out some of the strengths and weaknesses of VERSALEDGER II from my viewpoint. So chalk up a weakness.

Reviewing, auditing, and tracing the entries made to the set of books by someone else is a necessary part of accounting. Journals make this chore much easier, more pleasant and much less time consuming.

If someone cares to take me to task, VERSALEDGER II provides two optional reports that could possibly be defined very loosely as journals (but not by me). VERSALEDGER II calls these reports:

(A.) Review Checkbook. This report will list all the checks entered, deposits entered, and bank adjustments entered.

(B.) Transaction Register. This report lists everything in (A.) above, plus entries made with the Journal Entry option.

Some of you will be asking, "If there are no journals in which to enter the source documents for posting to the General Ledger, how does the information get into the General Ledger?" Fair enough. Let me tell you.

Menu Driven

VERSALEDGER II operates from four major interactive menus:

—the Main Menu
—the General Ledger Sub-menu
—Financial Statement and Check Register Posting
—the System Menu

VERSALEDGER II includes some other menu type screens to be used when selecting which of the various operating options or reports you want to use.

If you have seen manuals for menu-driven systems that provide actual screen reproductions with dialogue explaining purposes and how-to's (and you really like such features), unfortunately, I must inform you such features are not available in this system.

The manual provides a separate section for each of the menus. Within each of these sections, the purpose of the individual menu and the commands available under each menu are presented one after the other, just as they appear on the CRT.

In order to make selections from a menu, you use the arrow keys. A screen highlight tells you at which option you are positioned, and then a stroke on the ENTER key takes you to the screen related to that selection. From there, you follow the screen prompts to execute the options. You have to move sequentially through the menu options and stop on the one you want.

I have never seen this done before, and I'm not sure I like it. My personal preference is to have the menu selections numbered and then just input the number I want.

The time it takes to recover from a menu selection made in error or one you change your mind about is quite reasonable. In my experience while reviewing this program, the time elapsed when moving from one menu to another was 30 seconds. While I wouldn't rate this time as fast, neither would I rate it as slow in relation to the cost of VERSALEDGER II. I own programs costing five times as much that have no better performance in this respect.

Cash Vs. Accrual Accounting: The Check Register

This General Ledger package seems unique to me in its construction. It is "chained together" in such a fashion that the author suggests you may not even want to use the General Ledger application. He suggests instead you should use what he calls a Check Register.

This feature could be confusing. Let me try to explain it in the simplest way possible.

The easiest transactions to be entered into this system are cash transactions brought about by writing checks on the disbursement side (as opposed to currency) and depositing receipts on the collection side (either checks or currency). It may seem elementary to say so, but these are the very kinds of transactions you would enter in a checkbook record of receipts and disbursements.

Since the author of VERSALEDGER II has provided routines to print out records of these transactions which contain all the information you need to know about them, he has suggested that you might like to call this one-half of the package a Check Register, and possibly you may have no need for the other half, called a General Ledger.

VERSALEDGER II provides an added feature not often used in checkbooks — a provision for account distribution of all of your check and deposit entries. There is also a report called List By Account which groups entries by account number and lists the entries to each account.

A couple of observations are in order:

(A.) Although this feature provides what could almost be called a "cash basis" accounting system, it appears to me that no provision has been made for disbursements of currency.

(B.) If you create a new data file (a file on the diskette with a distinct filename where the program stores your transactions) for each month, the List By Account option will collect and display the transactions from each month and total them for you under the account to which you distributed them when you entered them.

(C.) VERSALEDGER II offers you no protection against the possible posting of a Check Register more than once. Transactions are posted immediately upon being entered in the Check Register portion of the package, and the posting affects the Financial Statements every time. As far as I am concerned, this is bad news. I think that users are going to have problems. Obviously the author does not agree or he would have done something about this potentially serious flaw.

Check-Writing

The Check-Writing section of the system has a provision for writing your checks. Earlier I told you the system used check stubs as source documents; then I added parenthetically "almost true." It was this Check-Writing feature I was referring to.

After reading the instructions in the manual concerning how this is accomplished, I tried to relate the procedure to a situation where I thought it woul work reasonably well and wasn't able to do so.

Blank check forms must be loaded in the printer when you are operating under the option Add To Checkbook. You proceed to make entries just as though you were entering the information from the stub of a previously hand-written check. Next a question comes on the screen: "WOULD YOU LIKE THIS ENTRY PRINTED (Y/N)?". If you answer "Y", a check will be printed, and you proceed to enter the next transaction.

This procedure is one check at a time, and I am not impressed with it.

So much for Cash Basis Accounting. If you have need for Accrual Accounting, then you will have to use the second half of the system, the General Ledger.

Cash Vs. Accrual Accounting: The General Ledger

The General Ledger section contains a menu option for making journal entries. In the days when I took College Accounting 101, they were called General Journal entries.

You know that if you sell goods on account and if you buy goods on account, then you must have a method to record what customers owe you and what you owe to your suppliers. By using the option that allows you to make General Journal entries, you are able to make all of these necessary entries. This is called Accrual Accounting, meaning an accounting system which deals with information having nothing to do with a cash-type entry.

VERSALEDGER II contains an option that allows you to make a one-sided entry. I think this is unfortunate. But maybe this is like beauty — it lies in the eyes of the beholder.

Before I end this discussion of Accrual Accounting, you should be aware of H & E Computronics' series of modules which are able to link to VERSALEDGER II. I have not seen these packages. It is my understanding, however, that they are intended to serve the need a business might have for Accrual-type Accounting. The collection of five (5) modules is called THE VERSABUSINESS SERIES. Here is a listing taken from one of their ads:

1. VersaReceivables—$99.95
2. VersaPayables—$99.95
3. VersaPayroll—$99.95
4. VersaInventory—$99.95
5. VersaLedger II—$149.95

The Matter Of Timings

My experience has shown me that most business people (those past 35 or so who own and/or operate small businesses) think of computers as a TV screen on their desk hooked up to something, somewhere designed to flash (and I mean fast) any kind of information about their business they would like to have, any time they would like to have it.

These people give no thought to how such information gets to them in the first place. No one has told them the word "flash" has a very close relationship to the word "cash;" the more flash you get, the more cash you are going to have to spend to get it.

I find myself always wanting:

1. The screen to change faster.
2. To move from one option to the other faster.
3. To call up a record faster.
4. To edit information at the end of a record faster.
5. To enter information faster.
6. To print information faster.
7. The list goes on with each line ending "faster."

I have learned, however, to evaluate a software package by giving a lot more attention to the reports a package will generate, and a lot less attention to how fast things "flash" on the screen.

Here are some times I observed while operating VERSALEDGER II:

1. Changing from menu to menu — about 30 seconds
2. Load sort program — about 15 seconds
3. Review checkbook (20 items) — about 2 minutes
4. Print income statement—about 9 minutes
5. Print balance sheet — about 4 minutes
6. List by account (about 120 accounts of which 23 had activity over a period of three months) — about 1 hour

In my opinion, there is nothing outstanding about the response times of VERSALEDGER II, and realistically, there cannot be at this price. It is a whole lot better than doing it manually, and the package is equal to others I have seen costing much more.

