Volume 4, Issue 5 **MAY, 1982**

Price $1.50

TRS-80® Microcomputer News

Information Published for TRS-80 users.

Pocket Computer

Color Computer Sorts

Johns Hopkins Results

Fort Worth Scene

It is getting difficult to remember what month it is. I just checked the calendar, which indicates that today is February 11, 1982. I am still waiting for the first printed copies of the February News (anything which could go wrong, and a few things we thought couldn't go wrong did for February), we are finalizing the March News, the April News is being typeset, we are writing the May News and planning the material for the June News. Very confusing. Any time someone asks a question about the "next" issue, we tend to stare blankly and ask the individual to be more specific.

This month's feature article is on some of the results of the Johns Hopkins Search finals. Obviously, we will be telling you some of the ways the top thirty entrants nationwide proposed using TRS-80 equipment to aid the Handicapped. We wish it were possible for us to bring you the complete story on each of these individuals and the methods they have devised. Some ideas are very straightforward, and most people would respond "Of course," while other ideas are very innovative.

One of the things I found most interesting about the results of the Johns Hopkins search is that many of the ideas are immediately workable, and several suggest easy modification to fit a particular need or handicap. While I find microcomputers to be invaluable in my day-to-day work, it is nice to know that they may also become an invaluable tool that will help individuals communicate effectively.

Also this month, we are giving the new TRS-80 Pocket Computer (PC-2) an introduction. The PC-2 should be arriving in some stores during May, and should be readily available toward the end of June. We hope that some of our enthusiasm for this new computer comes across in the article.

Z-80 ASSEMBLERS

We received a letter recently from Mr. Albert Landgrebe of Beltsville, Md. Mr. Landgrebe was concerned that there were a number of errors in the assembly language listing of Mr. Stern's Keep Order Sort (Dec., 1981). After comparing the listing sent by Mr. Landgrebe to the listing we published in December, I assume that the errors referred to involve addresses.

Those of you who are using Z-80 assembly language need to be aware that different assemblers handle addresses differently. The difference is that some assemblers, when they display the assembled code, display the two address bytes in LSB, MSB order as the Z-80 uses them, and other assemblers display the addresses MSB, LSB but output the assembled code to tape or disk with the bytes properly reversed.

How do you know what your assembler does? Look at a command with an address:

```
LD     BC, (7FØØH)
```

If your assembler gives you:

```
ED4B ØØ7F
```

your assembler prints LSB first as the Z-80 uses it. If your assembler gives you:

```
ED4B 7FØØ
```

your assembler prints the addresses MSB first. This can make comparing the assembled output to the source code easier, but when you are using the results of this type assembler, you need to exercise caution. For instance if you are moving the assembled Hex values into DATA statements in a BASIC program, you would need to remember that the address bytes on the printout are reversed, and change them in the DATA statement.

The second generation of Radio Shack pocket computers has arrived. Shown on this month's cover is the PC-2 with its four color plotter printer/cassette interface.

TRS-80 Microcomputer News is published monthly by Radio Shack, a division of Tandy Corporation. A single six month subscription is available free to purchasers of new TRS-80 Microcomputer systems with addresses in the United States, Puerto Rico, Canada and APO or FPO addresses. Subscriptions to other addresses are not available.

The subscription rate for renewals and other interested persons with U.S., APO or FPO addresses is twelve dollars ($12.00) per year, check or money order. Single copies of the Microcomputer News may be purchased from Radio Shack Computer Centers or Computer Departments for $1.50 suggested retail each. The subscription rate for renewals and other interested persons with Canadian addresses is Fifteen dollars ($15.00) per year, check or money order in U.S. funds. All correspondence related to subscriptions should be sent to: Microcomputer News, P.O. Box 2910, Fort Worth, Texas 76113-2910.

Retail prices in this newsletter may vary with individual stores and dealers. The company cannot be liable for pictorial and typographical inaccuracies.

Back issues of Microcomputer News prior to January, 1981 are available through your local Radio Shack store as stock number 26-2115 (Suggested Retail Price $4.95 for the set). Back issues of 1981 copies are not available.

The TRS-80 Newsletter welcomes the receipt of computer programs, or other material which you would like to make available to users of TRS-80 Microcomputer systems. In order for us to reprint your submission, you must specifically request that your material be considered for reprinting in the newsletter and provide no notice that you retain copyrights or other exclusive rights in the material. This assures that our readers may be permitted to recopy and use your material without creating any legal hassles.

Material may be submitted by mail to P.O. Box 2910, Fort Worth, Texas 76113-2910, or through CompuServe. The Microcomputer News' CompuServe user ID number is 70007,535.

Notes to Program Users:

Programs published in the Microcomputer News are provided as is, for your information. While we make reasonable efforts to ensure that the programs we publish here work as specified, Radio Shack can not assume any liability for the accuracy either of the programs themselves, or of the results provided by the programs.

Further, while Microcomputer News is a product of Radio Shack, the programs and much of the information published here are not Radio Shack products, and as such cannot be supported by our Computer Customer Service group. If you have questions about a program in the Microcomputer News, your first option is to write directly to the author of the program. When possible, we are now including author's addresses to facilitate communications. If the address is not published, or if you are not happy with the response you get, please write us here at Microcomputer News. We will try (given the limited size of our staff) to find an answer to your question and, in many cases, will publish the answer in an upcoming issue of Microcomputer News.

Comments on Our Program Listing Style:

In order to make the program listings we publish easier to read, we have adopted a style of inserting spaces to enhance readability, and we separate each program statement onto a separate line. While these techniques increase program readability, they also require more memory, and may execute more slowly than the original program did.

When you are entering a program for your own use, you may wish to eliminate many of the extra blanks (see your owners manual for required blanks), and you should certainly move multiple statements up to a single line where possible.

Trademark Credits

CompuServe™	CompuServe, Inc
DIF™	Software Arts, Inc
Dow Jones NEWS/RETRIEVAL Service®	Dow Jones & Co., Inc
Personnel Manager®	Image Producers
Project Manager™	Image Producers
ReformaTTer™	Microtech Exports
Time Manager™	Image Producers
VisiCalc®	VisiCorp™
Program Pak™	Tandy Corporation
SCRIPSIT™	Tandy Corporation
TRSDOS™	Tandy Corporation
TRS-80®	Tandy Corporation

TRS-80® Microcomputer News

Information Published for TRS-80 users.

Volume 4, Issue 5 **MAY, 1982**

Price $1.50

Contents:

All prices in TRS-80 MICROCOMPUTER NEWS are in U.S. Funds.

View From The Seventh Floor

Model II Hard Disk

by Jon Shirley,
Vice President,
Radio Shack Computer Merchandising

While I was off on a trip some elves entered my office and installed a hard disk on my Model II. At $4495, they must be wealthy elves, but I am certainly not going to complain. This addition has kept me here until late in the day two Saturdays now, but it has been worth it.

Why a hard disk? Eight megabytes, that's why! Of course, a hard disk is faster than a floppy, although how fast depends on what software you have. For example, PROFILE, which I use a lot, is very fast on hard disk. A COBOL program, especially one using ISAM (which all our COBOL programs do), is much faster. One four-disk program that we use internally in our factories which previously took about 5 hours to run, now takes only 45 minutes. That's fast!

Before I get a lot of letters, it is true that the BASIC that comes with the Model II runs a little slower. The reason is not yet known, but it is under study by our people and by Microsoft, and we expect a fix soon. Machine language, COBOL and RSBASIC will all write and read files much faster.

But the real plus of a hard disk is all that on-line capacity. I have two large PROFILE systems, VisiCalc and a bunch of files, plus a lot of miscellaneous programs, and I have so much free space left that I am not sure what else to install. Not having to swap diskettes is really great.

QUESTIONS ABOUT HARD DISK

I get a lot of questions about the hard disk from our people as well as from you so here are a few answers.

The most common question is **"Will my programs run?"** Well, if you wrote it in any of our languages, yes it will. If you wrote it in machine language, used our documented SVC calls and did not violate high memory, it will. Almost all of our programs will run on hard disk except for SCRIPSIT,™ the Program Editor, ReformaTTer (which is floppy only anyway), and the Bisynch packages. Why won't they run? Because we broke the rules. Even if SCRIPSIT did run, it would not be able to address more memory so a new version is required anyway. If you plan to buy a hard disk and have Scripsit you will be able to upgrade.

"I would think that the hard disk operating system takes up more RAM. If I have a big program will it still run?" Yes it will. We installed 16K more RAM on the hard disk interface board to get around that problem. So a hard disk-equipped Model II really has 80K. In fact, if you want to get picky, it has 82K since all Model II's have 2K of Video Memory which do not use up main RAM.

"I know you use the Model II floppy disk for backup, but what happens if I write a file bigger than one disk can hold?" Ah ha! Our hard disk operating system is very clever. When the SAVE function is used to backup to diskette, it knows how much floppy disk space is available and automatically tells you to insert another diskette when needed. And RESTORE, which puts the SAVE'd information back on the hard disk, knows which diskette you should have and keeps you from getting your disks out of order.

If you are shopping and see another hard disk system, perhaps at a slightly lower price and are tempted to buy it ask some questions. Will it run the software I have? Can I backup over multiple diskettes? Do I lose RAM memory? There are a lot of gotcha's to this, so be careful. By the way, one of our famous competitor's new hard disk will let you write a file bigger than one floppy, and they use a 5¼ inch floppy, but you cannot back it up.

All of our computer centers have the hard disk system and if you can find the time, get a demo. It's a lot of fun to use that power.

COLOR COMPUTER DOS

Many thanks to 68XX magazine for the praise for the Color Computer disk operating system and manual. After all the articles about our other operating systems, it is nice to see a very technical magazine praising our work. We are rather proud of the Color Computer disk system. It does a lot, yet does not take up RAM. And perhaps its most unique virtue is that it is a part of BASIC. You are never in DOS or in BASIC; you are just always in BASIC.

MODEL III TIME MANAGER

If you are a Model III disk system owner, take a look at Time Manager. This is a program that almost everyone can use. It is a dynamic calendar that reminds you of what you need to do each day, stores what you did do, and carries forward what you did not get done. It also does a lot of other things, but why not drop into any store and take a look? This program is so useful that I have noticed that some of our offices which were already equipped with Model II's have added Model III's just to run Time Manager.

REGARDING YOUR LETTERS

Finally, I would like to publicly thank some of you who have taken your valuable time to write us nice letters. We have a lot of hard-working people who do get tired of the usual problem correspondence and who just love those letters that are complimentary of good service or a good product. We do read all our mail, and we do answer all our mail (OR ELSE). So if you have a suggestion, send it in. We want to please you and we adopt more suggestions from our customers than you might think.

Until next month.

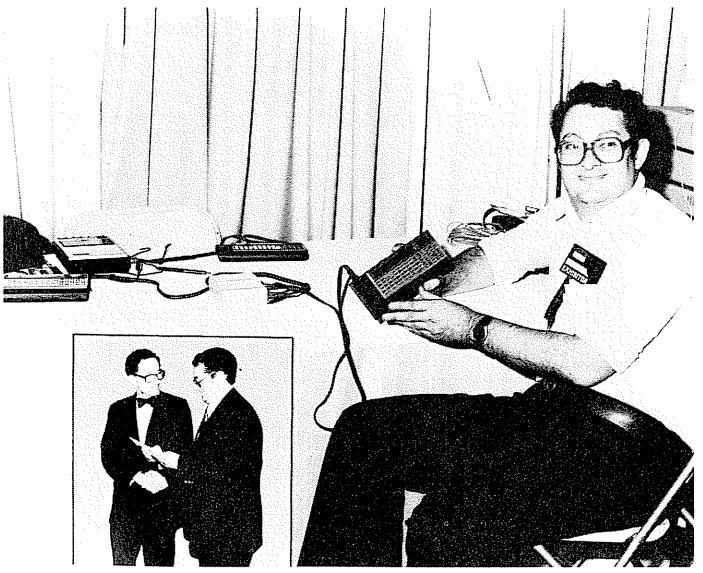
Johns Hopkins Search

Results of the First National Search for Applications of Personal Computing to Aid the Handicapped

The central theme of this issue of Microcomputer News is Language. Since language is a vehicle for communication, and many of the ideas presented for the search involved aiding communication for handicapped individuals, it occurred to us that this issue might be a very appropriate place to present some of the results from the Johns Hopkins National Search. The national competition looked for ideas, devices, methods and computer programs to help handicapped people overcome difficulties in learning, working and living, and to successfully adapt to home and community setting.

A POCKET TELECOMMUNICATOR FOR THE DEAF

The TRS-80 Pocket Computer (PC-1) was presented by Dr. Harry Levitt as an inexpensive, portable telecommunicator for the deaf in his top-prize entry. Levitt, a hearing and speech professor at the City University of New York, captured the competition's $10,000 first prize.



Levitt demonstrated how the Pocket Computer can be used in conjunction with a private or public telephone for instant communications. Messages previously stored in the computer's memory can be transmitted instantly, or a message can be typed into the computer's keyboard. In addition, the Pocket Computer has a compact and inexpensive line printer accessory available for a permanent, hard-copy record of the communications.

"Perhaps the most important advantage of all," reports Levitt, "is that the use of a Pocket Computer as a convenient, inexpensive communication device introduces the deaf telecommunicator user to the concept of an intelligent, computer-based communication system of almost unlimited scope and flexibility."

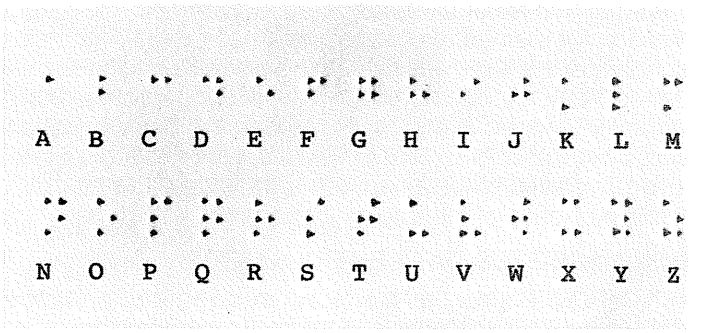
In addition to its advantages of compactness, memory, and telephone time and cost economies, Levitt believes that the system could help reduce communications barriers between the hearing and the deaf.

THE MICRO-BRAILLE SYSTEM

Randy W. Dipner of Colorado Springs, Colorado, has been able to produce Braille text using only a bicycle inner tube and off-the-shelf equipment from Radio Shack. Dipner's Micro-Braille System, which uses the Model II Computer and a Daisy Wheel II printer, allows production of standard raised-bump Braille text by people who have no training in Braille or Braille transcription.

Dipner's system won honorable mention and a $500 cash prize in the Johns Hopkins search.

Dipner has cut a section of soft rubber from a bicycle tire inner tube and used it as a sleeve over the hard rubber roller of a Daisy Wheel II printer.



Dipner also wrote a special printer-driver program that translates standard typewriter characters into their corresponding Braille dot patterns—but in reverse. Then, using high fiber paper to prevent punch-through, the Braille patterns are typed onto paper; only the period character on the daisy wheel type font is used. The periods leave impressions (dents) on the paper, which provide right-reading raised Braille characters when the paper is turned over.

Dipner's Micro-Braille System is compatible with Scripsit word processing software.

LEARNING FINGER SPELLING

Frostproof, Florida, high school student Daniel K. Johnston has developed computer programs that teach hand-sign language (or "finger spelling") for communication with the hearing impaired. Johnston's programs were written for the TRS-80 Model III and Color Computers.

These "Deafsign" programs, which use video graphics and written video instructions to teach sign language spelling and symbols to the hearing impaired and others, earned Johnston an honorable mention and $500 cash award in the search.

Johnston's programs include lessons on letters of the alphabet and word-symbol hand signs (including animated movements in the Color Computer version); increasingly rapid drills with written prompts included; and eventually, increasingly rapid drills with hand signs only.

Johnston's inspiration for developing these programs was a friend with a serious hearing impairment.

PROGRAMS FOR LEARNING DISABLED

Sandra Jackson, Judith Simmons and Lawrence Wedig of San Antonio, Texas, were awarded an Honorable Mention for their series of programs designed to help students with various levels of learning disabilities. These programs include Underachiever Motivation; Improvement of Underdeveloped Basic Skills; Increasing Concentration and Comprehension Skills; Assisting the Emotional Disturbed toward Normal Adjustment; and Assistance in Rapid Adoption of the English Language.

MENU ASSISTED DATA ENTRY SYSTEM

Jeffrey Fisher, of Houston, recognized that the typewriter-like keyboard used to enter data into most computers presents an obstacle to many handicapped persons, so he developed a remarkable system that allows data entry without a keyboard.

Fisher's "Menu Assisted Data Entry System" allows the use of external switches in place of the keyboard. It was designed for the Model III computer. The entry was one of the top thirty national-prize-winning entries in the Johns Hopkins search.

The system presents the user with an on-screen "menu" of letters, numbers, punctuation, special characters and commands. Then, using special switches that respond to chin movement, head pressure or puffs of breath, for example, the handicapped person can eventually build complete messages and print, transmit or otherwise communicate them.

PROGRAMS FOR COGNITIVE REHABILITATION

Survivors of brain injuries, including stroke victims, can now find help in a new series of nine computer programs for the Model III.

Rosamond Gianutsos of Adelphi University and Bellevue Hospital/NYU Medical Center and Carol Klitzner of Computer Software Solutions presented Computer Programs for Cognitive Rehabilitation: Personal Computing for the Survivors of Brain Injury (or COGREHAB, for short) during the search's national finals.

The COGREHAB software and accompanying handbook material provides a complete system for diagnosis and treatment of cognitive disorders, for use by professional, or by para-professionals, family members or the disabled themselves under professional guidance.

Program elements include speeded reading of word lists, visual field reaction time measurement, searching for shapes, free memory recall, memory span, triplet recall and sequence recall. Administrative elements include patient log and a visual field test visual angle calculator.

For additional information the COGREHAB materials contact Life Science Associates, One Fenimore Road, Bayport, NY 11705.

MESSAGE GENERATOR FOR THE NON-VERBAL

Richard Buus provided a means whereby people with severely impaired motor abilities can print messages on a computer video screen or, optionally, on a sheet of paper using a mechanical printer. The heart of the system is a Radio Shack computer equipped with a special interface circuit and programmed software provided by Buus.

The program is capable of displaying word lists (menus) on the video and providing a moving arrow to point at words on the menu one at a time. A word is selected by a simple switch closure (or optionally, if an external switch is not used, by pressing any key on the computer keyboard) when the arrow is on the desired word. Upon selection, the word is placed at the top of the screen and the process is repeated for additional words. If a printer is added to the system, the program permits transferring the message from the top of the screen for more messages.

The menus follow a layered hierarchy whereby a master menu first appears which permits selection of broad word categories or certain control functions such as speed setting, printing, or erasing. Upon selection of a word category from the master menu, a new menu then appears on the screen with its contents determined by the word category. If further breakdown is necessary, any of the words in this new menu can point to a sub-menu with a further breakdown of the words. Final selection can be from either the menu or sub-menu as appropriate.

For further information on computers to aid the handicapped write to:
IEEE Computer Society
P.O. Box 639
Silver Spring, MD 20901

NAME _____

ADDRESS _____

CITY, STATE _____ ZIP _____

Machine Language Sort

by William Barden, Jr.

Let's face it, readers, you're going to have to do some work if you're following this column. I could simply present you with a description of a new set of instructions every month; in a year you'd know in intimate detail how most of the 6809E machine language instructions worked. But the question is, could you put them together to implement some neat Color Computer assembly language program that would amaze and amuse your friends at your next party? Probably not.

I've taken another approach here — I'll provide you with a "neat" program, show you how I went about implementing it in assembly language code, and let you consult 6809E reference books for those instructions that you're unsure of. The advantage of this approach is that you can see the whole design cycle for a piece of useful code. Most of the difficulty in creating assembly language programs is not so much in decoding how the instructions work, but in analysis and design; the assembly language instructions fall into place (to a certain extent) once the program design has been established.

The "neat" program we'll be talking about in the next two columns is a program to sort a one-dimensional string array. This is an interesting assembly language application because it is hundreds of times faster than a sort done in BASIC.

In a test case that you'll see later, I generated a 101-element string array called A$ (I'm not too creative these days) and then filled it with random ASCII data of random length. A$(0) might be "⟨DFGTCX GF" and A$(25) might be "BCFF'(F", for example. The assembly language subroutine was then used to sort the data. How long do you think the sort took? (No, the reader from Dudtown, Iowa is grossly wrong!) Less than a second! This is a far cry from a typical BASIC sort, which took 20 minutes. And that, in a nutshell, is why assembly language is worthwhile; it can provide some very powerful processing for your system.

The sort program, called "SRTSTR", for "Sort String," will operate with a string array of any number of entries, and with string entries of any length. "Null" entries are sorted into the end of the array.

FOR YOU TRADITIONALISTS...

Traditionally, creating a program has been divided into a number of steps:

Step 1: Design. Research and analyze the problem.
Step 2: Flowchart. Use standard "flowcharting" symbols to show a "schematic" of how the program operates.
Step 3: Coding. Write down the assembly language statements of the program.
Step 4: Debugging. Run the program and correct errors or deficiencies.
Step 5: Documentation. Document how the program works.

This approach is still valid, although it has been modified in recent years by such things as "top down design" and "structured programming" techniques. We'll use the 5 steps above, with some modifications, in showing you how to implement SRTSTR. We'll pay particular attention to the design, flowcharting, and coding phases. You can use the same approach for other assembly language programs you'd like to implement.

DESIGNING SRTSTR

The design phase of any program development can be divided into a number of substeps. After you've been coding for some time, you do these substeps automatically in your own mind. They are roughly:

Defining the problem.
Researching the problem.
Determining its feasibility.
Looking at alternative design approaches.
Picking a design approach.
Defining the design.

DEFINING THE PROBLEM

In this case the problem is a lot simpler than "How do we implement a BASIC compiler?" or "How do we design a nuclear reactor monitoring program?." It's simply, "How do we implement a fast string array sort?." It seems like a minor task, but then again everything is relative.

RESEARCHING THE PROBLEM

The next substep is research. To implement SRTSTR, we've got to become experts on strings, or at least learn enough to make us dangerous. In this case, unlike most others, it means getting into some of the "internals" of Color Computer BASIC. This will take some doing, so bear with me for the following descriptions.

The first place to start is in the Color Computer Extended Color BASIC manual. Buried in the text on "USR FUNCTION ARGUMENTS" is a description of some of the internal descriptions of strings. It seems that each string is defined by a five-byte "descriptor block," as shown in Figure 1.

The first byte of the block is the length of the string in bytes. The third and fourth bytes contain the address for the string. The second and fifth bytes are reserved for the use of the BASIC interpreter.

Reading further, we can see that the VARPTR command returns a pointer to the string descriptor block. If we wanted to find information about string ZZ$, for example, we'd find the location of the string descriptor block by first finding its address.

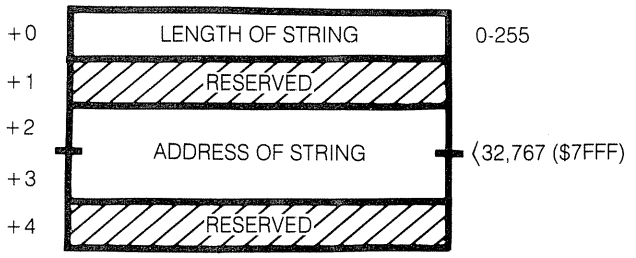


Figure 1. String Descriptor Block

```
1ØØØ N=VARPTR(ZZ$)   'N is address of dscrptr blk
```

Once we had the address of the descriptor block, we could then use PEEKs to look at the contents of the block, such as the location of the strings:

```
1ØØØ N=VARPTR(ZZ$)
1Ø1Ø PRINT "STRING ZZ$ ADDRESS IS",
     PEEK(N+2)*256+PEEK(N+3)
```

Note that the address of the string is in standard 6809E address format — 16 bits, defining a location of $0000 through $FFFF, where "$" indicates hexadecimal.

Now that we know how to access the descriptor blocks, we can compile some notes on strings. Some of the things that we can glean from BASIC code that examines the blocks are:

- Strings that are embedded in BASIC statements, such as "1000 A$="THIS IS STRING NUMBER 1" remain in the statement and the descriptor block reference to them contains the address of the string within the BASIC statement.
- Strings that are assembled or modified, such as the A$ string in "1000 A$=B$+C$+CHR$(20)" are located at the "string storage" area in high RAM.
- The length of a string is equal to the number of characters in the string and cannot exceed 255.
- "Null" strings, such as A$="", are strings of 0 lengths, and this length is recorded in the descriptor block.

We're rapidly becoming experts on strings while researching the SRTSTR problem. Now what about string arrays?

A string array, of course, is defined by a DIMension statement like "1000 DIM A$(100)". The DIM statement allocates a block of storage for the string array A$. You must be certain to set aside enough storage area for the string array by an initial "CLEAR 3000" or similar statement.

The VARPTR function will return the address of any array element descriptor block, such as A$(0), A$(25), or A$(100). A little investigation in BASIC reveals that the addresses from the VARPTR are offset by 5 from each other and that descriptor blocks for an array are in a "list," as shown in Figure 2.

The "array variable list" for a one-dimensional string array contains not only the descriptor blocks for each of the string elements in the array, but a "header" of data pertaining to the array.

This header consists of 7 bytes before the "0th" element descriptor block.

The first two bytes (-7 and -6) are the array name. The second byte of the array name, if any, has 128 decimal added to the value for the character, as you might find out from PEEKing with BASIC.

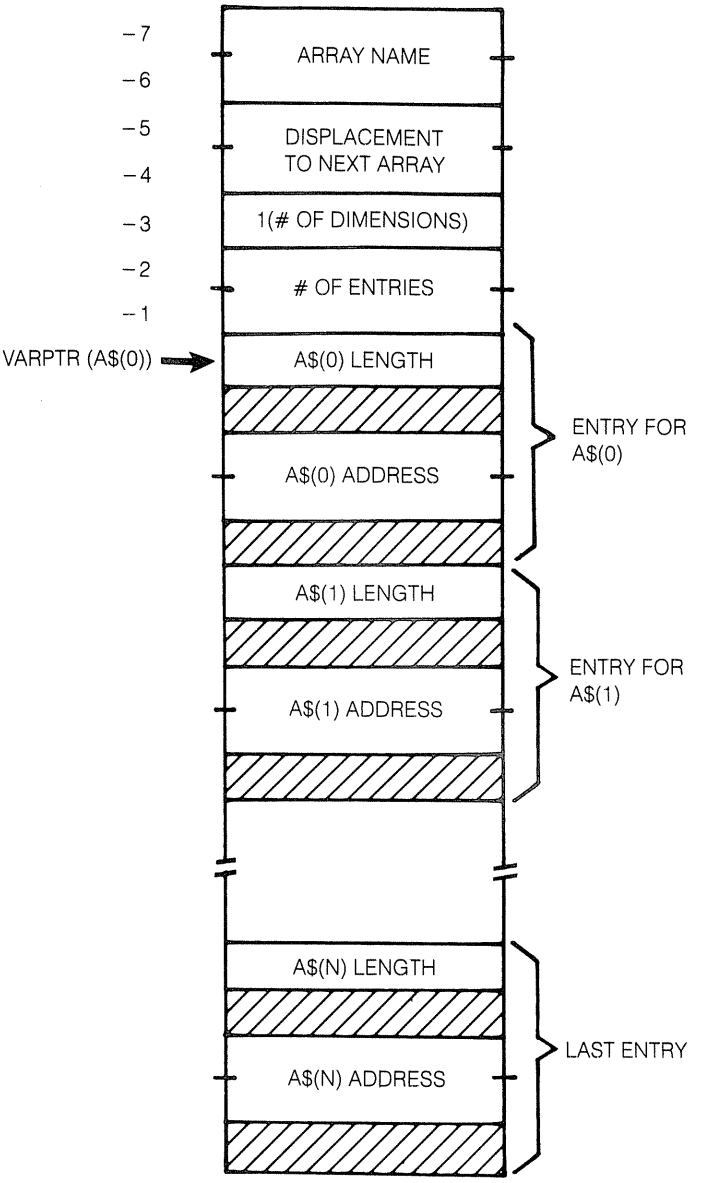


Figure 2. String Array Variable List

The next two bytes (-5 and -4) hold the "displacement to the next array." This is a 16-bit value that when added to the address of the array variable list will point to the **next** array variable.

The next byte (-3) contains a 1 for one-dimensional arrays.

The next two bytes (-2 and -1), contain the number of entries in the array. An array defined by "DIM A$(100)", for example, would have a 100 in these two bytes.

We'll only be using the "number of entries" parameter in SRTSTR.

The remainder of the array variable list is made up of one 5-byte descriptor block for each element of the array. The "length" byte of each descriptor block contains the length of the string for the element, while the "address" bytes contain the address of the string. A typical situation is shown in Figure 3, where three array strings are shown with their corresponding entries. In this case, the strings were in BASIC statements, rather than in the string storage area, accounting for the low address values.

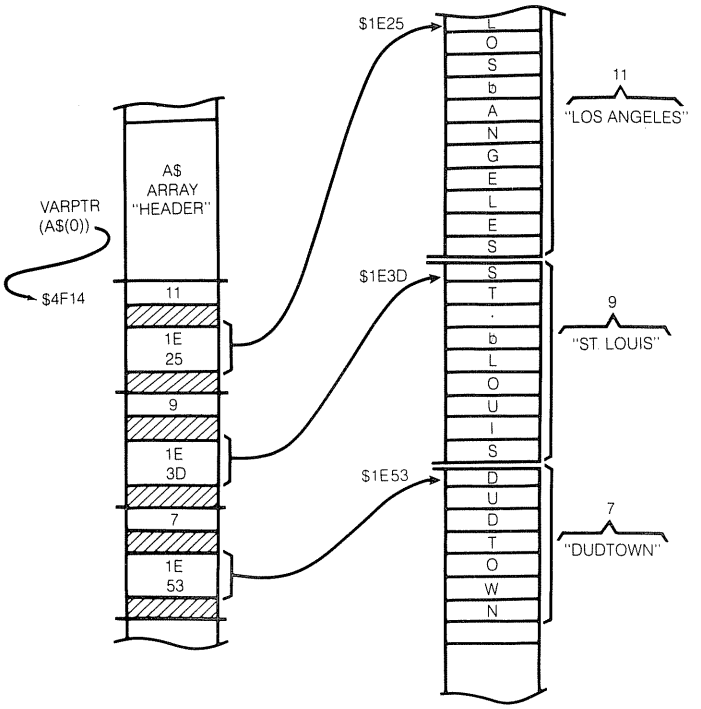


Figure 3. String Array Length and Address Variables

IS IT FEASIBLE?

In case you've forgotten where we are, we're in the middle of the design phase for SRTSTR. We're now experts on strings and string arrays. We must now give some attention as to whether it's feasible to implement SRTSTR.

It certainly sounds messy to sort strings if they are all over BASIC programs and the string storage area. Sure, we **could** find the string addresses in the string descriptor blocks, go to the BASIC program and string working storage area, and rearrange BASIC statements and the string working storage area to sort the strings, but it would take more research and a significant program to do the sort.

What about sorting a string array by changing only the string descriptor block data? This sounds a lot more reasonable. Instead of moving strings around, we'd only have to move the "pointers." This scheme is shown in Figure 4 for a few sample strings of a string array. The strings remain stationary; only their descriptor blocks are reordered within the string array list. We know where the strings are and can easily compare one string with another to get the proper order.

LOOKING AT ALTERNATIVE DESIGN APPROACHES

Now that it seems feasible, let's give some thought about how we're going to actually order the data. This is a complicated subject. Entire books have been written on "sorting" and "searching" techniques. This may require extensive research on its own! I'm not a sorting and searching expert, but let me give you some ideas on ways to proceed.

We could set up a block of memory equal to the amount used in the array list for block descriptors. We could then look at each string and find the "smallest" (the one closest to the beginning of the alphabet), put the descriptor block in the first five bytes of our "working storage," delete the array descriptor block (by marking it with all ones), and repeat the process for all the strings in the

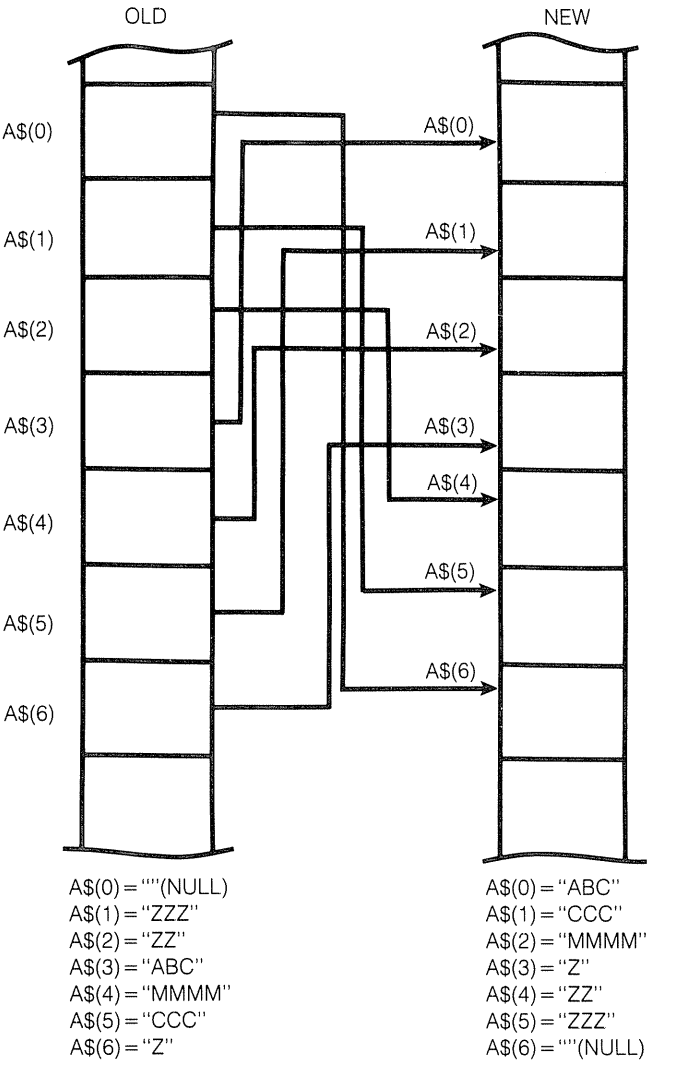


Figure 4. Ordering a String Array

array, as shown in Figure 5. At the end, we'd move back the new, sorted descriptor blocks.

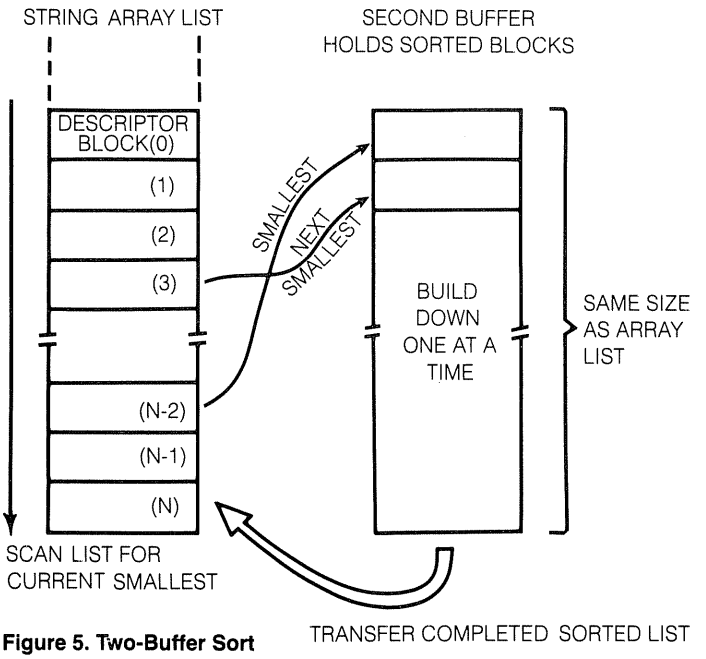


Figure 5. Two-Buffer Sort

This would require a large block of memory, but would be fairly fast.