Bugs In The Program And Subsequent Vendor Support

I found a couple of bugs in the source code. These errors caused a break in the execution of the program and an error message came on the screen. The problem was with a few CLEAR statements that

were omitted at the end of a few programs. I was able to find and correct the problem.

Then I called H & E Computronics to ask them about this problem. The first person I talked to was not able to help me (it was a Saturday), but when I told him who I was and what I was doing, he said he would try to find the right person and have him return my call.

The person who returned my call was Richard Kaplan, and I know he is listed in the manual as the author of this program.

Mr. Kaplan told me he was aware of the problem and how to fix it. I discussed with him the technical support offered by H & E Computronics for the program, and he told me it would be supported from 9:00 a.m. to 5:00 p.m. (Eastern time) on week days. I expressed my appreciation to him for getting back to me so fast; he assured me he did not return my call quickly only because I was reviewing the program.

I liked talking to Richard Kaplan. I'm guessing he just may be a "Bob-Snapp type of guy," and in my book, this is a compliment of the highest order. (For the uninformed, Bob Snapp is widely known for trying to help people.)

It would be unfair to labor the point about a couple of bugs I found in the program. However, it would be just as unfair not to tell you how totally baffled I would have been by these bugs if I had just started microcomputing. In my opinion, this is the most frightening thing to have happen to you when you are a babe in the woods.

Notwithstanding what any purveyor of software or hardware will tell you, it is your right to have adequate support while learning about your new purchases. There is a learning curve to contend with, and it is much longer than that required to learn to drive a car.

If you haven't already had it happen to you, you are going to find that sales people do not stay around forever. One of the questions you should ask every person from whom you are purchasing software is how long have they worked at what they are doing and how long has the business been around. You will be surprised how many answers of "less than six months" you will receive.
I think that you will be supported of you decide to purchase VERSALEDGER II.

Closing Remarks
I have tried to combine a review of VERSALEDGER II with a limited dissertation of what I think users of microcomputers in small businesses need to know to make intelligent decisions regarding software purchases.

I function as a systems consultant in the world of microcomputers. You can be helpful to me, my clients, and other small business people by sharing your views and comments regarding VERSALEDGER II and the statements in this article. You may contact me at the address listed at the beginning of this review.

VERSALEDGER II is available from:
H & E Computronics
50 N. Pascack Road
Spring Valley, NY 10977
Cost: $149.956
Call: 800/431-2818
(except N.Y. residents)
In N.Y. call: 914/425-1535

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

NEW PRODUCT ANNOUNCEMENT

Jere F. Stahl, President of Stahl Headers, has announced the availability of Stahl Business programs, touted to be "the most complete business management software package available for a small manufacturing operation."

The Stahl Business Programs feature:

* Complete manufacturing inventory with Bills of Materials
* Bill of Labor Operations
* Sales Bill of Materials
* Complete cost for every job card
* Two cost levels
* Three selling price levels
* Three hourly rate figures
* Two material markup percentages
* Complete sales history
* Order Entry (maintains back orders)
* Mailing Lists
* Customer Lists
* Invoicing
* Accounts Payable
* Sales Analysis
* UPS Calculations
* Pricing Forecasts

In addition, General Ledger and Accounts Receivable will be available in August, 1983.

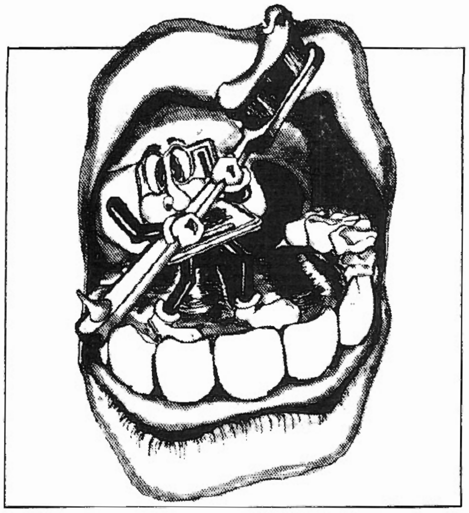
The program currently runs on the Model II/12/16 in Z80 mode; a CP/M version is "in the works."

For $25, you can get a copy of the 157 page Manual to examine. If you buy the program, you'll receive a full credit, and if you return the manual in good condition within 30 days, you'll be refunded $15.

This package can be puchased for a limited time for $3,495.00.

Want to know more? Contact:
Mr. Jere Stahl
Stahl Headers, Inc.
1515 Mt. Rose Avenue
York, PA 17403
717/846-1632

SPECIAL USER REPORT



COMPUTERIZED DENTISTRY IS PAINLESS

Dr. Donald Gutkin
1368 McPhillips Street
Winnipeg, Manitoba
Canada R2X 2M4

Attention practicing dentists: have you ever longed for an easily implemented and relatively inexpensive way to computerize your accounts receivable, your billings each month, your insurance forms, and most importantly, the record-keeping methods in your offices? Of course you have! Well, as a practicing dentist myself, let me tell you such a system exists, and it is the incredible and painless Dental Office Management System from MICRO/SYS80.

This system operates on the Radio Shack Model II, 12 or 16, any of the TRSDOS operating systems, i.e., floppy control or hard disk, and is written in the RSCOBOL language. The COBOL program is delivered to the end user as a machine language program that I have found will run ten to thirty times faster than any similar programs written in the BASIC language.

What This System Did For Us

We incorporated the Dental Office Management System with ease and confidence just under a year ago in our small dental health clinic consisting of a multiple dentist/hygienist environment. The computer keeps track of each dentist/hygienist on an individual basis and also keeps track of the individual procedures produced by them.

The system allowed us to easily define our particular office needs. For example, we required an office size of ten to twelve thousand patients and the ability to handle several hundred dental procedure codes and fees, several dentist/hygienists, and over ten thousand transactions a month. This requirement was handled easily. Each dentist is assigned a file for the maintenance of ID numbers, addresses, telephone numbers, social security/insurance numbers, etc. These are created for their inclusion on third party insurance forms and account reports.

The system is designed to accommodate both Canadian and American needs, by having the ability to produce ADA insurance forms and CDA insurance forms. The programmer from MICRO/SYS80 will also custom design any claims format the user wishes. Patient files are also easily created to include addresses and accounting information necessary for completion of monthly statements, records, and production of third party insurance forms as needed. The program also accommodates all other differences required of Canadian and/or American needs.

The outside limits of the system, in terms of the number of accounts and patients, the number of procedures available for recall, the number of transactions available each month, the number of dentists/hygienists that can be accommodated, and the number of insurance companies that can be kept on file is limited only by the size of the storage media utilized. The lowest numbers are for use on four floppy drives and will hold 2000 accounts and 4000 patients. The maximum number of entries is approximately 15,000 to 50,000 patients. This would then require one to four hard disk drives.

I found that the hard disk is really the only way to fly. Floppy eight inch drives can be used, but access time and back-up is greatly slowed down. Far too much wear and tear on the floppies occurs when they are utilized continuously for eight to ten hours a day. We presently use the 8.4 meg. Radio Shack hard disk, and even though it is a fairly expensive Winchester disk, I recommend it highly. It greatly enhances the speed and reliability of the system, and the hard disk provides a tenfold increase in functional speed, as well as fast back-ups to tape or diskette. (A tape back-up by non-Radio Shack distributors is now available; however, this system is very expensive.) We use floppies for our back-up and have had little trouble. We are also able to back-up the hard disk and take the material off the premises which gives us an edge over fires or vandalism. We can easily recreate our office at any time with the back-up.

Since implementing the Dental Office Management System, we have seen our efficiency and ease of operation greatly increased. Patients are located in the system by name or by account name or number. Since the charges for each patient are so simple to enter using the System, any non-experienced staff member can easily be trained in a short period of time to enter data. Entering data becomes an easy task because patient records are charged using the procedure codes established by our office, along with our office's standard and conventional fee guides.

Important note: All of these standard charges can be overridden. Payments can also be applied at this point, recalls can be changed or made, appointment dates set, and an insurance form or receipt can be automatically issued.

At any time, any of the information in any of the files can be added to, changed, deleted, searched, displayed, and listed. This feature provides the staff with the means of correcting mistakes without a major production occurring.

Management Reports

How about management reports? Here is a way to reduce accounting costs and improve office efficiency at the same time: Sound impossible? Not any more. For all files, listings can be made in alphabetic or numeric order, on the screen or printer, or both. You can define in each report what data you wish to display. In other words, the whole report or parts of it can be displayed or printed at any time. We can produce any kind of mailing labels, i.e. recall labels or lists, appointment labels or lists, insurance company labels or lists, and account or patient labels or lists.

Daily, this system generates a reconciliation of work and payments produced, charge slips, and next day appointment lists. We have the option to print out our insurance forms by the day or for any time period we wish to designate.

The monthly reports are unbelievable! We can generate the billing reports, receivable reports, production reports, and collection reports for each dentist, each hygienist/assistant, and/or the whole office. Statements are so easily produced it is hard to imagine. What used to take our office two to three weeks to accomplish now takes two to three hours without any personnel involved or office time being utilized — we process statements overnight while the office is closed! We can initiate the statements to print for any time designation; i.e. we can send all the statements, or only those outstanding for over 30, 60 or 90 days. The amount of mailings is effectively reduced, and those accounts needing special attention are picked out. The monthly reports keep track of all the doctors individually, as well as the office totals which are broken into current amounts, month-to-date, and year-to-date amounts, including running totals.

Billing Reports

The billing report generates lists of accounts that are delinquent 30, 60 or 90 days or more, and provides us with the means to tag our statements with dunning messages. The statement routine automatically assigns the correct notation for each type of outstanding balance. Can any manual accounting system do this for you? If inquiries about accounts are phoned in, then we can access the system files through the keyboard, or if the system is tied up, we have a report, called the "Account/Patient Report," that lists all individual procedures for patients within the account designated. At the month end, the computer purges accounts, ages them, and deletes completed transactions. Again, this was never accomplished before and is now accomplished at night when the office is closed. All purged account information is stored in a handy history file for future recall, with the archiving time period (up to thirteen months) set by the dentist.

The Best Feature

The biggest plus for the system is its ease of implementation. Our experience was probably typical: with the operator's manual (included with the system) and the demonstration model, our receptionists and the other staff members learned to operate our dental office computer within two weeks — without outside help! Having a working demonstration model allowed us to create a mock, miniature office situation prior to actually implementing our real live patients and their records.

Letting go of the old manual system is always the most difficult thing to do, so we ran the computer and the manual systems for about 60 days to both compare the two systems for accuracy, but also to boost our confidence. Nobody believes that a computer is right. Well, we found out differently. The girls no longer have tedious accounting chores to do day in and day out. They also "painlessly" learned to accept the system and drop their old-fashioned ways.

The Bad Features

One of the pitfalls we found in the system is its ability to merge the office management system information with a word processor. It didn't take too long to change that, however. The MICRO/SYS80 programmer is now creating the word processing function.

Another feature we found annoying is the limitation on how much information can be entered at one keyboard and terminal at one time. To remedy this situation, we intend to incorporate a multi-terminal/task system as soon as the Radio Shack people develop the multi-terminal software for the Model II, 12 or 16. (By the way, to my understanding, the Model II will be able to have this capacity if the necessary enhancement boards are added to the system. The update boards are presently available, but the operating system is not.)

The Right System For You

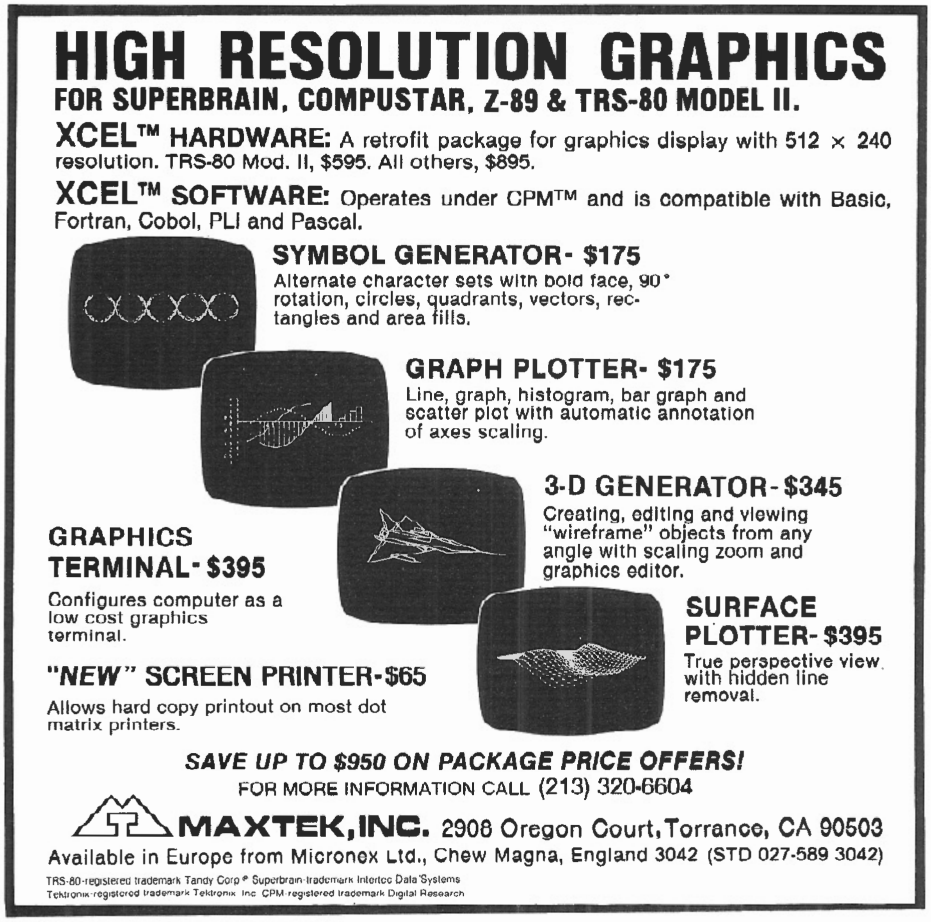
Many office management systems are available to the dental profession. As a matter of fact, hundreds of different systems are out there that range in cost from a few hundred dollars to tens of thousand of dollars. The hardware range is equally as large. The Dental Office Management System from MICRO/SYS80 is a system that can be purchased for a reasonable price, and it has the essential ability to grow as office needs grow.