A second method might be a "bubble sort." In a bubble sort, only the original block of memory containing the data items to be sorted is used. The technique is to compare each item with the one below it. If the item below is "less" than the current item, then the two items are swapped, with the "lighter" item bubbling to the top.

The compare and swap process is repeated for the entire list. After the last two items have been compared, if any swap has been made, an additional pass through the list is made. The passes continue until no swaps have occurred, indicating that the list is ordered. Figure 6 shows the process in a simple example.

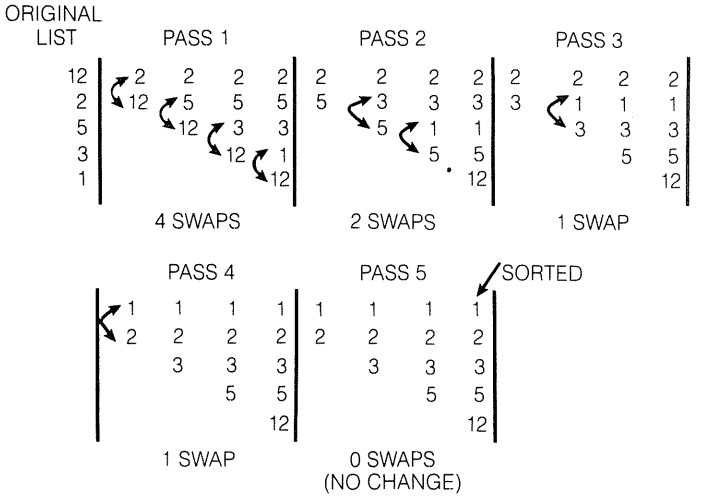


Figure 6. Bubble Sort Action

The bubble sort takes a long time for large lists, but has the advantage of using only the original list memory space.

PICKING A DESIGN APPROACH

There are many more techniques that could be used, including such sorts as the "Shell" and "Shell-Metzner" sorts, but let's decide in favor of the bubble sort because of its efficient use of memory. The slow speed of the sort will not be a detriment in assembly language. Of course, we're using a little intuition here, and not doing the proper design effort in studying alternatives. Entire programming departments have fallen as a result of such quick decisions, but let's rush in . . .

DEFINING THE DESIGN

Now that we've defined the basic design approach, let's do some serious thinking about what kinds of problems are involved, based upon our original research into strings.

First of all, what is an "ordered list of strings"? What about "AAA" versus "AA"; which should come first? What about "null" strings, strings of 0 length? The DIM function sets all array elements to "null" strings initially, and if some elements haven't been used, we're going to have to consider them in the sort.

The order we'll be using will be based on the actual numeric value of each character in the string. Fortunately, alphabetic and numeric characters are ordered in "ascending" order and this is no great problem. Special characters, however, are not so obvious. Special characters are defined by ASCII codes, and in lieu of another defined order, we'll simply order the string characters on the basis of their ASCII codes, putting "Pascal Blaise" before "Pascal,Blaise" and "PASCAL, BLAISE" before "Pascal,Blaise".

We'll make the arbitrary judgement to put strings which are leading portions of larger strings before larger strings; "Blechman,Fred" will appear before "Blechman,Freddy", for example.

We'll also make the arbitrary judgement to put all null strings at the end of the array. This will avoid leading null strings on printouts and displays.

At this point it pays to sit down and do a good deal of thinking about some of the problems and special cases which may come up in solving the programming problem. Time spent in the design phase will pay for itself later.

FLOWCHARTING

Is the flowchart an anachronism in these days of interactive BASIC interpreters? Not so much in assembly language coding, where things must be thought out more carefully than in BASIC. Also, flowcharting is valuable any time complex, long, or long complex programs of any type must be designed.

Flowcharting doesn't have to be done with standard flowcharting symbols. Some programmers **never** flowchart. Others never do and wish they had. The important thing is to methodically set down an algorithm for solving the programming problem.

In the SRTSTR design, it's probably a good idea to formalize the approach. It's not quite small enough to simply "sling down" some assembly language code. (OK. I'll be honest. I **did** sling down some assembly language code. But I paid for it! Two serious logic errors showed up that could have been detected by flowcharting. Flowcharting would have saved me time in the long run. Do as I say, not as I do . . .)

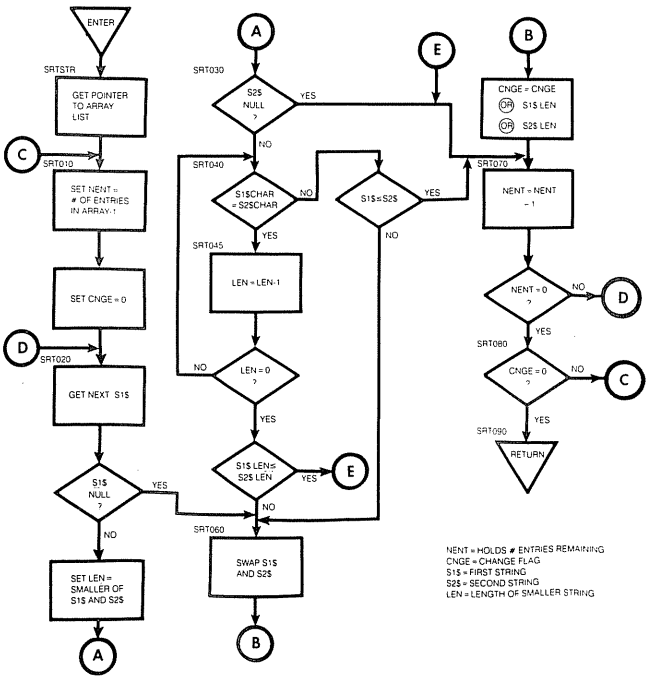


Figure 7. SRTSTR Flowchart

Figure 7 shows the SRTSTR flowchart. This is a "logical" flowchart not tied in with the architecture of the 6809E, although it could be. As a matter of fact, the catch here is that unless you've done quite a bit of assembly language programming, the flowchart you define at this point may have to be reworked when you get to the actual coding. Of course, the same thing could be said for the design phase; it's tied in to the microcomputer to be used, the system architecture, the microprocessor, and so forth. We've got to start somewhere, however . . .

The symbols used in the flowchart are "standard" flowcharting symbols. I say standard, because there are variations from company to company and from programmer to programmer. The rectangle is a "process" box and defines most operations. The diamond is a "decision box" and defines a point at which a branch decision is made. The triangles define the entry and exit points. The circles are "on-page connectors" that are a substitute for "flow" lines that might make the flowchart confusing.

The labels above the processing or decision boxes are put in for reference only, but would normally correspond to labels in the assembly language program.

The "flow" is normally from top to bottom and from left to right.

Let's follow the flowchart to see how our design will be implemented. First of all, we get the pointer to the array list. This would be the VARPTR pointer. The number of entries in the array is picked up from the "header" and put into a variable called NENT. One is subtracted from this value, as the last string has no "lower" string for a bubble-sort compare.

Next, a CNGE variable is set to 0. This variable is set any time a bubble-sort swap occurs so that another pass can be made through the compare process.

We're now at SRT020. The first pair of strings is now ready for comparison. Before the compare, however, the first string S1$ is tested for a null length. If it is a null, a swap of S1$ and S2$ is always done, as we must move null strings to the bottom of the array.

Before the compare also, we set variable LEN to the smaller of S1$ and S2$, so that the compare of strings doesn't "run out of characters" for the compare on one of the strings if they are of unequal length.

Next, S2$ is tested for a null length. If it is null, the swap will never be done as the null string is already at the "bottom" of the pair.

The compare is (finally) performed at SRT040. The compare compares the first character of S1$ with the first character of S2$, the second character of S1$ with the second character of S2$, and so forth. If any two characters are not equal, the bubble sort swap is done if the S1$ character is greater than the S2$ character. If the characters in the two strings are equal to the end of the shorter length, the swap is done if the S1$ length is greater than the S2$ length.

If the swap of the two strings is done, the two string descriptor blocks are swapped in the string array list; the actual strings themselves are left untouched in their BASIC statements or string working storage.

If a swap is done, the CNGE variable is changed from 0 to non-zero. The result is CNGE OR S1$ length OR S2$ length, where OR stands for a logical OR, as in BASIC. The reason for this strange logic is that it is possible to swap two null strings continuously. ORing in the length avoids this error condition. I might add that this is one of the error conditions discovered in debugging that could have been caught with more thought in the design and flowcharting phase of SRTSTR.

The "working count" of NENT is decremented by one. If this count is not 0, there are remaining pairs of strings to be compared, and the next pair is picked up at SRT020. Note that the next pair starts with the previous S2$! If the NENT count is 0, one complete pass has been made through the program, and the CNGE flag is tested. If CNGE is not 0, a swap of at least one non-null string has occurred, and another pass through the array list is made at SRT010. If CNGE is 0, the sort is done, and a return to the calling program is made.

Will this "algorithm" work? One way to test it is to implement it by using a BASIC program. We've done that with the program of Listing 1. This short program sets up random data in a 101-entry array and then sorts the strings by the SRTSTR algorithm. Run the program to see how long a BASIC version of SRTSTR takes. The current array element number is displayed until the data is sorted.

Listing 1

```
1ØØ ' BUBBLE SORT DEMONSTRATION
11Ø CLEAR 4ØØØ
12Ø DIM A$(1ØØ)
13Ø PRINT "***RANDOM ARRAY***"
14Ø FOR I=Ø TO 1ØØ
15Ø IF RND(1Ø)=5 THEN GOTO 19Ø
16Ø FOR J=Ø TO RND(15)
17Ø A$(I)=A$(I)+CHR$(RND(58)+32)
18Ø NEXT J
19Ø PRINT A$(I)
2ØØ NEXT I
21Ø CLS
22Ø PRINT "***SORT STARTING***"
23Ø K=Ø
24Ø FOR I=Ø TO 3ØØ
     : NEXT I
25Ø C=Ø
26Ø FOR I=Ø TO 99
27Ø PRINT @ 26Ø,"PASS";K
28Ø PRINT @ 275,"INDEX";I
29Ø IF A$(I)="" THEN GOTO 32Ø
3ØØ IF A$(I+1)="" THEN GOTO 33Ø
31Ø IF A$(I)<=A$(I+1) THEN GOTO 33Ø
32Ø  B$=A$(I+1)
     : A$(I+1)=A$(I)
     : A$(I)=B$
     : C=C+LEN(A$(I))+LEN(A$(I+1))
33Ø NEXT I
34Ø K=K+1
35Ø IF C<>Ø THEN GOTO 25Ø
36Ø CLS
37Ø PRINT "***SORT ENDED***"
38Ø FOR I=Ø TO 3ØØ
     : NEXT I
39Ø FOR I=Ø TO 1ØØ
4ØØ PRINT A$(I)
41Ø FOR J=Ø TO 5Ø
     : NEXT J
42Ø NEXT I
```

Next month we'll show you how the program is implemented in Color Computer assembly language. If you feel ambitious, try implementing your own assembly language version, using SRTSTR flowchart — I'll warranty its validity, but only for 30 days until the next column!

Radio Shack Languages

This month's article will be about our various languages — FORTRAN, BASIC (Interpreter & Compiler), COBOL, Assembly, and Pascal.

Definitions:

Source Code: Generally a data file produced by the programmer and a program called an Editor. The source code is written in the particular language you are using.

Object Code: The data file produced by the Language compiler from your Source Code file. The purpose of object code is to provide the computer with machine language instructions to perform the task programmed in a high level language. Object Code is generally very machine dependent.

Languages:

FORTRAN — We have FORTRAN compilers available for the Models I, II & III. All versions of FORTRAN work the same. For example, if you have written source code on the Model I, then this same code can be used on Models II and III.

This FORTRAN has no overlays. What this means is that your entire FORTRAN program must fit in memory at one time. The ability to overlay means that you have the capability to load into memory a portion of your program and then overlay that portion with a new portion when you are through. The July, 1981 Microcomputer NEWS contained an article (page 11) by Robert Minty which explains one method of creating FORTRAN programs in "modules" which can share an in-memory COMMON data area.

BASIC — We have the BASIC Interpreter, both ROM and Disk, and the BASIC Compiler, Disk only. The BASIC Interpreters are not fully compatible with the Compilers; the reason being that the Interpreters are written by Micro-Soft and the Compilers by Ryan McFarland. This means the two languages are going to have syntax incompatibilities. In the Question and Answer section, we will describe one method of transferring programs between Interpreted BASIC and Compiled BASIC. These compilers are available for Models I, II & III.

COBOL — We have received many questions about the COBOL Compiler licensed to Radio Shack by Ryan McFarland Corporation. RS COBOL is based on the ANSI COBOL revision of May 1974. It consists of an editor (CEDIT), a compiler (RSCOBOL), and an execution module (RUNCOBOL). The COBOL package operates the same way on the Model I, II & III. RS COBOL offers sequential, random and indexed-sequential (ISAM) files. With a few exceptions, because of different screen formats, COBOL programs compiled on one model TRS-80 should run on a different model. This is one of the features provided by the RUNCOBOL modules.

Some COBOL users face a reorientation with microcomputer COBOL when compared to the COBOL development systems they are used to in a main frame environment. In such an environment, there are many utilities that perform functions such as creation and operations on data files. With RS COBOL, COBOL data files are most normally created with a COBOL program. One exception to this is a limited compatibility of one-key COBOL ISAM files with Radio Shack Compiler BASIC ISAM files as explained in the Compiler BASIC manual. To create a COBOL data file, the file must be opened for output only. Prompts to the screen and input from the keyboard to build the file are accomplished through the accept and display statements. Sorting the file must be accomplished through user programmed logic (i.e., RS COBOL does not support the SORT verb.)

Printer control has been another subject of concern for COBOL users. In BASIC, you just LPRINT character strings (CHR$) to control the printer. This is accomplished in COBOL by defining a variable in Working-Storage with a usage of COMP-1 (i.e. 01 PRT-CNTRL PIC 99 COMP-1.), placing the proper value in the variable, and then writing it to a printer file. To select a file for the printer: SELECT file name ASSIGN TO PRINT, "PRINTER".

Radio Shack COBOL provides a way to handle business information in an efficient manner. COBOL programs are easily maintained because the variables are laid out in one section of the program, files in another, and logic can be structured in such a way as to be much easier to trace than BASIC. The fact that Radio Shack COBOL follows the American National Standards Institute COBOL (with both enhancements and limitations) gives the user a wealth of resources for learning aides. Any ANSI COBOL instruction book would offer assistance in understanding the Radio Shack COBOL instruction set. Further help is available through the Computer Customer Service Language Group.

Assembly — We have, at present, two (2) types of assemblers, the Series I and the Disk Editor/Assembler. The Disk Editor/Assembler is a Macro Assembler with a linking loader. This means you could have a diskette with all of your much-used subroutines on it and be able to link only the subroutines you wanted into a main program. As with the FORTRAN, there is no overlay capability.

The Series I has no Macros or linker, however the Editor can chain in different subroutine source files, and combine them into a larger program which can then be assembled into a larger object code module. Otherwise the coding is similar to the Disk Editor/Assembler.

Disk Editor/Assembler is available for Model II. Series I is available for all three machines and can be purchased on either disk or tape for Models I and III.

The Color Computer Program Pak based Editor/Assembler includes an integrated text editor and assembler allowing efficient development of 6809 software. In-memory assembly allows object code to be asembled directly into memory, where the debugger allows quick

debugging of the code. Storage is on cassette tape.

Tiny Pascal — Tiny Pascal is currently available only on Model I tape. We will have a Model III version of Tiny Pascal shortly.

The syntax and structure of Tiny Pascal is essentially the same as standard Pascal. The Radio Shack Tiny Pascal compiles to an intermediate object code which executes much more quickly than our standard BASICs. If you want to get started in Pascal, this is a good program to begin with.

Books you might examine include:

Programming in PASCAL; Grogono. Addison-Wesley, 1978.

PASCAL: User Manual and Report; Jenson & Wirth. Springer-Verlag, 1974.

A Primer on PASCAL; Conway, Gries, and Zimmerman. Winthrop Publishers, 1976.

PASCAL, An Introduction to Methodical Programming; W. Findlay and D. A. Watt, Computer Science Press, 1978.

COMMONLY ASKED QUESTIONS

Question 1: What causes a Load File Format error in COBOL?

Answer: This happens when there is a problem with the diskette or the disk drives. There may be physical or magnetic defects on the part of the disk where the program you are accessing is located. There may be disk drive problems — head alignment, motor speed too slow, etc. To find out which of these two most likely is causing your problem, try other disks, copying the program to another disk, or another copy of your program disk, etc. If this fails to get round the problem, take your system to the store to have the drives checked.

Question 2: While running a COBOL program, an error "UNDEFINED FILE" occurs. Why?

Answer: An attempt has been made to OPEN a nonexistent file for some function other than OUTPUT. A file must first be created with open output, and records written to it before the file can be opened for input, or I-O, to process the data.

Question 3: While using the runtime diskette in compiler BASIC, error messages such as ERR 24 or 28 on access to files (usually ISAM) occur, which do not occur if I use the full compiler BASIC diskette. Why?

Answer: You have to copy RSBASIC/LIO and RSBASIC/LIB to the runtime diskette. These files are missing on the runtime diskette.

Question 4: How do you transfer Interpreter BASIC programs to Compiler BASIC disk on Model II?

Answer:

1. LOAD "file name"
2. SAVE "file name 1",
3. COPY file name 1:0 (Interpreter BASIC) To file name 1:1 (RSB)
4. RSBASIC
5. OLD file name
6. SAVE file name 2/BAS

Syntax corrections must then be made before running the program.

Question 5: Is it possible to delete the space used by indexed records which have been flagged as deleted in RSBASIC or RSCOBOL?

Answer: No.

Question 6: Why does the machine hang up when handling strings in BASIC?

Answer: Because it is doing a "garbage sort" or "gargage collection" routine, trying to free string space. The different ways to improve the response time would be to reduce the usage of string handling functions such as A$ = B$, LEFT$, MID$, etc. Also more string space can be cleared. See the October, 1981 Microcomputer News for a more complete explanation.

Question 7: How can I move machine language programs from tape to disk on a Model I or III?