The cooperation, back-up, and support we received and continue to receive from the programmer and distributor has been excellent. Someone is always available to lend a helping hand, and there are no front desk receptionists to deal with.

I strongly recommend this system for any progressive dental office interested in "painlessly" converting to a computerized office management system.

Ed. Note: According to Mr. Frank Weiss of MICRO/SYS80, the System costs $2000.00 for U.S. dentists and $2800 for Canadians. The System is an off-the-shelf program that will be customized to the purchaser's particular needs. Mr. Weiss told two/sixteen: "Softwear doesn't wear out," so it comes with a life-time guarantee and no maintenance charges.

For more information concerning the Dental Office Management System discussed in this review, contact:
Mr. Frank Weiss
MICROSYS/80
236 Waverly Road
Southampton, PA 18966
215/355-5706



why won't

XKILL

work?

Keith F. Weatherhead
RR#2, Box 97A
Wilmington, IL 60481

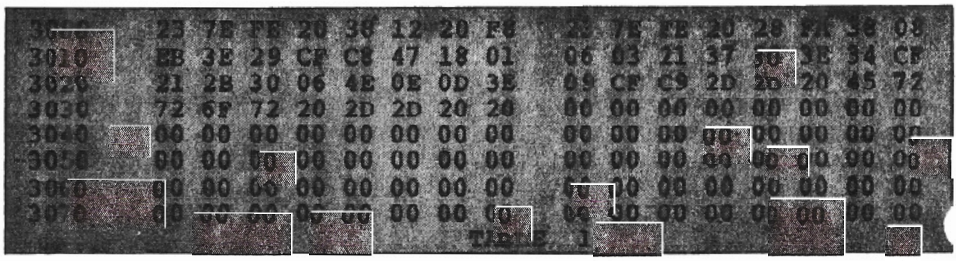
When I received my September-December issue of *two/sixteen*, I read it cover to cover, as usual. When I came to the program called KILLABS by Roger Conant ("Kill the Kill Prompt with Killabs," Vol. 1, No. 3, p. 47) I decided to enter it as it dawned on me just how annoying the TRSDOS KILL command prompt is.

Upon entering the code in Table 1 (this is the program code from the original article), I was now ready to save (DUMP) my new program to disk. There was only one small problem; I already had a program named KILLABS. Since this program did not have anything in common with Mr. Conant's program and all my other modified TRSDOS commands or utilities have a prefix of "X", I decided to call this program XKILL.

I DUMPed the program; then typed

XKILL <enter>

My system loaded the program and promptly returned the error message: "File not found". I thought, well of course, I did not enter a filename. Even though the error message was not what I would have preferred to see for that particular situation, it did advise me that a legitimate error had occurred. So I created a file called TEST and verified its presence. I then keyed in XKILL TEST <enter> and again I received "File not found." Now how could this be? I had just created the file and verified it was there. I tried a few more times, but still it did not work. I re-checked the hex listing from Table 1 against what was in memory, and everything matched. At this point I knew that looking at a hex listing was not going to solve my problem.



```
3000   23 7E FE 20 38 12 20 F8   23 7E FE 20 28 FA 38 08
3010   EB 3E 29 CF C8 47 18 01   06 03 21 37 30 3E 34 CF
3020   21 2B 30 06 4E 0E 0D 3E   09 CF C9 2D 2D 20 45 72
3030   72 6F 72 20 2D 2D 20 20   00 00 00 00 00 00 00 00
3040   00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
3050   00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
3060   00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
3070   00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
```

TABLE 1

Using my disassembler, I dismantled the program. After looking at the source code for a few minutes, the problem became apparent to me. Mr. Conant had made an assumption while creating his program: he assumed that the program name would always be seven characters in length.

To understand the problem, let's take a closer look at the first six bytes of the program. They are:

01 09 00 09 54 5D

When these six bytes are disassembled, they become:

```
00100  01 09 00      LD    BC,0009H   ;    LOAD 'BC' WITH '9'
00110  09            ADD   HL,BC      ;    ADD & STORE IN 'HL'
00120  54            LD    D,H        ;    LOAD 'D' FROM 'H'
00130  5D            LD    E,L        ;    LOAD 'E' FROM 'L'
```

All looks normal, except lines 120 and 130 could be done with one command (EX DE,HL), so what is the problem?

To further understand the situation, we must refer to the TRSDOS Owner's Manual, Technical Information Section, "Programming with TRSDOS." Under the paragraph titled "Program Entry Conditions", it explains that when you execute a program, TRSDOS loads the program into RAM and then sets up the three major register pairs BC, DE, and HL, before turning control over to your program. In this case, the BC and DE register pairs are of no interest to us. However, HL as explained, is set to the address of the first character of the last command entered, minus one. When you hit the <enter> key, TRSDOS subtracts the address of the first character entered from the address of the carriage return, thus returning the length of the command entered. That value is then stored in the byte preceeding the command which enables the program to determine how many characters have to be analyzed for that command. This value includes the program name, leading spaces, and parameters, if any, but NOT the carriage return.

Where is this leading? You remember the disassembled source code above? Let's manually execute it to see what happens. Given the command

KILLABS TEST <enter>

the address put into HL prior to program execution is the address of the byte preceeding the "K" in KILLABS.

At this point, we are ready to reinact the program. We load the BC register pair with 9 and add it to the address in HL as supplied by TRSDOS (instruction #2, in the above sequence). Now we load the value in H and L into D and E (instructions #3 and #4) and now have the starting address of our filename. If you count the nine characters (don't forget you are starting from the length byte!), you should be at the first "T" in TEST.

Let's change something external to the current program code. Remember, I called the program XKILL. Let's work through the same code we just went through only with the command buffer containing

XKILL TEST <enter>

If you followed the above instruction sequence correctly you should be looking at the "S" in TEST. That means my program thought the file-name was ST, and that is why I received the error message "File not found".

I saw a potential disaster pending. Consider the following: you have two files on your PAYROLL data disk. Your main data file is PAYDAT, which contains your 1983 PAYROLL data. Since you wanted to retain your 1982 data you renamed that file EXPAYDAT. Now that you have all your year-end work done and your 1982 file is on an archive disk, you decide to delete it from your normal working disk. You enter the command

XKILL EXPAYDAT <enter>

and go about your business. The next week you go to run your PAYROLL program and receive an error message "File not found". Time to have a massive heart-attack! You do have a current back-up, don't you? (Remember hearing people talk about a man named MURPHY? I wonder if it was at times like this).

Ok now, let's take a survey: how many of you out in magazine-land entered the KILLABS program, changed the name, and the new name is NOT seven characters in length? You have the right idea: go check your diskettes to see what you are missing.

If you have been patient enough to go through this pre-lesson explanation and do not have any interest in learning the programming solution, I will give you a working solution in the fewest steps possible.