Answer: Tape programs can be moved to disk using Disk DEBUG. You would have to know the starting, ending and entry point addresses of each of the tape programs. This does not, however, mean that all tape programs will run on disk, nor does it give a program which uses tape I/O disk capabilities. Further, moving a program from tape to disk will not change a Model I only program into a Model III program.

To know the answer to which tape programs would run on disk, check with your local Radio Shack store or call Customer Service. However, we can only provide addresses for our software and only of those that we support transferring to disk. See the March, 1982 Microcomputer News for a detailed article on moving Model I/III machine language programs to disk. This article includes a complete procedure as well as the needed addresses to accomplish the move.

Question 8: How do I prevent extra line feeds when doing a top of forms command with a printer hooked up to a Model I/III.

Answer: On the Model I, you have to get the LPC driver program from your local Radio Shack store and load in. (This program is available at no charge, #700-2007.) On the Model III LPC is part of TRSDOS and can be loaded from TRSDOS READY by typing in LPC.

Question 9: When doing arithmetic operations on Model I, II, or III with decimal numbers, why is it that the final value is sometimes off a little?

Answer: The result of arithmetic operations on either Models I, II, or III is off by a small value because of a rounding problem in the BASIC on these computers. One way around the problem is shown in the following example: If you had a variable A whose value was supposed to print out as 10.50, but always prints out as 10.4999, etc. use the following formula: B = INT(A*100 + .05/100).

Question 10: What causes an IE error?

Answer: An IE error implies that a TRSDOS error has occurred which BASIC can not interpret. On Models I/III, use CMD "E" and on Model II use PRINT ERRS$ to find out what TRSDOS error has occurred.

Question 11: On Model III when running MEMTEST, message ROM C 2F84 appears on screen, but 2F84 does not match what the manual tells me to expect. Why?

Answer: 2F84 is correct.

Question 12: Why does Model III display a checksum of 84, instead of 64?

Answer: The new ROM, which was released after the manual was printed, has a different value than the old ROM.

Question 13: What causes an error 4 (CRC error)?

Answer: It is most probably caused by a media fault.

The best solution would be to copy all data and program files onto another diskette, and then bulk erase the original.

Question 14: Why does a DIRECT STATEMENT IN LINE # error message occur when I am LOADing an ASCII file into BASIC?

Answer: The specified line # is greater than 240 characters long.

Question 15: On Model II, how do you use the routing procedure on page 50 (or 9/2) of the Operation and BASIC Language Reference Manual with a TRSDOS disk?

Answer: Step 3 is replaced with:

```
DEFUSRØ=&HØØ6C
```

Step 4 is replaced with:

```
X=USRØ(Ø)
```

Computer Customer Services Phone Numbers Until June 1, 1982

8AM to 5PM Central Time

Model I/III Business Software
Outside Texas 1-800-433-5641
In Texas 1-800-772-5973

Model II Business Software
Outside Texas 1-800-433-5640
In Texas 1-800-772-5972

Education Software
Outside Texas 1-800-433-1679
In Texas 1-800-772-5914

All Other Calls Related to Computers
Outside Texas 1-800-433-1679
In Texas 1-800-772-5914

Switchboard — 1-817-390-3583

Phone Number Changes, Part Two

In last month's Newsletter we notified you of the plan to change our department's telephone system on June 1st. This month we want to tell you how to determine which group to contact within Computer Customer Service in Fort Worth, should you elect to call us instead of your local Radio Shack Computer Center Customer Service Representative.

There are seven main groups of customer support personnel. They are:

1. Model I/III Business Software
2. Model II/16 Business Software
3. Languages and Compilers
4. Color and Pocket Computers (all questions)
5. Hardware and Communications (all questions)
6. Educational Software
7. Games, Books, and New Product Information

MODEL I/III BUSINESS GROUP

The Model I/III business group, as its name implies, handles questions about the business software for the TRS-80 Models I and III. What constitutes business software? For example, is VisiCalc a business package? Yes, VisiCalc is considered a business package, falling into the category of planning and forecasting software. The Model I/III business software group answers questions on the accounting software, the word processing software, real estate and financial packages, the mailing list packages, the data base management software, the planning and forecasting software and the new manager series. Basically the Model I/III business group handles all Model I/III software except the games, compilers, and communications software.

The Model I/III business group can be reached at 817-870-2041 (Effective June 1, 1982).

MODEL II/16 BUSINESS GROUP

Virtually all of the software written for the TRS-80 Model II and Model 16 falls into the category of business software except the language compilers, editor/assemblers, and communications software. As with the Model I/III group the Model II/16 group answers questions on all of the accounting software including General Ledger, Accounts Receivable, Accounts Payable, Payroll, the Inventory Management System, and Business Mailing List II. There are different versions of our accounting software that accommodate different user needs. All of these different packages are supported by the Model II/16 business group. The data base management systems such as Profile II and Versafile (26-4510) are also supported by this group. VisiCalc, a fine planning and forecasting tool, also falls into the category of business software. The Model II/16 group can also answer your questions on Scripsit, our word processing software, and its companion, the Scripsit Dictionary. The Model II/16 group also supports the medical and legal professional software as well as the job costing package.

The Model II/16 group can be reached at 817-870-2042 (Effective June 1, 1982).

LANGUAGES AND COMPILERS GROUP

The language and compilers group assists with questions about the operation of the BASIC interpreters, BASIC compilers, COBOL compilers, FORTRAN compilers, and the editor/assembler packages for all of our computer models. This group also handles various utilities such as DEBUG, as well as all of the TRSDOS operating systems. The language and compiler group cannot, however, write software routines for you, or help you debug your programs. These representatives can answer your questions as to how specific commands of a particular language function, or the specific steps required in compiling a COBOL program. Of course, before using a particular language compiler you must know how to program in that particular language. We cannot teach you FORTRAN, COBOL, or BASIC over the telephone.

The language and compiler group can be reached at 817-870-2044 (Effective June 1, 1982).

COLOR AND POCKET COMPUTER GROUP

The color and pocket computer group answers questions not only about the software for each computer, but questions about the operation of the computers as well. Much of the software for the color computer is in ROM paks

and is virtually foolproof. Still there are many facets of the color computer with which our representatives can help. The new mini disk system and expanded memory capacity add a whole new dimension to the color computer.

The pocket computers, the PC I and the new PC II, have their own unique software, mostly programs designed to be used on the move. If you have questions about the PC I, the PC II, or any of the pocket computer software, stop, and give the color and pocket computer group a call.

The color and pocket computer group can be reached at 817-870-2150 (Effective June 1, 1982).

HARDWARE AND COMMUNICATIONS GROUP

The hardware and communications group can help you in many different ways. Their basic function is to provide technical assistance to our more technically oriented customers. All of these representatives have attended our extensive Radio Shack Certified Technician Training School. This month long training course covers everything from turning on a computer to computer logic theory. If you feel you are having problems with your computer system that do not appear to be software related, discussing the symptoms with the hardware and communications group may confirm that service is needed. However, keep in mind that a diagnosis over the telephone can only be a recommendation that service may be needed, and the Radio Shack service technician should be consulted for a complete inspection of the equipment.

The hardware and communications group can also help to determine the compatibility or incompatibility of interfacing different pieces of computer equipment, such as the connection of various serial printers with the various TRS-80 models. We may not be able to advise you pin for pin what configuration you need, but we can tell you what our computer expects to see on a particular output bus.

In addition to hardware related questions this group also handles questions on the communications software. This includes not only the communications software on the Model II's TRSDOS 2.0a but the communications software for the Models I and III. Questions about the Binary Synchronous packages — the IBM 3270 emulator and the IBM 3280 compatible remote terminal packages — are also answered by this group.

Of course, along with the communications software this group can also help with the connection of modems to your TRS-80 system.

As our computer systems become more sophisticated, hardware and communications support will become increasingly more important. With the introduction of our recently announced DT-1 Data Terminal and Arcnet (when it becomes available) a very powerful, complex system can be built. The hardware and communications group will assist you with these new products.

The hardware and communications group can be reached at 817-870-2571 (Effective June 1, 1982).

EDUCATIONAL SOFTWARE

The expanding educational software line has added a new group to Computer Customer Service. This group is staffed by educators who can help you implement a computer learning program. Our courseware can take a student from kindergarten through eighth grade with reading and math packages for various class levels. Contact the educational software group for questions on the new MicroPILOT and Author I packages. These two packages are designed to allow the educator to design his/her own courseware for a computer assisted instruction course. Our educational software group can help you apply these packages to your classroom. We realize the importance of the educational use of computers and the educational software group is dedicated to helping you get the most out of your TRS-80 classroom.

The educational software group can be reached at 817-390-3302 (Effective immediately).

Games, Books, and New Product Information

There are almost twenty entertaining games designed for the TRS-80 Models I and III that can stump even the most ardent gamesman/gameswoman. The games group can help you get out of a maze or past a stubborn ghost. The group may not give you a complete solution to a game, but they will give you hints on how you can proceed past a difficult spot. You wouldn't want them to take all of the fun away from you.

Books provide our customers with additional information in the use of our computers, what other people are using them for, and programs that can be used for fun, learning, or business. You may have entered a program from one of our books and it does not appear to be functioning correctly. The games and book group will help you determine where the problem lies. This group can also help clarify points made in a book and help you better understand your TRS-80.

New product announcements always bring a torrent of calls to computer customer service. The games, books, and new products group gets the information first, to give our interested customers the most accurate information available at the time a new product is released. If your local Radio Shack Computer Center or Radio Shack Computer Department has not received information about a new product release, the new product information group is here to fill in the gap.

The games, books, and new product information group can be reached at 817-870-2271 (Effective June 1, 1982).

CONCLUSION

The individuals that make up the support groups at Computer Customer Service have been trained to give you expert advice each time you call. However, no two people learn at the same rate and one representative will be more knowledgeable than another. Consequently, when you call there may be an occasion when your representative must confer with his/her fellow group members to confirm an answer to your questions or to assist with your problem. We will try to keep this to a minimum.

Remember, if you find yourself in a situation that cannot seemingly be solved by your local Radio Shack store, Computer Department, or Computer Center, consider a call to Computer Customer Service. Take a few minutes to gather some pertinent information to help our representatives and keep the length of your call to a minimum. Here is the information you should have before placing your call.

1. Your name
2. Your telephone number

3. What TRS-80 computer system you are using and the number of disk drives in the system, if applicable.
4. The stock number and name of the software package you are using.
5. The version number of the software.
6. What error codes the software is generating.
7. How the error occurred and the function you were performing when it occurred.
8. Information about any patches or program corrections you may have made.

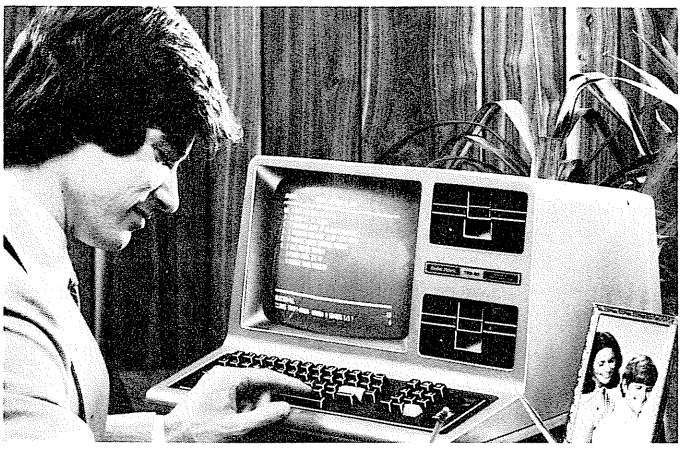
We are here to help you make your computer as enjoyable to use and as productive as you expect it to be. Our aim is to give you accurate, prompt, and courteous service in the tradition of Radio Shack.

Effective June 1, 1982 Computer Customer Services Address and Phone Number

8AM to 5PM Central Time

Computer Customer Services
400 Atrium, One Tandy Center
Fort Worth, Texas 76102

Model I/III Business Group	817-870-2041
Model II/16 Business Group	817-870-2042
Languages and Compilers	817-870-2044
Color and Pocket Computer Group	817-870-2150
Hardware and Communications Group	817-870-2571
Educational Software	817-390-3302
Games, Books, and New Products	817-870-2271



Ultra Precision Multiplication

Golden Richard, III
5815 Annunciation
New Orleans, LA 70118

This program offers a solution to the problem of obtaining exact answers to multiplication problems that use very large numbers. The "A" array contains the solution to the problem. The only limit on the length of the numbers to be multiplied is the max. length of a string, which is 255 characters. I am working on routines to handle addition, subtraction, and division.

```
1Ø 'ULTRA PRECISION ROUTINE
2Ø 'BY GOLDEN RICHARD III
3Ø '5K MEMORY REQUIRED
4Ø CLEAR 1ØØØ
     : DIM A(6ØØ
5Ø CLS
6Ø INPUT"FIRST NUMBER"; A$
7Ø INPUT"SECOND NUMBER"; B$
8Ø FOR T=LEN(A$) TO 1 STEP -1
     : IF MID$(A$,T,1)="." THEN D1=LEN(A$)-T ELSE
     NEXT
9Ø FOR T=LEN(B$) TO 1 STEP -1
     : IF MID$(B$,T,1)="." THEN D2=LEN(B$)-T ELSE
     NEXT
1ØØ PRINT D1,D2
11Ø FOR T=1 TO LEN(A$)
     : H$=MID$(A$,T,1)
     : IF H$<>"." THEN H1$=H1$+H$
     : NEXT ELSE NEXT
12Ø A$=H1$
13Ø FOR T=1 TO LEN(B$)
     : H2$=MID$(B$,T,1)
     : IF H2$<>"." THEN H3$=H3$+H2$
     : NEXT ELSE NEXT
14Ø B$=H3$
15Ø FOR P=LEN(B$) TO 1 STEP -1
16Ø OS=LEN(B$)-P
17Ø FOR T=LEN(A$) TO 1 STEP -1
18Ø T1=LEN(A$)+1-T
19Ø A(T1+OS)=A(T1+OS)+VAL(MID$(B$,P,1))*VAL(MID$
     (A$,T,1))
2ØØ NEXT
21Ø NEXT
22Ø DD=T1+OS
23Ø 'SORT
24Ø DP=DP+1
25Ø IF A(DP)>9 THEN A(DP+1)=A(DP+1)+INT(A(DP)/1Ø)
     : A(DP)=(A(DP)/1Ø-INT(A(DP)/1Ø))*1Ø
26Ø A(DP)=INT(A(DP)+.5)
27Ø IF A(DP+1)<>Ø OR DP<DD THEN 24Ø
28Ø PRINT"ANSWER: "
29Ø PRINT STRING$(64,"-");
3ØØ PRINT
31Ø FOR T=DP TO 1 STEP -1
32Ø IF T=(D1+D2) THEN PRINT".";
33Ø PRINT MID$(STR$(A(T)),2,1);
34Ø NEXT T
35Ø PRINT
     : PRINT STRING$(64,"-")
36Ø END
```

Languages and More

CompuServe Offers a Wide Variety of Programming Languages

Editor's Note: The CompuServe Information Service is one of the largest information and entertainment services available to owners of personal computers and computer terminals. With each issue of the TRS-80 Microcomputer NEWS, various features of CompuServe will be discussed. The CompuServe Information Service is sold at Radio Shack stores nationwide and in Canada.

In addition to the many consumer and business/financial services, CompuServe offers a variety of programming languages tailored toward the computer hobbyist.

Most of the languages have on-line documentation or printed documentation which can be ordered through the documentation ordering section of our Feedback feature.

CompuServe offers the following languages:

AID

Algebraic Interpretive Dialogue language. For information on obtaining additional documentation, enter: R FEEDBK

APL

A Programming Language. An extended version of APL-SV. For documentation enter: TYP SYS:APL.DOC

BASIC

Beginner's All-purpose Symbolic Instruction Code is a greatly enhanced version of Dartmouth BASIC, with string manipulation features, double precision, a linking capability, expanded looping and transfer capabilities, file-to-file sorting, a debug mode, and much more. For information on obtaining additional documentation, enter: R FEEDBK

BLIS10

PDP-10 expression language. For information on obtaining additional documentation, enter: R FEEDBK

CROSS ASSEMBLERS

Cross-assemblers are available which generate listing and object files for 8080, 6502, CDP1802, 6500, 6800, 6809, Z-80, SC/MP and 6800 microprocessors. For instructions on the use of these cross-assemblers, enter: TYP SYS.CROSS.TXT

FOCAL

Fast on-line calculation language. For documentation enter: TYP SYS. FOCL10.DOC

MACRO

PDP-10 symbolic assembler language. For information on obtaining additional documentation, enter: R FEEDBK

PASCAL

This version of PASCAL and its accompanying documentation has been obtained from the Digital Equipment Computer User Society (DECUS) and as such, CompuServe cannot assume responsibility for any errors which may exist. To Order PASCAL documentation, enter: R FEEDBK

Or to print the documentation at your location, enter the terminal command TER FORM REAL and then enter: TYP SYS:PASCAL.MAN. The document will take 47 minutes (time may vary) to print at 300 BAUD (30 CPS). The cost in connect time will be about $3.99.

For additional assistance in using PASCAL, enter: TYP SYS:PASCAL.MEM

SNOBOL

Text manipulation language. For documentation enter: TYP SYS:SNOBOL.DOC or to obtain additional documentation, enter: R FEEDBK

XF4 (FORTRAN)

XF4 is an extended FORTRAN language set developed by CompuServe, with the following important features: dynamic arrays can be set during execution; functions and subroutines are available for processing string variables; random access I/O provides for effective management of large amounts of data; structured programming techniques, including IF statement blocks and DO WHILE, can be used; conversational debugging can be performed during execution; and a core image can be saved, eliminating later recompilation.

For information on obtaining additional documentation, enter: R FEEDBK

DECWARS GAME AVAILABLE AS MULTIPLAYER ACTIVITY

A new multi-player game is now available called Decwars. Decwars is a real-time space battle game that up to 10 people can play. You can choose whether to be the captain of a Federation ship or an Empire (Klingon) ship, and you can attack enemy ships or bases or capture planets. The Romulans are also a part of this game. By using the communications channels provided, you can join with other players to form teams to more quickly destroy enemy ships. The weapons should be familiar to you; the ships are armed with phasers and photon torpedoes and can manipulate their engines through impulse or warp drive. Documentation is available on-line using the HELP * command.

CUBE PUZZLE SOLVER

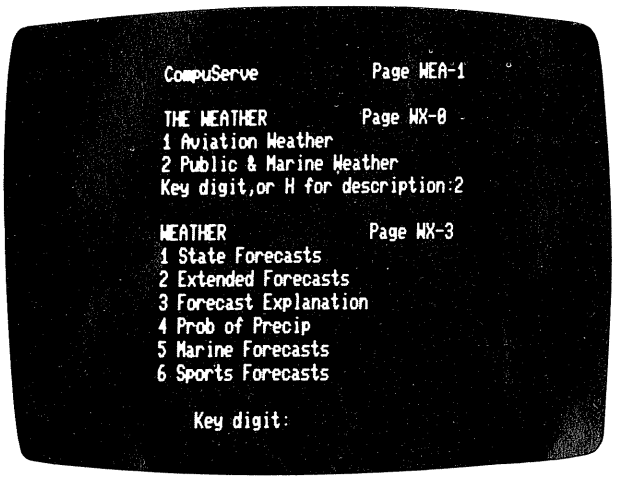
Now available from the games menu page is a program that will tell you how to solve the popular six-color cube puzzle. If you use this program, it will tell you how to manipulate the cube so that each of its sides will be a solid color.

So if you are baffled by the cube, use the Cube Solver and find the solution.

CONTINUOUSLY UPDATED WEATHER INFORMATION

Available now on the Special Services menu is "Aviation and General Weather." Aviation weather is a service for pilots containing information received from the NOAA high-speed wire. This is the same data used in International Flight Service locations. Much of the aviation weather data is transmitted in abbreviated form, but help is available if you don't know what certain abbreviations mean.

The general weather service consists of public weather and marine weather reports. These reports are written in plain English. Help is also available in the public and marine areas if you need help in retrieving the information.



BALANCE YOUR CHECKBOOK AND CALCULATE YOUR NEXT RAISE

CompuServe's new Checkbook Balancing and Raise Calculation programs are now available in the Home Management Programs area.

The Checkbook Balancing program simulates your bank checking account by providing standard deposit and withdrawal features including such transaction types as bank machine withdrawals and deposits, pay-by-phone, overdraft advance, and automatic deposit. Attractive features include overdraft protection, minimum balance alerts, transaction summary, service charge recording, and end-of-month balancing. Lists of outstanding checks and deposits can be generated at will. Checks can be cancelled or cleared, and the account balance adjusted for convenient and accurate record keeping.

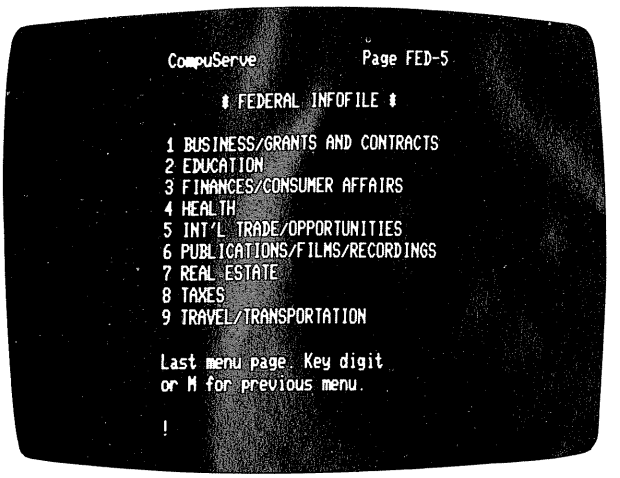
The Raise Calculation program figures the increase in one's current salary in terms of annual, monthly, period, weekly, daily and hourly income.

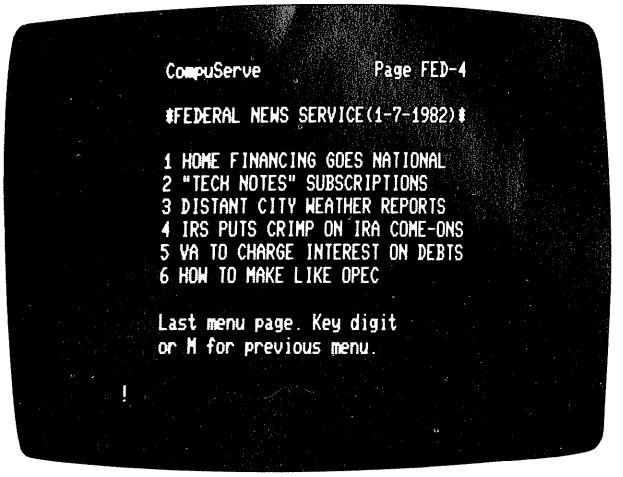
Both of these practical fiscal management programs can enhance the efficiency and effectiveness of personal financial planning.

THE FUTURE FILE AND FEDERAL REPORTS

CompuServe has also announced the addition of two information providers; "Federal Reports" and "The Future File." Both are updated with new material on a bi-weekly basis.

In the Federal Reports section, subscribers can access articles and reports on recent legislation and read about its impact on the nation's people. Particular emphasis is placed on activities that effect the financial and investment climate of the United States.

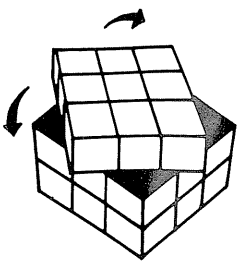




Future File includes articles and interviews with futuristic authorities on such subjects as business, politics, and the military that shed light on what might be expected from these arenas in the years to come. New technology, today's decisions, and their potential impact on tomorrow's society are discussed.

* * *

Questions and comments about the CompuServe Information Service can be sent to Richard A. Baker, editorial director, CompuServe Information Service, P.O. Box 20212, Columbus, Ohio 43220 or through Feedback, CompuServe User Information.



Disclosure II

Expanded Corporate Information Now Available From Dow Jones

Editor's Note: With over 32,000 subscribers, Dow Jones News/Retrieval® is the leading provider of online business and financial information. We will keep you up-to-date on new software, new data bases and new prices in upcoming issues of TRS-80 Microcomputer News. Dow Jones News/Retrieval is sold at Radio Shack stores nationwide and in Canada.

DISCLOSURE ONLINE became DISCLOSURE II early in 1982. This new enhancement to Dow Jones News/Retrieval, provided by Disclosure Inc., allows for greater selectivity and provides much more extensive information from wide-ranging composite data files with the Securities and Exchange Commission (SEC). Business and financial data is available on approximately 6,000 public companies extracted from 10-K, 10-Q, and 8-K reports as well as proxy statements and registration statements for new public companies.

DETAILED DATA ON OVER 6,000 COMPANIES

Each of 6,000 companies will have "profile" information including exact name, address, place of incorporation, Standard Industrial Classification (SIC) codes, the exchange on which its stock is traded, ticker symbol, and a textual description of business.

The financial data maintained for each company includes the number of shares of common stock outstanding, number of employees, a listing of subsidiaries, names of officers and directors, as well as asset, liability, and income statement items for the previous three years.

EXPANDED FINANCIALS

DISCLOSURE II contains numerous enhancements which reflect the new integrated disclosure program adopted by the SEC in September, 1980, including significantly more items of financial data, more "people" information and a textual "Management Discussion" section.

There are more than 40 balance sheet items for each of two fiscal years, more than 15 income statement items for each of three fiscal years; quarterly income statements covering the current fiscal year's quarterly data from the 10-Q reports; business segment data showing sales and operating income for up to 10 lines of business; and a five year summary including consolidated sales, net income and earnings per share.

The additional profile information includes a cross-reference field that contains a listing of new or previous company names, headquarters telephone number, and Fortune 1,000 ranking, shares held by officers and directors, and any new auditor with the date of change. More data is also included on officers and directors of companies, including their ages and annual remuneration. Corporate events, such as executive changes and mergers and acquisitions, are also included.

NEW MANAGEMENT DISCUSSION REPORT AN EXCELLENT ADDITION

Under the SEC's new disclosure policies, an important addition to Form 10-K is the management discussion item. The full text will be available online and focuses attention on operational results and the financial condition of the company. This section includes a discussion of cash flow, liquidity, future plans, the effects of inflation, the need for capital and how this will be satisfied. This information could prove to be very valuable to those users seeking a more complete picture on the company's basic structure and future outlook.

THE COST

The usage for accessing the DISCLOSURE II data base is the same as the News data base: $1.00 per minute during prime hours and .50 cents per minute during non-prime hours. In addition to usage, there is also a special access fee per company. This is how it works: When a user selects one of the items on the menu for any one company, he will automatically be charged an additional $4.00 access fee during prime hours, $2.00 during leisure hours. As long as he stays with the same company, he can retrieve as many reports or documents as he wishes without paying an additional access charge; however, upon switching to another company, another access fee will be added. It is, therefore, best to stay with one company until you have retrieved *all* of the information you want on that company before going on to another company's file.

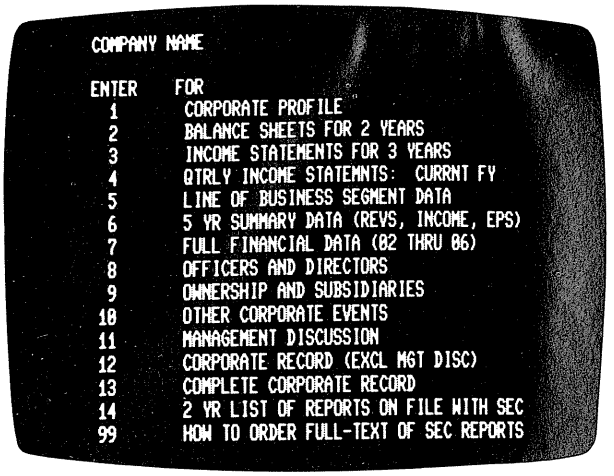
HOW TO ACCESS DISCLOSURE II

After you enter your regular password, News/Retrieval will send the message "ENTRY QUERY."

Step 1. Type ⟨/⟩⟨/⟩⟨**D**⟩⟨**S**⟩⟨**C**⟩⟨**L**⟩⟨**O**⟩and the ⟨**ENTER**⟩ key. You will then see a welcome and the following message:

TO CONTINUE, ENTER COMPANY STOCK SYMBOL AND PRESS RETURN

Step 2. Follow the instructions as they appear on your screen or print-out. Upon entering a company stock symbol and the ⟨**ENTER**⟩ key, you will see the following menu:



Step 3. Select the information you want by pressing the appropriate code number followed by the ⟨**ENTER**⟩ key.

Note: In this data base, there is a special access fee charged on a per company basis:
$4.00/prime time
$2.00/non-prime time

You may retrieve as many of the SEC reports as desired by paying this one access fee . . . as long as you stay with one company. It is advisable, therefore, to stay with one company until all data is retrieved.

Color Computer Notes

Richard G. Neill
5170 Bramblewood Dr.
Acworth, GA 30101

Just a note to say that I like Microcomputer News. I have the Color Computer and never cease to be amazed by the power and functions that I have available. I look forward to purchasing a disk system for it, but for now am 'just' restricted to the cassette. Of course we do have the file system, 1500 baud, and much more. Perhaps someone could come up with a merge for cassette. Sure would speed up program development.

Oh, it seems that the SKIPF statement can be used to check for a good CSAVE much in the same way Model I/III users use CLOAD?. It won't check it, but it will tell if the dump will reload (and won't alter memory).

Try this, after printing to the screen, stop program execution with the INKEY$ function (not INPUT or LINEINPUT) then POKE 65314,8. It creates red letters on a pink screen — something that I have heard 'can not be done.'

Keep up the Color computer articles (keeps me from having to translate Mod I/III). Congrats on William Barden's series.

Verifying Programs and Data Files

Detroit Computer Support Group

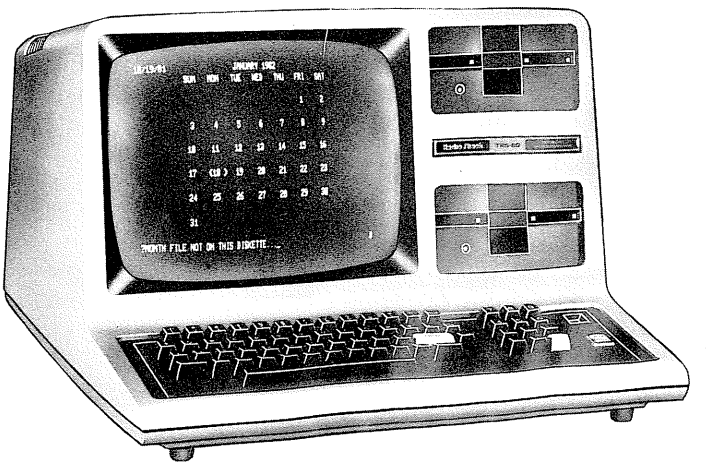
It is often nice to be able to compare two program or data file listings to verify that they are the same. Here is a short program to accomplish that on Models I, II and III. This program assumes a Logical Record Length of 256, and two disk drives.

```
1Ø CLEAR 2ØØØ
2Ø INPUT "ENTER FILE NAME"; F$
3Ø PRINT "PRESS ENTER WHEN 'GOOD' DISK IS IN
     DRIVE Ø"
4Ø PRINT "AND 'SUSPECT' DISK IS IN DRIVE 1"
5Ø LINEINPUT A$
6Ø OPEN "R", 1, F$
7Ø OPEN "R", 2, F$+":1"
8Ø FIELD 1, 128 AS A$, 128 AS B$
9Ø FIELD 2, 128 AS A1$, 128 AS B1$
1ØØ FOR X=1 TO LOF(1)
11Ø GET 1, X
12Ø GET 2, X
13Ø PRINT X,
14Ø IF A$<>A1$ OR B$<>B1$ THEN PRINT "BAD SECTOR
     FOUND" ELSE PRINT
15Ø NEXT X
16Ø CLOSE
17Ø END
```

Notes: In general, this program should be used with ASCII program and data files. Bad Sector here simply means that the information in the two files is different, and says nothing about the quality of the diskette itself.

If your logical record lengths do not equal 256, make the following changes:

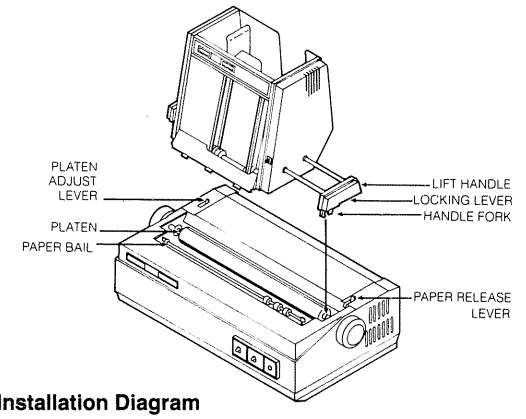
```
25 INPUT "ENTER LOGICAL RECORD LENGTH (LRL)", L
6Ø OPEN "R", 1, F$, L
7Ø OPEN "R", 2, F$+":1", L
73 IF L=1 THEN FIELD 1, 1 AS A$
     : FIELD 2, 1 AS A1$
     : B$=""
     : B1$=B$
     : GOTO 1ØØ
75 IF L/2=INT(L/2) THEN S=L/2
     : S1=S ELSE S=INT(L/2)
     : S1=S+1
8Ø FIELD 1, S AS A$, S1 AS B$
9Ø FIELD 2, S AS A1$, S1 AS B1$
```



Envelope Feeder for Daisy Wheel II

Sometimes, after using a new product for a while, you wonder how you ever got along without it. The TRS-80 Envelope Feeder, which operates on the Daisy Wheel II printer, is *that kind of product*. Any business, non-profit organization, politician, professional or individual with a need to address multiple envelopes will find the Envelope Feeder to be a time and money saver.

The Envelope Feeder is capable of automatically feeding up to 200 good grade (20 lb. to 24 lb. paper weight) personal or legal size envelopes. Full-envelope printing lets you print on any horizontal portion of the envelope, and the feeder includes automatic thickness adjustment, so there's no worry with adjusting the Envelope Feeder for the varying paper weights (thickness).



Feeder Installation Diagram

The Envelope Feeder is easily installed and when printing is finished, it is easily removed. There is a minor hardware modification that needs to be made to the Daisy Wheel II before the Envelope Feeder can be operated. If you already operate a Sheet Feeder (26-1448) with the Daisy Wheel II, just plug the Envelope Feeder into the printer.

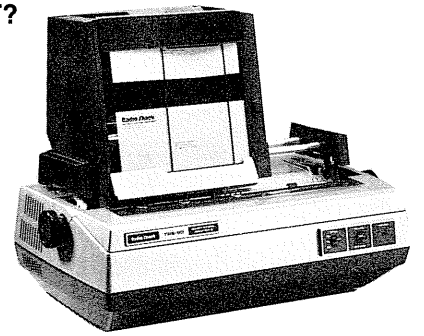
PRODUCTION INCREASED 600%

There are many advantages to using the Envelope Feeder. Manually printing envelopes on a typewriter or even feeding them one at a time into the printer is very time consuming. One study has shown that in the time required to print 200 envelopes by conventional methods, 1,200 envelopes could be printed using the Envelope Feeder. For the mathematically disinclined that's a 600% increase in output.

Once the software and Envelope Feeder are set up, 200 envelopes can be printed without operator intervention. Since the envelopes are fed automatically, the operator is freed from the tedious and time consuming job of handling each envelope as it comes through the printer (or typewriter).

When you consider the time saved (Time is money!) or the cost of expensive continuous form envelopes, the Envelope Feeder looks unbeatable.

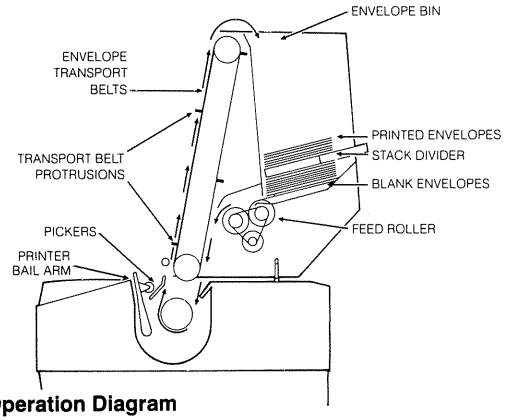
HOW BIG IS IT?