1. Boot your system—Load TRSDOS
2. Key in: CLEAR <enter>
3. Key in: LOAD prog <enter> (prog = your program's name)
4. Key in: DEBUG ON <enter>
5. Key in: DEBUG <enter>

NOTE: If you used Mr. Conant's relocation address, then you should be looking at the code. If you did not use a relocation address of 2800H then enter M and give DEBUG the address you used.

6. Depress the <F1> key
7. Move the cursor to the second byte location (currently an 09) and change it to the length of your program name plus two (+2).

NOTE: REMEMBER you are working with hex, and if the length of the program name was 8, the new hex value will be 0A — not 10!

8. Depress the <F2> key
9. Key in: O (That's the letter O — NOT ZERO)

10. At the TRSDOS READY prompt, reenter the dump command and save your updated program back to the diskette.

NOTE: On the DUMP command, you do not need to specify a relocation or transfer address. Simply enter 2800H (or wherever your program starts) for the STA value and the END value will be 7FH above your starting address.

Your program will now function as it should have if you had named it KILLABS.

Classified Advertising

A non-dealer subscriber may place one ad per issue at the rate of $7 per column inch.

A dealer may place one classified ad per issue at the rate of $20 per column inch. Dealer classifieds must be for the sale of specific products or services at specified prices.

Ads must be for hardware, software, or services which would be of interest to Model II or 16 users (no mystical societies chain letters, book clubs, etc.).

Ads will be set in the same size type as the text in the magazine.

Closing date for classified ads is the 15th day of the month prior to issue date (e.g., the January-February classifieds will close December 15.

We reserve the right to determine what heading an ad should be placed under (e.g., HARDWARE, SOFTWARE, MODEMS, PROGRAMMING), but we will accept advertisers' reasonable requests for specific headings.

The Proper Solution

Now that half of our audience is in the process of correcting their programs, I will go over a coding procedure that will prevent this from happening. If you intend to write your own programs, whether you sell, publish, or just keep them for your own use, the following principles should be applied to allow flexability and assure error free operation.

After we go through the code in Figure 1, be sure you understand the concepts before you try to use them. You may also need your TRSDOS Owner's Manual for this discussion. An assembler reference may also be helpful.

As we learned earlier, when a program is loaded by TRSDOS the HL register pair is set to the address of the byte containing the length of the TRSDOS command or Program name and parameter(s) entered. Since the command length value will be low enough, (unless there are many parameters or they are extremely long [32 characters in length]), we must immediately skip the length byte before we do any testing. You can, if desired, store it in a register or memory cell if your program will need it. If you fail to skip this byte before doing the test in Line 1130, there is a great chance you will take the error exit.

Since we have advanced the buffer pointer, it should now be pointing at the first character of the program name. This is where you may need your TRSDOS Owner's Manual (Character Code Chart in the Appendix). We compare the value of a space (blank-20H) with the character in the command buffer. If the value is less than the value of a space, the Carry Flag in the Z80 processor will be set. When the instruction in Line 1150 is encountered, the program will jump (branch) to the code at the label CALERR.

If the character is greater in value than a blank, the flags will be reset to show a no-carry condition. The instruction in Line 1160 will test to see if a match occurred. Since it would not have compared, the program will transfer control back to the code at the label XKILL.

This process will be repeated until one of two situations occurs: first, if a value of 1F hex or less is encountered, the jump to CALERR will be taken. Second (and normal progression), the conditional jump in Line 1160 will not occur if a space was encountered. The program will now fall through to the instruction at Line 1170. Line 1170 through Line 1210 does the same type of testing. However, one of the terminating tests has been changed. Notice Line 1200 says to jump on compare (Z) rather than no-compare (NZ). This change allows the program to skip over any leading spaces prior to the first significant character of the first parameter.

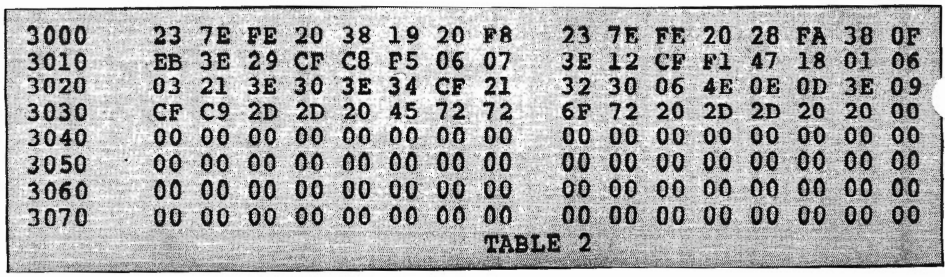
At this point we will not return to the label FNDPRM, and there will not be a CARRY, so we will simply fall through the code to the exchange register instruction in Line 1260. Depending on the program application, you may have to do other testing. In our case, if the supplied parameter (filename) contains any illegal characters or syntax, then TRSDOS will flag it and return to us the appropiate error status. Any other testing in this case is not necessary.

The first ten lines should always be used (in some form) whenever you write a program that accepts parameters on the call. If you remember my problem, and understand the function of the lines detailed, then you can see that had the code been developed in this way it would not have mattered what I named the program, as well as whether I typed extra leading spaces, or only one space before the filename. The rest of the documented source code should be self-explanatory and not present any problems.

It has been my experience though many years of writing programs that documenting the source code while creating it can save hours of grief and frustration when trying to implement changes. Other than the front-end code and a few cosmetic changes, plus the documenting of the source code, the basic program is the same in appearance and operation as the orignial program submitted by Mr. Conant.