Some specifications of the Envelope Feeder are:

Length	21½"
Width	9" (front to back)
Height	11¾"
Weight	9½ pounds

OPERATION OF THE ENVELOPE FEEDER



Feeder Operation Diagram

The Feeder Operation diagram depicts the operation of the envelope feeder.

1. By attaching the Envelope Feeder to the printer's platen, the Envelope Transport Belts are synchronized to the printer platen through the Feeder Idler Gear.
2. Blank envelopes are stacked in the Envelope Bin and the Stack Divider is placed on top.
3. As the platen turns, the positions of the Envelope Transport Belt protrusions are sensed by the Envelope Feeder.
4. At the proper time, a motor drives the Feeder Rollers which load a blank envelope from the bottom of the stack and positions the envelope in the holding area above the platen.
5. As the Feeder Transport Belts advance, their protrusions align the envelopes horizontally and precisely space and guide them into the platen pickup area.
6. The envelopes are under Printer control while being typed and the platen ejects them into the Envelope Wire Guides otherwise known as Output Pickers.

7. The printed envelopes are picked up by the Transport Belts and directed through the upper deflector which deflects them into the Envelope Bin on top of the Stack Divider.

The description of the feeder's operation might leave you with the impression that Rube Goldberg has struck again, but be assured that the Envelope Feeder does work, and it is fast.

A HIGH-QUALITY END PRODUCT

The major advantage of the Envelope Feeder is that the end product is a high quality addressed envelope. No cheap-looking labels or window envelopes which shout ad or bill. The use of the printer is optimized and maybe best of all, the Envelope Feeder is easy to use. The Envelope Feeder is stock number 26-1260 and has a suggested retail price of $1195.00 (Installation extra, if required).

Plug 'n Power™ Program Changes

After careful analysis, the BASIC programs supplied with the 26-1182 Plug 'n Power Controller have been corrected to insure reliable operation with Model I and III programs. These changes are needed because some of the X-10 modules have greater variations in frequency response than the original programs allow for.

For Model I users, the following changes may be made to the Model I programs listed in the manual on pages 25 through 28. These changes may give you better operation:

In line 160 change

```
C=C*2+1 to C=C*2+2
```

In line 340 change

```
DATA 251,6,3Ø,... to DATA 251,6,45,...
```

For Model III users, here is a new control program, along with needed modifications for various computer sizes.

```
8Ø REM CNTRL16 16K MODEL III DISK BASIC (MEMORY
     SIZE = 32511)
1ØØ DIM A(22)
     : FOR I=1 TO 22
     : READ A(I)
     : NEXT
11Ø FOR X=32512 TO 32614
     : READ G
     : POKE X, G
     : NEXT
12Ø CLS
     : INPUT "INPUT HOUSE CODE A-P"; Z$
     : Z=ASC(Z$)-64
     : POKE 32613, A(Z)
13Ø PRINT " "
     : INPUT "INPUT UNIT CODE OF 1-16"; D
14Ø PRINT " "
     : INPUT "INPUT COMMAND CODE OF ON=1 OFF=2
     CLR=3 ALL=4 BR=5 DIM=6"; E
15Ø E=E+16
     : C=3
     : IF E<21 GOTO 17Ø
16Ø PRINT " "
     : INPUT "INPUT NUMBER OF STEPS 1-1Ø"; C
     : C=C*2+2
17Ø POKE 32614, A(D)
     : POKE 32615, 3
     : GOSUB 19Ø
18Ø POKE 32614, A(E)
     : POKE 32615, C
     : GOSUB 19Ø
     : GOTO 12Ø
19Ø DEFUSR1=32512
     : X=USR1(Ø)
     : RETURN
2ØØ REM:****************************************
21Ø REM:IF USING MODEL III BASIC
22Ø REM:CHANGE STATEMENT 19Ø TO
23Ø REM:POKE 16526, Ø: POKE 16527, 127: X=USR(Ø):
     RETURN
24Ø REM:****************************************
25Ø DATA 96, 224, 32, 16Ø, 16, 144, 8Ø, 2Ø8, 112,
     24Ø, 48, 176, Ø, 128
26Ø DATA 64, 192, 4Ø, 56, 8, 24, 88, 72
27Ø DATA 243, 245, 197, 213, 58, 1Ø3, 127, 79,
     2Ø5, 62, 127, 2Ø5, 62, 127
29Ø DATA 2Ø5, 62, 127, 2Ø5, 57, 127, 58, 1Ø1,
     127, 6, 4, 2Ø5, 44, 127, 58
3ØØ DATA 1Ø2, 127, 6, 5, 2Ø5, 44, 127, 13, 32,
     225, 2Ø9, 193, 241, 251, 2Ø1
32Ø DATA 7, 212, 57, 127, 2Ø5, 62, 127, 22Ø, 57,
     127, 16, 244, 2Ø1
33Ø DATA 17, 2, 2, 24, 3, 17, 2, 1, 245, 197, 14,
     255, 237, 12Ø, 23, 48
34Ø DATA 251, 6, 65, 16, 254, 62, 3, 237, 81, 6,
     154, 16, 254, 237, 89
35Ø DATA 61, 32, 3, 193, 241, 2Ø1, 6, 21Ø, Ø, 16,
     253, 24, 235, Ø, Ø
```

For 32K make the following changes:

```
8Ø REM CNTRL32 32K MODEL III DISK BASIC (MEMORY
     SIZE = 48895)
11Ø FOR X=-1664Ø TO -16538...
12Ø ...: POKE -16539, A(Z)
17Ø POKE -16538, A(D)
     : POKE -16537, 3
     : GOSUB 19Ø
18Ø POKE -16538, A(E)
     : POKE -16537, C
     : GOSUB 19Ø
     : GOTO 12Ø
19Ø DEFUSR1=-1664Ø
     : X=USR1(Ø)
     : RETURN
23Ø REM:POKE 16526, Ø: POKE 16527, 191:...
```

In DATA lines 270, 290, 300 and 320 change all occurrences of 127 to 191. There are a total of 12 127s which need to be changed in the DATA lines.

For 48K make the following changes:

```
8Ø REM CNTRL48 48K MODEL III DISK BASIC (MEMORY
     SIZE = 65279)
11Ø FOR X=-256 TO -154...
12Ø ...: POKE -155, A(Z)
17Ø POKE -154, A(D)
     : POKE -153, 3
     : GOSUB 19Ø
18Ø POKE -154, A(E)
     : POKE -153, C
     : GOSUB 19Ø
     : GOTO 12Ø
19Ø DEFUSR1=-256
     : X=USR1(Ø)
     : RETURN
23Ø REM:POKE 16526, Ø: POKE 16527, 255:...
```

In DATA lines 270, 290, 300 and 320 change all occurrences of 127 to 255. There are a total of 12 127s which need to be changed in the DATA lines.

Profile Plus

After an initial delay as an improved manual was written, Profile Plus for the Model II is on its way to production, and some of the many enhancements are the results of phone calls and letters from Profile II users. The most important point is that the data file structure from the original Model II Profile II was not changed.

If you are currently a user of Profile II, you can upgrade to Profile Plus with a minimum effort. After a "DO" file conversion is executed, the only other thing you must accomplish is to load each of the print and label formats and save them back out. This will immediately give you some of the new features such as the ability for an index and instant screen switching, but the majority of the new features will show up when you study the new field designators that are available. You can go back into your current screens and implement some of them, but adding any additional fields may require changing the field numbers in each of your reports screens to match.

Now, some of the features of Profile Plus:

USER MENUS — Allows users to create their own customized menus. All programs can receive the file name, format number and selection criteria from these User Menus.

INDEXING — Supports an INDEX file which allows high speed access to records on a PROFILE file. INDEXING also allows you to browse through a file as if it were in sorted order. INDEXING may be performed on any field within segment one.

ASSOCIATED FIELDS — Allows you to cluster multiple fields into search groups. When performing search operations, Profile Plus will scan all associated fields in each record for a match.

MATH PACKAGE — Supports addition, subtraction, multiplication, and division. The results of these mathematical operations are stored in fields of the record. Up to 16 equations of up to 63 characters are supported. Calculation results may be specified as being integer, two-place decimal or unrounded formats.

RECALCULATE MODE — Allows the mass recalculation of totals for selected records in the file.

SCREEN SWITCHING — Permits the use of multiple screens during a single execution of the program. The user may switch screens while a record is being displayed. All five possible screens will be available if they are defined.

ADVANCED FIELD TYPES — Implements 12 new field types/edits, including date fields, must-fill fields, and date-of-last-update fields.

DUPLICATE KEY — Uses the 'HOLD' key to act as a duplicate key during update or add operations. The field duplicated will come from the last record successfully updated.

TWO LINE REPORTS — Allows an extra line of data to be extracted from each record during reports.

SINGLE LINE TOTALS — Supports 2 single line totals on reports. An overflow area is provided if required.

SUPPRESSING REPORT LINES — Permits the suppression of title, heading, or detail lines. This allows the creation of summary reports.

EIGHT LINE LABELS — Supports labels of from one to eight lines of data in length.

PAUSE BETWEEN PAGES — Supports either single sheet or continuous form operations in both the report and label printing programs.

SELECTABLE RECORD LENGTHS — Allows you to specify the record length for segments 2 to 4 of the database. Each can be from 1 to 256 characters. This prevents wasted disk space and may be used to increase the maximum number of records on a multi-segment file to over five thousand.

ADVANCED SCREEN EDITING — Supports insert line and delete line during screen creation. Hardcopying of the defined screen is also supported.

ADDING FIELDS AND SEGMENTS — Supports the adding of new fields and segments to existing files.

UPPER/LOWER CASE SUPPORT — Ignores upper/lower case differences during sorting or searching operations.

MODIFIED BREAK KEY — Requires that the BREAK key be pressed a second time to confirm your intention. This prevents accidental exiting of PROFILE programs.

TYPE AHEAD — Enables type ahead in all areas except when first entering the update mode.

The catalog number for Profile Plus is 26-4515 and retail is $299.00, but if you are now a Profile II owner, you should order 26-4517 for only $120.00.

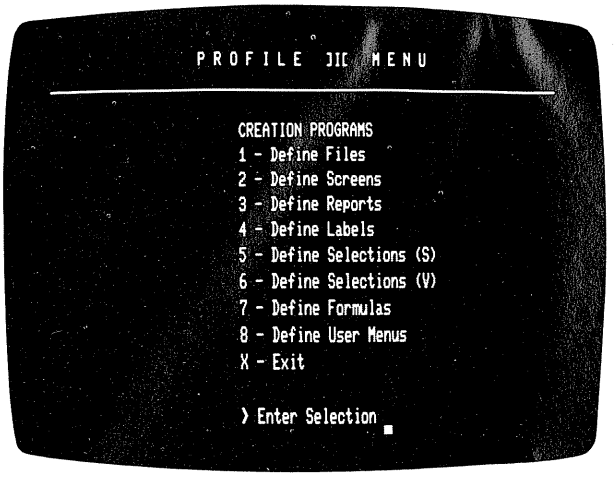
The upgrade package contains all of the new modules and several "merge" modules to upgrade your original Profile II disk to the Profile Plus level, and two "DO" files make the conversion very painless. Both packages contain the "NEW" manual that will start you at ground zero and carry you to the expert level of data handling with Profile.

By the way, the original Profile II (26-4512) will stay in the line, so if you don't feel you need all of the features of "Plus" you can still start with the basic package then, if you wish, add the plus features at a later date.

PROFILE III+

For those of you who want the power of Profile Plus, but not the storage capacity (or price) of the Model II, we are happy to announce PROFILE III+, which should be in your local Radio Shack Computer Center in the near future. It requires a Model III TRS-80 with 48K and at least one disk drive. Of course, you will also need a printer if you want to use the PRINT facilities.

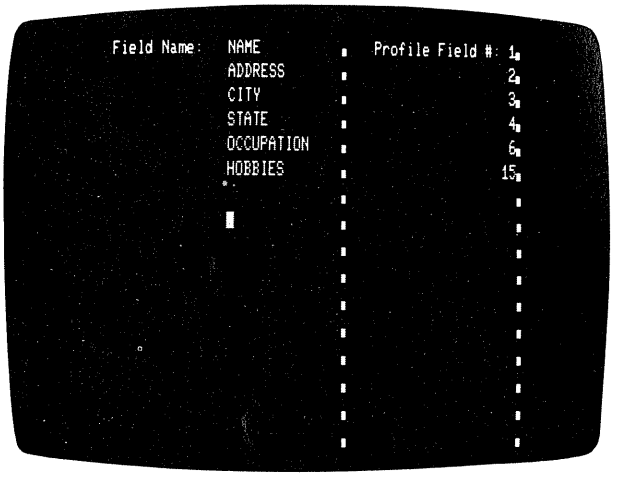
Like its Model II big brother Profile Plus, Profile III+ supports files with up to 99 fields per record, of which as



many as 36 can be used for sorts and searches.

Up to five different user-defined data entry screens are possible; their format is totally free, and can even use the Model III special characters! On the data entry screen, you can specify what type of information goes into each field, using any of these formats:

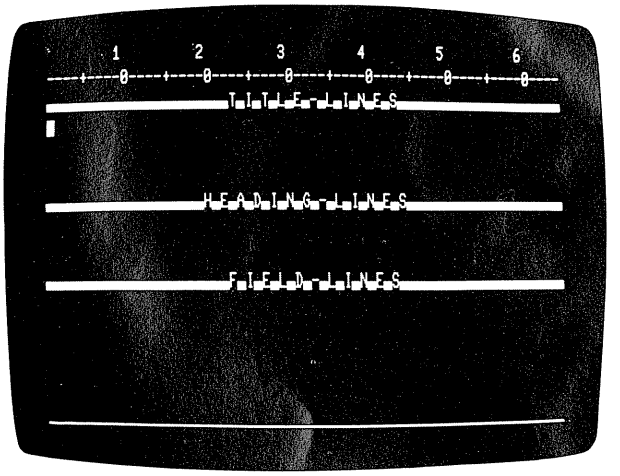
Alphanumeric
Numeric
Decimal (dollars and cents)
Protected
Date, MM/DD/YY
Date, YY/MM/DD
Alphanumeric Must Fill (Cannot be left blank)
Numeric Must Fill
Decimal Must Fill
MM/DD/YY Must Fill
Date of Last Update (MM/DD/YY)
Date of Last Update (YY/MM/DD)



By using more than one screen, you can have a field that can be examined by anyone, but only users with a password can alter the information. You can even have fields that are not visible to everyone. If your existing manual data entry system is not too large, you can duplicate it on the screen for maximum ease of data entry and operator training.

Profile III+ allows you to define up to five report formats and five mailing label type formats.

A complete math package lets you define up to sixteen math formulas. The results are stored in fields of the record. You also have the ability to change the math formulas for a file and to recalculate every field in the file. (This might be handy if, say, your prices were based on the price of gold and you wanted to adjust your price list.)



NEW FEATURES OF PROFILE III+

In addition to all the features of Profile Plus, several additional ones are available on the Model III. These include:

— The records themselves can be anywhere from one to 1020 characters in length (the Profile Plus limit is 853 characters).

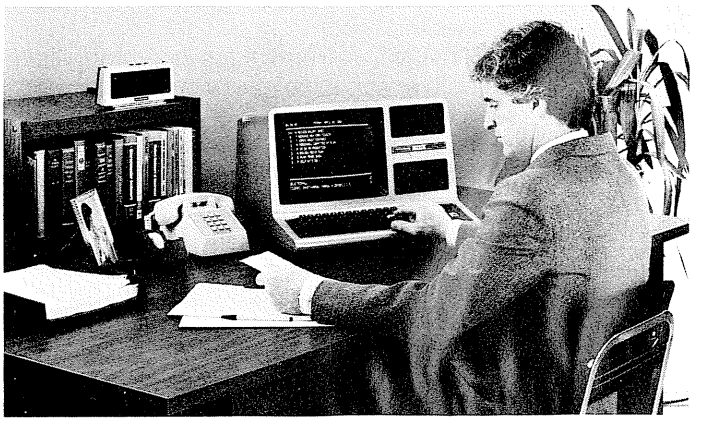
— The searchable portion, or KEY segment, can be from one to 255 characters long (Profile Plus KEY segments are always 85 characters long).

— Mass recalculate, hardcopy, purge, and delete functions are supported (Profile Plus supports mass recalculate).

— The powerful 16 field search ability found in the Profile Plus SELECTOR/EFC program has been extended to work with the Profile III+ report and label programs.

— Profile III+ can sort more records, faster, than Profile Plus.

Model III Profile III+ will be available as stock number 26-1592, for a suggested retail price of $199.00. Since Profile III+ is a totally new program for Model III, no upgrade is available for users of the current Model III Profile (26-1562).



DIVYING UP

Mr. Gilroy Has a Problem. But He Knows What to Do

Mr. Gilroy runs the marketing division for a manufacturing company which makes hand tools — lots of hand tools. Tools for carpenters. Tools for plumbers. Tools for mechanics. Tools for shadetree tinkerers.

Mr. Gilroy's problem was that he had to sell all those tools to the right people. The tools for mechanics had to be advertised mainly to auto shops — but also to top-quality hardware stores. The carpenters had to be reached through distributors they dealt with . . . and the distributors wanted cooperative advertising programs. Still other tools were sold to a major department store chain under another name. Mr. Gilroy had to apportion some money for that sales effort too.

To solve those problems, Mr. Gilroy turned to another kind of tool — a business person's tool. The VisiCalc program had caught Mr. Gilroy's eye at the local Radio Shack dealer, running on a TRS-80 Model III (it also runs on the TRS-80 Models I and II). He had invested in the program and used it extensively to ADD UP budgets and handle sales projections. Now he turned to his TRS-80 and the VisiCalc program to BREAK DOWN his budget into its constituent parts and reform the figures into the information he needed.

THE ESSENCE OF SAVING TIME

One of Mr. Gilroy's major worries was that the figures he started with would not be the figures he needed to end up with. In other words, he had always had to guess at what his total budget needed to be, but he might find that — once divided up — it would not allow enough funds for one or another of the lines he was managing. This meant that percentages might have to be changed, lines combined in different ways and the budget refigured.