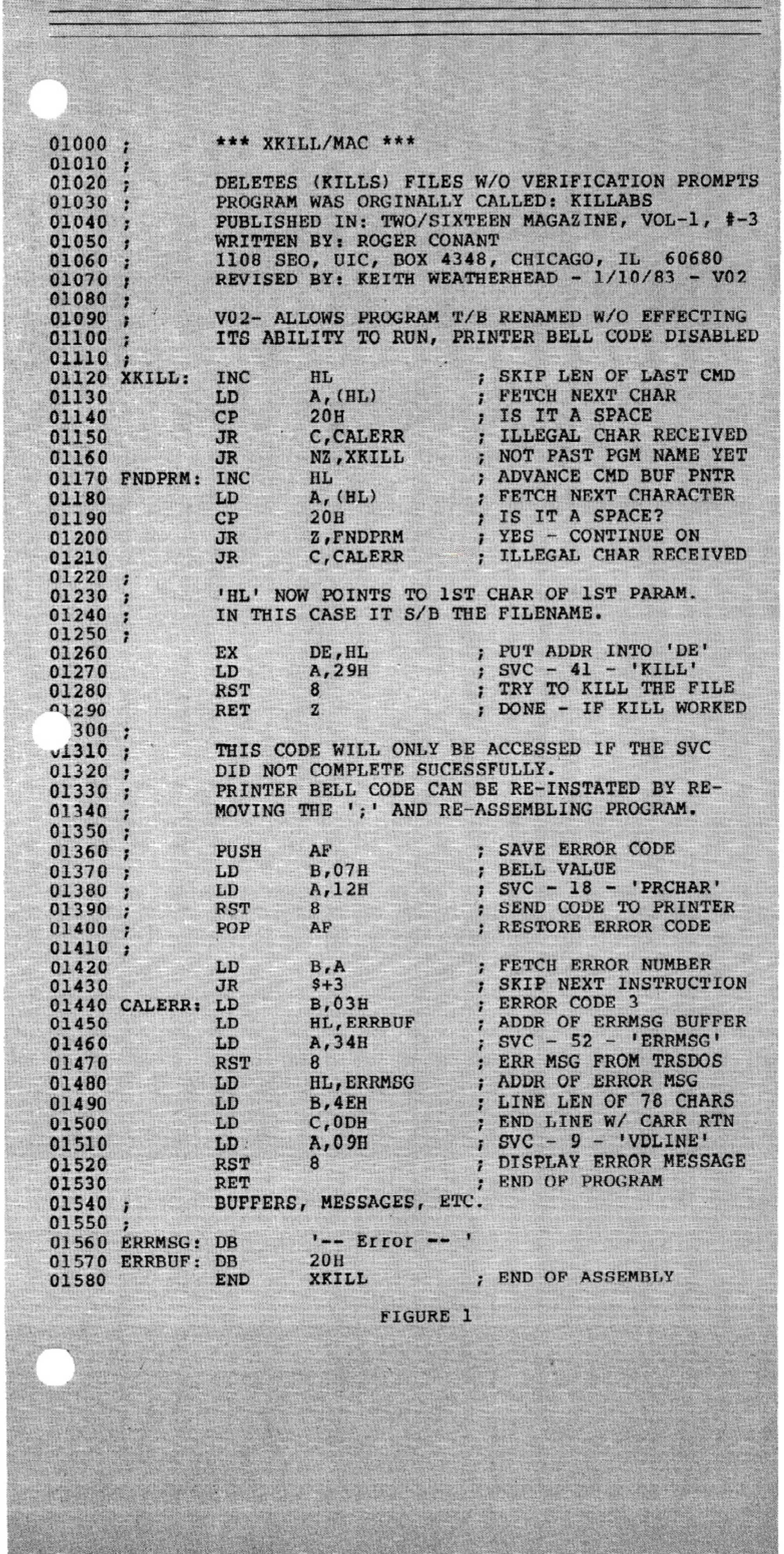
Despite my nitt-picking, coding preferences and expectaitons, the program originally submitted is useful and definitely appreciated.

For those who do not have an EDITOR/ASSEMBLER, the code in Table 1 is the revised program with the printer bell code disabled. The code in Table 2 is with the printer bell code activated.



```
3000    23 7E FE 20 38 19 20 F8    23 7E FE 20 28 FA 38 0F
3010    EB 3E 29 CF C8 F5 06 07    3E 12 CF F1 47 18 01 06
3020    03 21 3E 30 3E 34 CF 21    32 30 06 4E 0E 0D 3E 09
3030    CF C9 2D 2D 20 45 72 72    6F 72 20 2D 2D 20 20 00
3040    00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00
3050    00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00
3060    00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00
3070    00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00
```

TABLE 2



```
;               *** XKILL/MAC ***
;
;               DELETES (KILLS) FILES W/O VERIFICATION PROMPTS
;               PROGRAM WAS ORGINALLY CALLED: KILLABS
;               PUBLISHED IN: TWO/SIXTEEN MAGAZINE, VOL-1, #-3
;               WRITTEN BY: ROGER CONANT
;               1108 SEO, UIC, BOX 4348, CHICAGO, IL  60680
;               REVISED BY: KEITH WEATHERHEAD - 1/10/83 - V02
;
;               V02- ALLOWS PROGRAM T/B RENAMED W/O EFFECTING
;               ITS ABILITY TO RUN, PRINTER BELL CODE DISABLED
;
XKILL:  INC     HL              ; SKIP LEN OF LAST CMD
        LD      A,(HL)          ; FETCH NEXT CHAR
        CP      20H             ; IS IT A SPACE
        JR      C,CALERR        ; ILLEGAL CHAR RECEIVED
        JR      NZ,XKILL        ; NOT PAST PGM NAME YET
FNDPRM: INC     HL              ; ADVANCE CMD BUF PNTR
        LD      A,(HL)          ; FETCH NEXT CHARACTER
        CP      20H             ; IS IT A SPACE?
        JR      Z,FNDPRM        ; YES - CONTINUE ON
        JR      C,CALERR        ; ILLEGAL CHAR RECEIVED
;
;               'HL' NOW POINTS TO 1ST CHAR OF 1ST PARAM.
;               IN THIS CASE IT S/B THE FILENAME.
;
        EX      DE,HL           ; PUT ADDR INTO 'DE'
        LD      A,29H           ; SVC - 41 - 'KILL'
        RST     8               ; TRY TO KILL THE FILE
        RET     Z               ; DONE - IF KILL WORKED
;
;               THIS CODE WILL ONLY BE ACCESSED IF THE SVC
;               DID NOT COMPLETE SUCESSFULLY.
;               PRINTER BELL CODE CAN BE RE-INSTATED BY RE-
;               MOVING THE ';' AND RE-ASSEMBLING PROGRAM.
;
;       PUSH    AF              ; SAVE ERROR CODE
;       LD      B,07H           ; BELL VALUE
;       LD      A,12H           ; SVC - 18 - 'PRCHAR'
;       RST     8               ; SEND CODE TO PRINTER
;       POP     AF              ; RESTORE ERROR CODE
;
        LD      B,A             ; FETCH ERROR NUMBER
        JR      $+3             ; SKIP NEXT INSTRUCTION
CALERR: LD      B,03H           ; ERROR CODE 3
        LD      HL,ERRBUF       ; ADDR OF ERRMSG BUFFER
        LD      A,34H           ; SVC - 52 - 'ERRMSG'
        RST     8               ; ERR MSG FROM TRSDOS
        LD      HL,ERRMSG       ; ADDR OF ERROR MSG
        LD      B,4EH           ; LINE LEN OF 78 CHARS
        LD      C,0DH           ; END LINE W/ CARR RTN
        LD      A,09H           ; SVC - 9 - 'VDLINE'
        RST     8               ; DISPLAY ERROR MESSAGE
        RET                     ; END OF PROGRAM
;               BUFFERS, MESSAGES, ETC.
;
ERRMSG: DB      '-- Error -- '
ERRBUF: DB      20H
        END     XKILL           ; END OF ASSEMBLY
```

FIGURE 1

HAVE YOU DONE ANY PATCHING LATELY?

Paul Naitoh, Ph.D.
Applied Psychophysiology Group
4446 Marseilles St.
San Diego, CA 92107

You may not realize it, but even as a complete novice, you have had lots of experience with PATCHing. You PATCH up a hole in the wall or a difference in opinion threatening your family relations. You might even PATCH the seat of your pants if it has a rip. However, you are willing to PATCH up a computer program to make it work properly, especially when you are not familiar or comfortable with a computer?

When you consider that PATCHing involves permanently changing the structure of machine-language program files, such as SYSTEM, BASIC, SCRIPSIT, and other programs, the task can seem frightening. Also, the fact that you don't get friendly instructions about how to actually do the PATCHing does not help. For example, many interesting PATCHes written by Bob Snapp appeared in *two/sixteen magazine* (Vol. 1 #2, pp. 26-30). Alas, these useful patches are not given in a "user-friendly" manner.

Another example of PATCHES are those you see in Radio Shack's *TRS-80 Microcomputer News*. Some of these PATCHES must be made to correct errors in programs, or to make existing programs run more efficiently, or to up-grade programs. In these situations, you don't have the option to ignore them as you did with Bob Snapp's DOSFIXES.

The purpose of this article is to show you how to PATCH programs correctly in a step-by-step fashion. Obviously, I am not addressing those of you who are way ahead in the game of computer programming. My intention is to reduce for you a still-remembered sense of insecurity I felt towards PATCHing.

To begin PATCHing, you do NOT simply start typing, for example,

```
PATCH SCRIPTSIT A=DDF6 F='N' C='Y' <ENTER>.
```

Yes, this procedure will work. But if you do this, you do not have an easily accessible permanent record to refer to later that tells you whether you have completed a particular patch or not. Yes, you can jot down on a piece of paper the patches you have installed, but you know that sooner or later this paper will be lost. Then what are you going to do?

With a TRSDOS Library Command called BUILD,' you can create a permanent record in a disk file of all patches you have made under a name you choose. In our example, we will use the name FIXSCRPT to indicate a "fix" tq "SCRIPSIT." Whenever you wish to recall what sort of PATCHES are already made, you use a BUILD file (i.e., a file created by BUILD).

So, the very first thing you must remember when PATCHing is start with the TRSDOS Library Command BUILD.

At TRSDOS READY, type in

```
BUILD FIXSCRPT <ENTER>
```

TRSDOS responds to you by displaying: "Enter Command Line (1-80)" with 1-80 being the number of characters you can store on one program line.

Now you type

```
PAUSE
```

and follow "PAUSE" by the comments you wish to make on the PATCH you are going to install. For example, your comment on a SCRIPSIT 2.0 PATCH "to over-strike '0' by '/' to create 'Ø'" can be:

PAUSE OVER-STRIKE "0" BY "/" TO CREATE "Ø"; SEE SCRIPSIT REFERENCE MANUAL PAGE 73 <ENTER> *[78 characters — meets criteria for one program line]*

Then, TRSDOS shows your PAUSE message in a reverse field, and displays:

"Store line (CR/ESC)." If you wish to store the comment you just typed, press <ENTER>; if you wish to start over again, press <ESC>.

If you press <ENTER>, then TRSDOS informs you that your comment is recorded under the file name FIXSCRPT by displaying: "***Line Stored in File***".

Then TRSDOS displays: "Enter Command Line (1-80)", giving you the opportunity to add another line of comments if you require more space to complete what you wish to say. If you want to add more, type:

```
PAUSE (YOUR COMMENTS) — <ENTER>
```

When you are through with your comments, finally you are ready to type in a PATCH. For example, in the case of a SCRIPSIT PATCH to strike through 0 by /, you type in:

```
PATCH SCRIPSIT A=DDF6 F='N' C='Y' <ENTER>
```

If you have more than one PATCH, then repeat the above process of a comment followed by a PATCH.

Once you have completed typing in the PATCH, you can store all of these comments and PATCHES by pressing <ENTER> in response to the TRSDOS display: "Enter Command Line."

A final message from TRSDOS tells you that editing is now completed, and it flashes back a message: "TRSDOS READY."

You can see the results of your handiwork in completing a BUILD file, named FIXSCRPT, by examining your directory. Do you wish to see the content of this BUILD file? You can display the file by typing at TRSDOS READY

```
LIST FIXSCRPT <ENTER>
```

Or to print the contents of the file, type:

```
LIST FIXSCRPT {PRT}
```

What you have achieved so far is to BUILD a DO file which contains both the actual PATCH and your comment about it. But you have not yet PATCHed the PATCH into a machine-language program.

To do the actual PATCHing, key in the following:

```
DO FIXSCRPT <ENTER>
```

A note in case you've gotten confused: the file you have created using the TRSDOS Library Command "BUILD" is known as a "DO File." The commands you have entered into the BUILD file will be executed when you tell the system: "DO {filename}". The same results could be accomplished at TRSDOS READY, but then you would not have your permanent 'ecord on disk.

When you start a PATCH, TRSDOS first displays your messages line by line, and it prompts you to "Press any key to continue." Next TRSDOS executes the PATCH and then tells you either the PATCH is completed or "String Not Found—Abort."

Do NOT panic if you get the abort message. Nothing is done to your program when PATCHing is aborted. Simply, TRSDOS cannot find the machine-coded information at the location you have specified. This means one of three possibilities: (1) the program has been already PATCHED by you a long time ago; (2) the program has been altered, or (3) you did not correclty type in the informaiton needed for the PATCH.

You can also add to an existing BUILD (or DO) file. All you have to do is type in at TRSDOS READY

```
BUILD FIXSCRPT <ENTER>
```

and then proceed as outlined above.

Good luck, and happy PATCHing!

Review of the STAT MULTI-PACK

Richard Jensen
University of Illinois at Chicago
P. O. Box 4348
Chicago, IL 60680

Stat Multi-Pack is a very good package of advanced statistical routines that will appeal especially to researchers in psychology and education.

The package was reviewed in the Model II/16 version (on the Model 16 it operates under TRSDOS 2.0a). All the programs are written in interpreted BASIC, which tends to make them slow. Presumably a compiled version would be much faster. The calculations are handled in double precision.

Program Requirements

The package will work on a one-disk system; a printer is highly recommended since some results are routed only to the printer, not to the screen. An ample supply of paper is also called for, since allowance has been made for printing the maximum number of variables even if only a few are used.

What The Package Includes

The package includes three versions of ANOVA (including analysis of variance/covariance, and a within and between version), factor analysis, discriminant functions, and stepwise multiple regression. In addition, matrix manipulation and data transformation routines are included. All the procedures are menu driven, which makes for very easy use. The manual is clear enough on how to use the programs. While it does include brief, sophisticated discussions of technique, it is not a textbook. The user must already know statistics to use the package — two or three advanced undergraduate or graduate courses in statistics would be helpful.

Data Entry and Manipulation

In the all-important area of data entry and manipulation, the program has a relatively rigid structure that may not be amenable to the particular data a researcher has to work with. Each procedure creates files for 100 variables for each observation, regardless of the actual number of variables used. The 100 maximum will occasionally be restrictive in large projects. The data set can be stored either on the master disk or on any number of supplementary data disks. About 700 cases will fit on one disk. As a result, there is no maximum number of cases. Belanger's solution to the storage of data takes good advantage of the large capacity of the 8″ TRSDOS disks. By translating the programs to TRSDOS II, the user can store over 10,000 cases on a Tandy hard disk. However, there are no provisions for taking a data set created by another program and making it accessible to the powerful Stat Multi-Pak routines. Furthermore, the Stat Multi-Pak data files are not named. To save a file, one or more new disks must be dedicated to it. Anyone who creates a different data set every week or two will soon have quite a library of data disks and backups, most of them largely empty. Disks are cheap, to be sure, but a simple naming procedure would be most welcome.