Without the VisiCalc program, such refiguring would take him days — even had he been willing to undertake the task. With the program, he could make changes as he saw fit and as easily as using a typewriter . . . let the program do the complicated figuring. His "guesses" would become highly educated.

Here is how he approached his problem:

Total Budget This Year

↓

Product Line . . . weight (importance) $\times \frac{\text{Total Budget}}{100} =$ Budget Available per Line Item

Product Lines (organized by industry) = Total available per industry

Total per industry Divided by Promotional Means (advertising, direct sales, co-op etc.)

This shows Mr. Gilroy his total budget broken down three different ways:

1. Each product line is broken out and each item in the line is given a weighting factor within the total budget. The sum of the weighted amounts for a product line is the total amount budgeted for that product line.
2. The information is then broken down by industry. Within each industry an amount is budgeted for each of the product lines, with the sum of the product line amounts being the total amount budgeted for that industry.
3. Finally, Mr. Gilroy has the information broken down by the promotional method, how much of the total budget the method will get, and how many of the budgeted dollars for a particular industry will go to that promotional method.

Each element of a budget organized in the above manner is modular; factors "higher up" on the VisiCalc sheet can easily be changed, and still bear directly on numbers calculated farther down. For example, if the relative importance of the product mix changed, the "weight" of each product entry into the mix could easily be changed. In fact, were this simply a segment of a larger model, then the "weight" might have been transferred into the model from another model which had been designed to develop the market "weight" of each product.

All the factors which Mr. Gilroy believed to be necessary are accounted for: total budget, each product line, each line's importance, etc. The organization also gives him a means of tracking the budget per line item as well as the budget for a given market segment (plumbers, carpenters and others).

Finally, the division of the total budget is consolidated by promotional means. These figures can be transferred to other VisiCalc models. For example, Mr. Gilroy might find it interesting to match his advertising budget against the advertising of his top three competitors (as understood by monitoring their advertising in particular magazines). If he wished, and because his original VisiCalc model is modularly arranged, he could compare advertising efforts in a given industrial segment or across the board.

A simplified version of the model he built is reproduced below.

AN EXAMPLE YOU CAN TAKE APART

The first step in handling such budgetary divisions and recombinations is to set up the projected annual (or monthly) line item budget amount. Next the product mix is delineated. Here, the budget per product line item is the weight (as a percentage) times total budget divided by 100 — i.e., what percent of the total budget is the total line item worth?

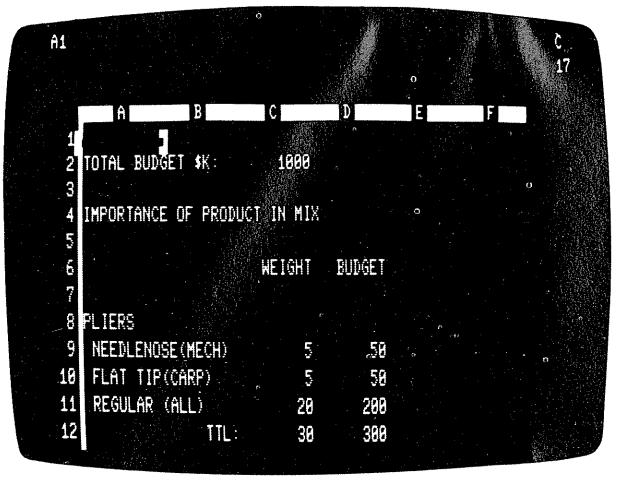


Photo 1

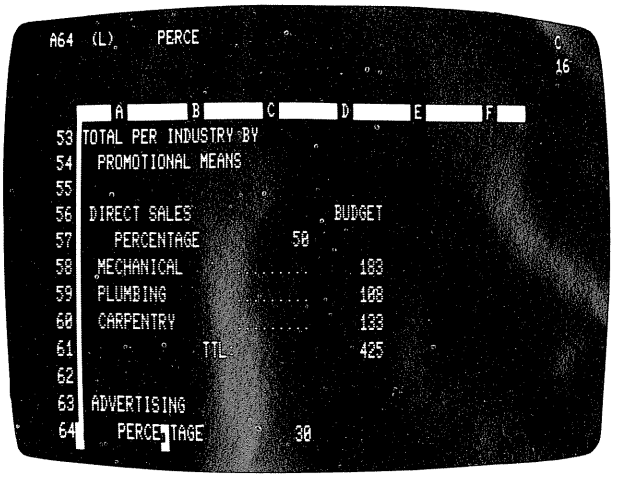


Photo 3

In a situation such as our example, where a company is selling into an identifiable number of discrete markets (even if it is a large number), weighting the product line in importance and dividing it by industry may make sense. In another situation, such as selling into a tiered distribution network, priorities might be ordered by importance of the channel.

Step two could be to divide the product lines (and their respective budgets) into the market segments for which they are bound. This converts the budget from division by product to division by market. Once the various product categories are divided into their target markets, the sums available to each of those product lines within the market may be consolidated.

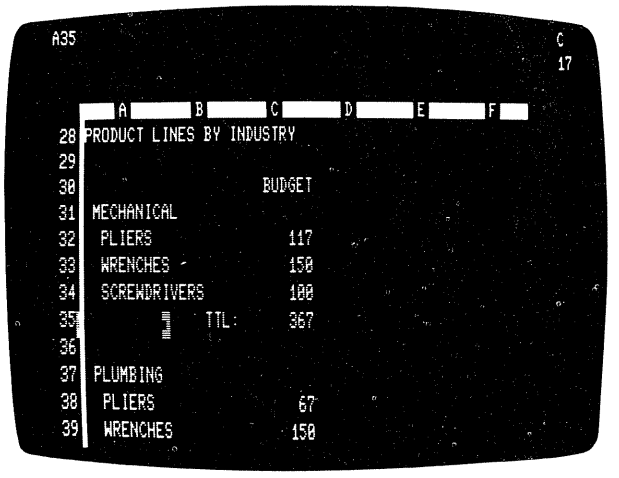


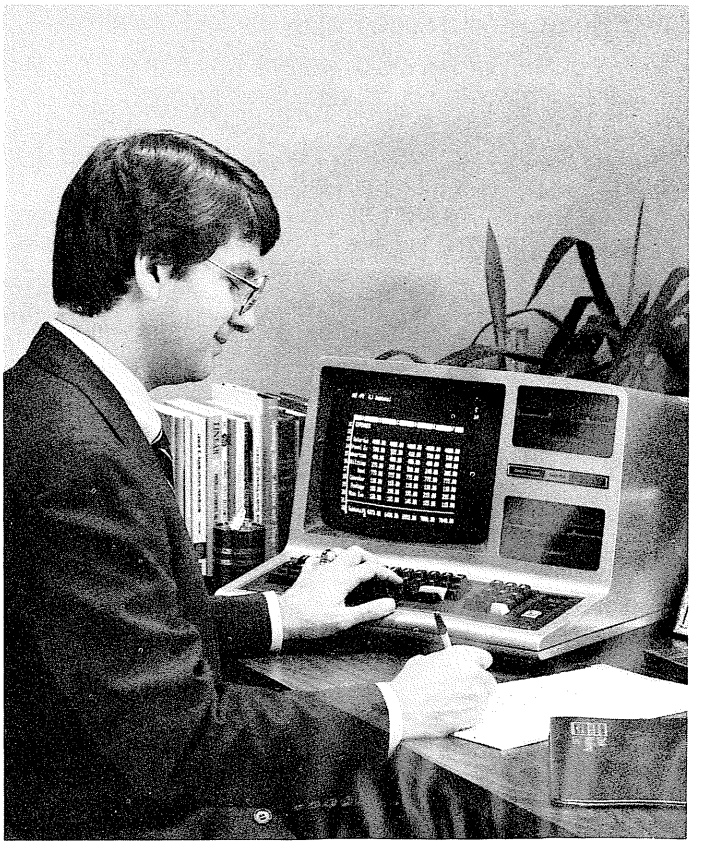
Photo 2

Sales Channels are next—and they too may vary with market (although we did not vary them here). The proportion of each industrial segment addressed by each distribution channel is calculated so that a budget figure may be arrived at for each channel.

It is important to note that this small model might have been calculated in the opposite direction: Mr. Gilroy could have begun by assuming a budget amount for his channels of distribution, then calculated how much of each channel was to go to each of his product lines, then divided that figure between the industries to obtain his total budget figure.

In Mr. Gilroy's fictional case here, because of his experience he was able to estimate roughly the total budget level. What he required of the computer was confirmation that what his judgement told him did indeed provide adequately for all his product lines. And, he wished to be able to make some "fine-tuning" adjustments without having to go through the hassle of refiguring. The VisiCalc program let him do all that.

As with all programs designed to aid you in working with financial information, it is necessary that you have some familiarity with the kinds of answers you are after, as well as the questions. No computer software can take the place of your good judgement.



SuperSCRIPSIT™—An Overview

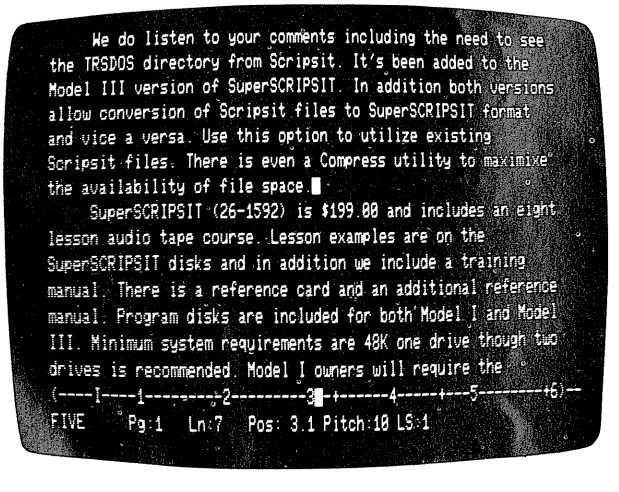
SuperSCRIPSIT has many exciting and powerful features—so many in fact that some will have to be discussed in future articles. Let's begin exploring this versatile word processor by taking a look at some of its key features.

I began this article in longhand on an airplane, but I became so frustrated with the contradiction of talking about text editing when I couldn't edit text that I pitched the article for a good book until I had access to a Model III. If you are still doing things longhand and are not sure what Word Processing is, then it's time to find out what all the excitement is about. Whether you are just getting your first look at word processing, or you are already using SCRIPSIT, we think you will find that SuperSCRIPSIT has all the "goodies" you have been looking for.

ADDITIONAL FEATURES OF SuperSCRIPSIT

Ten Programmable Keys. You can program each of the ten number keys to store up to 127 keystrokes for later recall or execution. Store a frequently used phrase, address, or signature block, or a sequence of commands or cursor movement under a single key to be recalled and executed as many times as you want with two simple key strokes.

Tab Lines and More. Each paragraph in superSCRIPSIT has its own Tab line which is used to set up the right and left margins and the tab positions for the document. The Tab line appears as the second line from the bottom on Screen 1.



Screen 1

Tab lines make it easy to highlight a quotation by using single spaced, indented copy while the rest of the document is full width and double spaced. Tab line editing includes an easy to understand set of features and the ability to save and recall as many as 11 tab lines. The stored tab lines allow instant recall of frequently used formats, and they are a boon in documents such as outlines that have complex requirements. The tab line feature makes reformatting a single paragraph fast and easy.

Features like **Align Tab** and **Centering** are easy with SuperSCRIPSIT. Align tabs allow fast entry of numeric data such as a column of prices. The align tab moves digits to the left as they are typed until a decimal point is entered. After the decimal point is entered, the decimal digits are moved to the right. The result of several entries using the align tab is a column of numbers aligned vertically by the decimal points. Any time you need information about align tabs or any of SuperSCRIPSIT's many other features, you can use the **seven help screens** that are available. (The seven help screens make up a seven page index of valid SuperSCRIPSIT commands.)

Pagination and the Status Line. As you type, SuperSCRIPSIT will **paginate** your text **automatically**. It keeps track of the line spacing and lines per page that you set when you created your document. We won't go into any detail now about the **status line**, but if you will refer to the bottom line of Screen One, you will see that the status line provides a lot of information like current line number and pitch selection as well as the current page number.

Revising a Document. Once a document is typed, SuperSCRIPSIT offers both simple cursor motion commands to move up, down, left, and right within a document and a set of advanced commands that allow movement to the next or preceding word, page number, paragraph, printed page or video page. SuperSCRIPSIT even gives you the ability to search for and replace one string of text with a new string of text.

Since SuperSCRIPSIT's documents can be as large as the available space on a disk (not to be confused with available memory), you will find that **large documents** are SuperSCRIPSIT's forte. In fact, a document of about 30,000 words can be stored on a data disk. For large documents, Block Action Commands become the key to easy editing. Once you define any amount of text as a block you may then delete it, copy it, move it, adjust it, search it, hyphenate it, print it, or change its line spacing.

Marking blocks is as easy as moving the cursor to the start of the block, marking the beginning and then moving the cursor to the end of the block and marking the end. SuperSCRIPSIT also has a text quantity method, or **speed marking command**, that will quickly define a word, sentence, paragraph, page, or all of the remaining text as a block.

PRINTING FEATURES

SuperSCRIPSIT offers an enormous variety of printing options. The first step in printing a document is the Open Document List. All the printing defaults displayed in the Open Document List except for line spacing (printer type, lines per page, pitch, line spacing, and information

on where to begin headers and footers) can be changed at this point. After the document is opened and printing is requested, the Print Text Options Screen defaults are displayed. These options include paper size, pause between printed pages, type of justification, number of copies, suppression of widow lines, and more. As you can see SuperSCRIPSIT offers many options that allow you to control the end product—the printed page.

True Proportional Printing. In proportional mode, every letter (A-Z) is assigned a specific number of units according to its width. For example, an "M" is three units wide, but an "i" is only one unit wide. The proportional mode also allows for partial spaces to aid in justifying text. This differs from SuperSCRIPSIT's Monospaced (Mono) mode where each letter and space is assigned an equal number of units.

If you look at SuperSCRIPSIT Screen 1, you will notice that there are two cursors (the two small white blocks). The lower one is a "ghost" cursor which will always appear on the "tab" line. By indicating the position where text will actually be printed, the ghost cursor helps you visualize how the text shown on the video will actually look on the printed page. This is especially useful in the proportional printing mode. With proportional justification the right and left margins on a printed document will both be even. SuperSCRIPSIT does this by automatic insertion of partial spaces between the words to fill out the lines.

You can select proportional mode printing on the Daisy Wheel II, Line Printer IV, or Line Printer VIII since these printers are capable of inserting partial spaces. Text printed using proportional justification is generally considered much nicer in appearance than text printed in a non-proportional (Mono) mode.

Headers and Footers. A header or footer consists of text that can be from one to several lines (but no more than 768 characters) in length. Headers and footers can be printed on each page of a document or only on specified pages, and either the header or the footer can include automatic page numbering. Headers are printed at the top of a page, and footers are printed at (you guessed it) the bottom of the page. An example header would be a book title which is printed at the top of every page in the book. If you notice TRS-80 Microcomputer News uses one of two page numbering footers at the bottom of each page. Our footers include the name of the publication and the month and year it was published and are right or left justified depending on the page number. I point this out just to give you an example of a footer. SuperSCRIPSIT provides for two headers and two footers per document. This means that you can print one header of text on all even numbered pages and another on all odd numbered pages. This is useful for two sided documents like Microcomputer News.

Form Letters. Form Letters are a major enhancement to the word processing capabilities for the Models I and III. SuperSCRIPSIT gives you the capability of printing the same form letter (also called a master document) multiple times, but each time the letter is printed different variables (from a Variables document) will be printed. The variables could be names, addresses, prices, or whatever you define them to be.

For example, a charitable organization wants to solicit contributions through the mail. They have a lengthy list of names and addresses of potential contributors. Using the Form Letters capability of SuperSCRIPSIT, the body of the letter received by each potential contributor would be the same, but the name and address would be different. SuperSCRIPSIT can prepare many master documents to merge with the same variables document or the same master document can be used to merge with different variable documents.

Special Printing Enhancements. You will find many enhancements in SuperSCRIPSIT, particularly in the area of printer support. SuperSCRIPSIT will be your first choice as a word processing package if you are trying to get maximum use from your printer. Our "correspondence" quality LP VIII and letter quality Daisy Wheel II, for example, can both provide features like underscore, double underscore (Daisy Wheel II only), **Bold**, strike through 0000000, ^{super}scripts for footnotes,[1] and $_{sub}$scripts for mathematical expressions and chemical names like H_2O. You can even control top of form for double pass printing and for printing two-column pages. When the program encounters the top of form code it will roll the paper down to the first printed line on the page and continue printing.

Printer Drivers. SuperSCRIPSIT will work with any current Radio Shack printer designed to work with the Models I and III. When you set up your default settings for printers, you indicate which printer you have. If your printer is a Daisy Wheel II, Line Printer IV, or Line Printer VIII you indicate that you wish to use the special drivers we have provided for these printers. If you are using a Line Printer III, V or VI, then just answer M for Mono. Also included is a serial driver which was set up for "standard" serial printers. A source listing is included for use in modifying the driver to fit your serial printer.

Special Characters. In addition to the system print codes like bold and underscore, etc. you can instruct SuperSCRIPSIT to print special characters (e.g. an English pound sign) and to perform print actions like backspace, line feed, and more. User Print Codes can be assigned to any of the numeral keys, and then those keys will insert your Print Code in the text for special printing.

A NOTE TO SCRIPSIT USERS

For many needs you may find that our original Model I/III SCRIPSIT is the most appropriate word processing package. There are no plans to "phase" out SCRIPSIT since it offers so many features at a very affordable price.

IN CONCLUSION

We do listen to your comments, including the one that pointed out the need to see the TRSDOS directory from SCRIPSIT. It's been added to the Model III version of SuperSCRIPSIT. In addition, both Model I and III versions allow conversion of SCRIPSIT files to SuperSCRIPSIT format. Use this option to utilize existing SCRIPSIT files or to work with BASIC ASCII files. There is even a Compress utility to maximize the use of available disk space.