Data entry is straightforward, with a nearly blank screen asking for the value for a particular variable for a particular case. Once the number is entered, it immediately vanishes. To check on the accuracy of data entry, a separate routine is needed; the "List Cases" routine will print out only four variables for each case. The "Matrix Manipulation" program, unfortunately, does not allow review or correction of data entries. The provision for missing values is inadequate, and no "select if" or "recode if" procedures exist.

Performance

The statistical procedures work well. The only bug I encountered involved the estimation of residuals in a multiple regression run. The menu driven system minimizes the problem of error trapping, yet a wide variety of options is preserved.

The strength of the package lies in advanced procedures that are rarely available for micros, and not at all for the Model II/12/16, especially factor analysis, discriminant function, and analysis of covariance. The latter allows up to five factors and 20 covariables. Compared to A-Stat (for the Apple II), Stat Multi-Pack is inferior in data manipulation, but significantly superior in terms of advanced procedures and greater capacity (100 variables versus 35 for A-Stat). Tandy's statistical analysis package for the Model II/12/16 permits far less data manipulation than Stat Multi-Pack, lacks most of the advanced procedures, is limited to 12 variables, and cannot deal with more than 100 or 200 cases.

Who's It For?

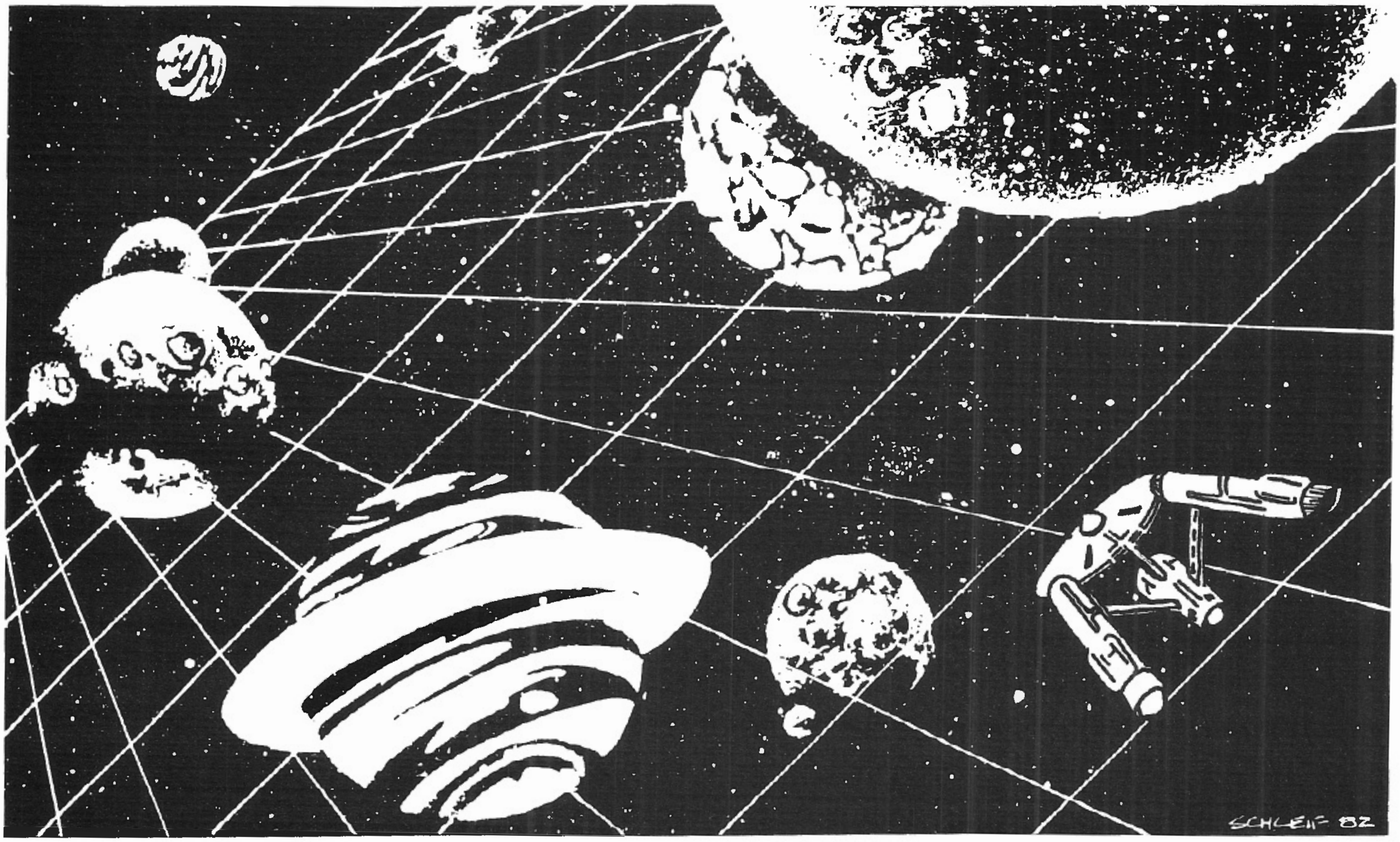
Stat Multi-Pack is an excellent choice for researchers in psychology, biology, and education. Sociologists, political scientists, and historians will also find it somewhat useful. They need a more useful missing value procedure, and would sorely miss the absence of a cross-tabulation procedure. They would also regret the absence of some statistics that the author should consider adding (such as standardized regression coefficients, probability values for F-tests, multiple R, adjusted multiple R2, and incremental R2, in the Multiple Regression module).

Economists and business researchers will not find the package very helpful because of the lack of time series procedures.

Stat Multi-Pack does not do everything. It does have advanced capabilities that are hard to come by this side of mainframe packages like SPSS, SAS, and BMD. Stat Multi-Pack is aimed at serious researchers who understand the sophisticated results they can obtain easily from this well-crafted package. It is the best package available for the Model II/12/16 running either TRSDOS or CP/M.

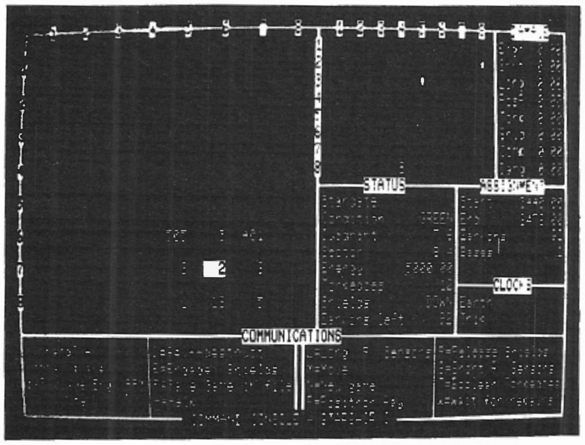
Stat Multi-Pack by Robert R. Belanger for
TRS-80 II/12/16
Price:$325.00
Available from the author
541 W. 6th Street
Azusa, CA 94702
213/969-4112

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.