SuperSCRIPSIT (26-1592) is $199.00 and includes an eight lesson audio tape course. Lesson examples are on the SuperSCRIPSIT disks and in addition we include a training manual. There is a reference card and an **additional** reference manual. Program disks are included for both Model I and Model III. Minimum system requirements are a 48K, one-drive system although two drives are recommended. Model I owners will require the lowercase kit.

Model I/III Bugs, Errors and Fixes

Accounts Payable (26-1554)

In versions prior to 3.0 and 3.0, the Accounts Payable program allows the user to delete any invoice at any time even when the invoice is Selected, but the # of invoices selected is not decremented.

The problem is corrected by following the steps listed below.

1. Backup the diskette(s) and make the changes on the Backup copy of the program.
2. In BASIC load the program by typing LOAD"INVOICES".
3. Make the following corrections:

Changes (Retype the line or refer to the Edit section of the owners manual)

VERSION 3.0 AND LATER (MODEL I/III)

```
Old Line: 219 K=KK:I(K)=Ø:I=I-1:IT=IT-1:ID=ID+1
          :PL=987:W1$="DELETED":GOSUB1Ø9:
          GOTO71
New Line: 219 K=KK:IS=IS+(I(K)<Ø):I(K)=Ø:I=I-1
          :IT=IT-1:ID=ID+1:PL=987:W1$=
          "DELETED":GOSUB1Ø9:GOTO71
```

VERSIONS PRIOR TO 3.0 (MODEL I/III)

```
Old Line: 795 K=KK:I(K)=Ø:I=I-1:IT=IT-1:ID=ID=1
          PL=987:W1$="DELETED":GOSUB43Ø
          :GOTO13Ø
New Line: 795 K=KK:IS=IS+(I(K)<Ø):I(K)=Ø:I=I-1
          :IT=IT-1:ID=ID+1:PL=987:W1$=
          "DELETED":GOSUB43Ø:GOTO13Ø
```

4. Type SAVE"INVOICES" to save the changes in the program.
5. AT TRSDOS Ready, make a backup copy of the corrected diskette.

Accounts Receivable (26-1555)

The version 3.0 A/R program adds a finance charge when balance = 0.00 for returned merchandise.

The problem is corrected by following the steps listed below.

1. Backup the diskette(s) and make the changes on the Backup copy of the program.
2. In BASIC load the program by typing LOAD "PROCESS".
3. Make the following corrections:

CHANGES (Retype the line or refer to the Edit section of the owners manual)

```
Old Line:  59Ø PRINT@576,"ACCOUNT : ";
           CVI(DD$);:IF(CVD(DB$)=ØAND
           ASC(DH$)=Ø)ORASC(DF$)=7THEN67Ø
           ELSEGOSUB144Ø:AB#=PB#:GOSUB166Ø
New Line:  59Ø PRINT@576,"ACCOUNT : ";
           CVI(DD$);:IF(CVD(DB$)=ØAND
           ASC(DH$)=Ø)ORASC(DF$)=7THEN67Ø
           ELSEGOSUB144Ø:AB#=PB#:FF#=Ø#
Old Line:  63Ø AB#=AB#+FC#:IFFC#>ØTHENGOSUB163Ø
New Line:  63Ø GOSUB166Ø:AB#=AB#+FC#:IFFC#>Ø
           THENGOSUB163Ø
Old Line:  164Ø LPRINTTAB(5)VC$;TAB(16)VD!;
           TAB(24)USINGL2$;VH#;:LPRINT
           TAB(5Ø)USINGL2$;AB#:RETURN
New Line:  164Ø LPRINTTAB(5)VC$;TAB(16)VD!;
           TAB(24)USINGL2$;VH#;:LPRINT
           TAB(5Ø)USINGL2$;AB#:FF#=FF#-
           (VH#*(VH#<Ø)):RETURN
Old Line:  166Ø RQ#=INT(CDBL(RQ)*1ØØØDØ+
           .5DØ)/1ØØØDØ:FI#=INT(CDBL(FC!)*
           1ØØØØDØ+.5DØ)/1ØØØØDØ:FC#=Ø:IF
           SGN(PB#+PR#)=1ANDFC$="Y"THENFC#=
           INT((PB#+PR#)*FI#*1ØØDØ)/1ØØDØ
           :FA#=FA#+FC#
New Line:  166Ø RQ=INT(CDBL(RQ)*1ØØØDØ+.5DØ)/
           1ØØØDØ:FI#=INT(CDBL(FC!)*1ØØØØDØ
           +.5DØ)/1ØØØØDØ:FC#=Ø:IFSGN(PB#
           +FF#+PR#)=1ANDFC$="Y"THENFC#=
           INT((PB#+FF#+PR#)*FI#*1ØØDØ)
           /1ØØDØ:FA#=FA#+FC#
```

4. Type SAVE"PROCESS" to save the changes in the program.
5. At TRSDOS Ready, make a backup copy of the corrected diskette.

Business Mailing List (26-1558)

In version 3.0 of Business Mailing List, the record number increases by one after Reconstruct.

```
52ØØ 'RECOVERY 3.Ø
521Ø CLS:N=Ø:TN=1:F=Ø
5215 PRINTTAB(2Ø)"** FILE RECOVERY **"
522Ø FORQ=ØTOCP:V(Q,Ø)=Ø:V(Q,1)=Ø:NEXT
5225 PRINT@32Ø,"** RECORD NUMBER :"N
523Ø N=N+1:J=N:GOSUB184Ø:GOSUB19ØØ:GOSUB191Ø
5235 IFE1$<CHR$(32)OREØ$>CHR$(127)THENN=N-1:GOTO
     525Ø
524Ø NS=1:GOSUB321Ø:TN=TN+1:EL$=E1$
5245 IFN<CPTHEN5225
525Ø GOSUB 179Ø
5255 PRINT:PRINT"> RECOVERY COMPLETE - PRESS
     <ENTER> ";
526Ø GOSUB 146Ø:GOTO 21Ø
```

The above module can be used to recover lost data or to re-sort data which is out of order. This is usually caused by system failure or abnormal exit from the Mailing List program. To perform the "recovery," follow the following procedures.

1) Enter the module (lines 5200 to 5260 above)
2) SAVE the module using the following BASIC command: SAVE "RECOVER/ASC",A ⟨**ENTER**⟩
3) Type: LOAD"MLS" and press ⟨**ENTER**⟩
4) Type: MERGE"RECOVER/ASC" and press ⟨**ENTER**⟩
5) Type: RUN and press ⟨**ENTER**⟩
6) When the Mailing List menu appears, press the ⟨**BREAK**⟩ key.
7) When the screen shows "READY", type: GOTO 5200 and press ⟨**ENTER**⟩.
8) The system will examine your data files and retrieve, re-sort, and index all the valid data found. When this process is complete, the screen will show: "RECOVERY COMPLETE — PRESS ENTER". Press a key.
9) The Mailing List Menu will appear. Press the @ key to exit the program immediately.
10) Make BACKUP copies of your "recovered" disks. Examine the data carefully before copying over your original disks. If the data is not correct, try the operation again.

NOTE The existing recover is for versions prior to 3.0 only. The above is 'RECOVER' for 3.0 BML. This can also be used for versions prior to 3.0.

Medical Office Systems (26-1568)

During edit transactions of Model III Medical Office Systems version 1.0, if you add a third line (edit line 3) to an account that has two transactions, the inserted line 3 will be found on the next account.

To correct this problem follow the procedure below.

1. At TRSDOS READY, type BASIC.
2. Type LOAD"EDITOR".
3. Add the following line.

```
1185 IFCL>NLTHEN1160
```

4. Type SAVE"EDITOR" to save the change.
5. At TRSDOS READY, make a backup copy of the corrected diskette.

Real Estate Vol. I (26-1571)

In the "Rate of Return" program in Model I Real Estate Volume I version 1.0, the modified internal rate equals the regular internal rate of return. To correct this follow the steps listed below.

1. CLOAD the Rate of Return tape.
2. Make the following corrections:

CHANGES (Retype the line or refer to the Edit section of the owner's manual.)

```
Old Line: 800 R1=R1+R2:D9=0:D0=C:FORI=1TOP1-1
          :ONAGOTO810,820
New Line: 800 R1=R1+R2:D9=0:D0=C:IFA=2THEND0=
          D0-(D0*R3)
```

ADDITION: Type line 805 followed by an ⟨**ENTER**⟩.

```
805 FOR I=1TOP1-1:ONAGOTO810,820
```

3. Type CSAVE"A" to save the changes in the program.

Real Estate Vol. III (26-1573)

In Real Estate Vol. III version 3.0 if the "End of Period Payment" is used, the first factor in the factor column on the hardcopy option is deleted and all factors are moved up by one. The screen display is alright.

The problem is corrected by following the steps listed below.

1. In BASIC load the program by typing CLOAD"VARI".
2. Make the following corrections:

CHANGES (Retype the line or refer to the Edit section of the owners manual)

```
Old Line: 6070 B=(1/(1+(B/N1)))^(I-1):GOTO6090
New Line: 6070 IFJ=1THENB=(1/(1+(B/N1)))^(I-1)
          :GOTO6090
```

3. Type CSAVE"VARI" to save the changes in the program.

NOTE On the Model III, the "up-arrow" key appears as a left bracket on the screen but it functions as an exponentiation symbol in the code.

Micro Movie (26-1903)

IMPORTANT NOTICE FOR 4K MODEL III USERS — When operating Micro Movie on a 4K TRS-80, the Save Screen in Memory option cannot be used. On the TRS-80 Model I, the screen will show NO ROOM IN MEMORY if this option is requested. If you request this function on the Model III, the request is simply ignored. If you return to the menu, the screen contents will be lost.

Model III Operation and BASIC Language and Reference Manual (26-2112)

Corrections need to be made to page 43 of this manual. Under the sentence 'When you start the Computer, the RS-232-C is initialized to the following "default characteristics":' change the information so it reads:

Baud Rates	300
Word Length (bits)	8
Parity	None
Stop Bits	1

COBOL Compiler (26-2203)

Following are two corrections for the COBOL Compiler version 1.3b. The first corrects a problem that occurs when attempting to call an independent segment overlay. The second corrects a problem that occurs when using the TAB option with the ACCEPT statement in an indexed file.

1 — When RUNCOBOL (MOD I/III) attempts to call an independent segment overlay, it will have a LOAD FAIL ERROR message and return to DOS. RUNCOBOL is unable to load and execute the machine language program (independent segment) and is forced to return to TRSDOS.

The following PATCHes will correct the above problem:

```
PATCH RUNCOBOL/CMD (ADD=9DB8,FIND=CD11AEC2419F,
                    CHG=000000000000)
PATCH RUNCOBOL/CMD (ADD=9DC4,FIND=2A,CHG=21)
PATCH RUNCOBOL/CMD (ADD=AD0D,FIND=19,CHG=09)
PATCH RUNCOBOL/CMD (ADD=ACFE,FIND=D60A,CHG=C318)
PATCH RUNCOBOL/CMD (ADD=AD00,FIND=30,CHG=9A)
PATCH RUNCOBOL/CMD (ADD=9A18,FIND=00000000,
                    CHG=FE0ADA04)
PATCH RUNCOBOL/CMD (ADD=9A1C,FIND=00000000,
                    CHG=ADD611DA)
PATCH RUNCOBOL/CMD (ADD=9A20,FIND=00000000,
                    CHG=11ADFE06)
PATCH RUNCOBOL/CMD (ADD=9A24,FIND=00000000000000,
                    CHG=D211ADC302AD)
PATCH RUNCOBOL/CMD (ADD=9B65,FIND=2A69AE4E23,
                    CHG=46ED43D1AF)
PATCH RUNCOBOL/CMD (ADD=9B6A,FIND=46ED43D1AF,
```

2 — On Model I COBOL Ver. 1.3b and TRSDOS 2.3b, when using the TAB option with the ACCEPT statement in an indexed file, improper program halts occur, contrary to the usage described on pages 139-144 of the RSCOBOL manual.

Below is a program used to create an indexed file. The current ACCESS clause stipulates a RANDOM file, but this may be changed to DYNAMIC or SEQUENTIAL if desired.

(Continued on Page 41)

The TRS-80 AUTHOR I System

Lesson Development Without Programming

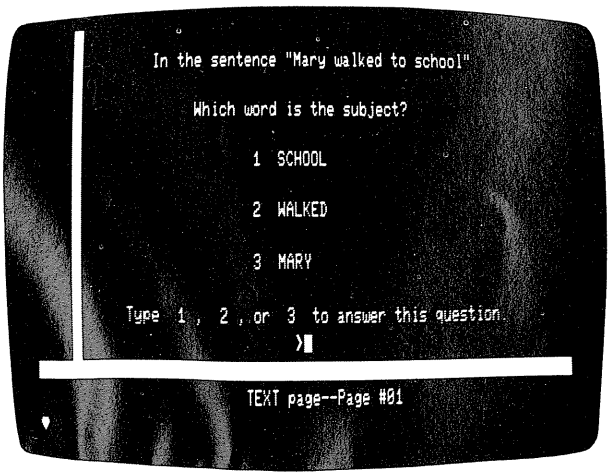
A new authoring system from Radio Shack, TRS-80 AUTHOR I, makes it possible for teachers and curriculum designers to develop microcomputer-based lessons in any subject area. No programming experience on the part of the lesson developer is required.

Almost any course material can be adapted to the TRS-80 through AUTHOR I. AUTHOR I Lessons can include graphics, text, questions for the student to answer, feedback messages and hints for the student, and a glossary built into the lesson. By following a few simple steps the teacher can even design lessons that include branching. Different paths can be set up within the same lesson, and the computer will automatically "branch" students along one path or another as determined by each student's performance on key questions.

With AUTHOR I you can create versatile, sophisticated educational materials which present all of the lesson content available through more traditional media but which are interactive, immediate, and individualized in ways no book or film can match.

CONSTRUCTING A LESSON WITH AUTHOR I

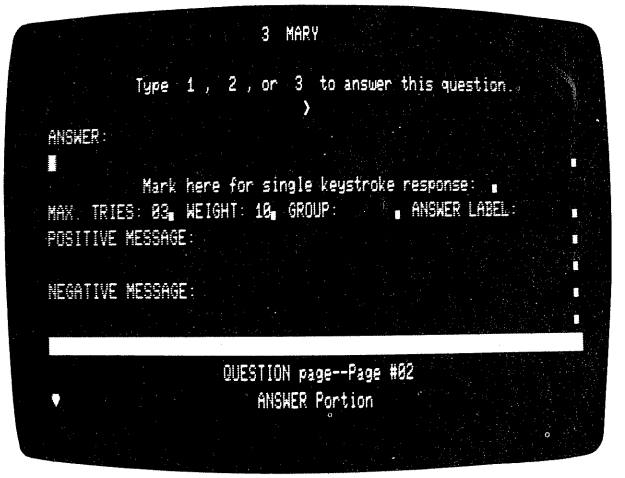
TRS-80 AUTHOR I is a screen-oriented system, which means that construction of the lesson is based on a screen-by-screen approach. This approach enables the lesson developer (teacher) to type lesson material onto the TRS-80 Model III or Model I screen exactly as he or she wants the students to see that material during the lesson. The teacher can easily draw graphics onto the screen after pressing a couple of keys to switch into the graphics mode. Screen One shows an example of a lesson text page as it might appear to the lesson developer while he or she created the page.



Screen 1

Instructions for the computer are easily entered into the computer. AUTHOR I includes specialized screen displays where the teacher can type branching instructions, feedback messages to be stored until they are needed, and information that will help the computer evaluate student responses to lesson questions. In each case, the screen display guides the teacher with well-defined information fields and, where appropriate, with a menu of possible actions at the bottom of the display. These specialized displays allow the teacher to type instructions simply and literally, without having to use any computer language commands.

Let's take a look at one of these specialized displays. Screen Two shows an AUTHOR I "Question Page," where the teacher types information that the computer will use in judging student responses to a question. The Question Page is a representative example of a page used to direct the lesson:



Screen 2

First, the Question Page allows the teacher to record one or more acceptable answers to a question, including different forms of the same answer. Next, the teacher can place a mark in the "single keystroke" space if he or she wants the computer to accept and judge the response as soon as one character has been typed.

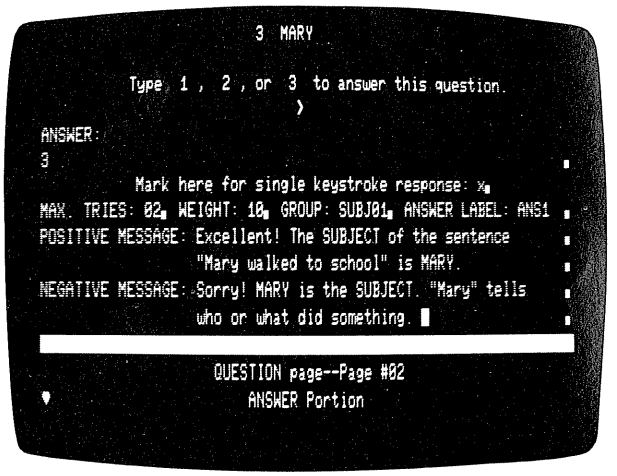
The "MAX. TRIES" category then provides a place for the teacher to specify the number of tries available to each student on this question, or to accept the default value of three tries. (As soon as this screen display is completed, the teacher will be able to use the Hints display to record as many as five hints per question. Hints keyed to specific incorrect responses or to the number of tries are then

"lifted" individually from the Hints display and shown to the student after incorrect responses.

Next, the "WEIGHT" category allows the teacher to assign a value to this question in relation to all other questions in the lesson or in the question group. If the teacher wished to define this question as part of a question group within the lesson, he or she would then type a six-character label in the "GROUP" field. Similarly, a six-character label in the "ANSWER LABEL" field would identify this one question distinctly. Later, the teacher could use the scores and responses that these labels represent in student reports and as criteria for branching (sending different students along different lesson paths).

Finally, the teacher can fill in the "POSITIVE MESSAGE" and "NEGATIVE MESSAGE" fields with feedback messages to be displayed to the student when he or she answers the question correctly or runs out of allowed tries.

Screen Three provides a view of what the same Question Page might look like once it had been completed.



Screen 3

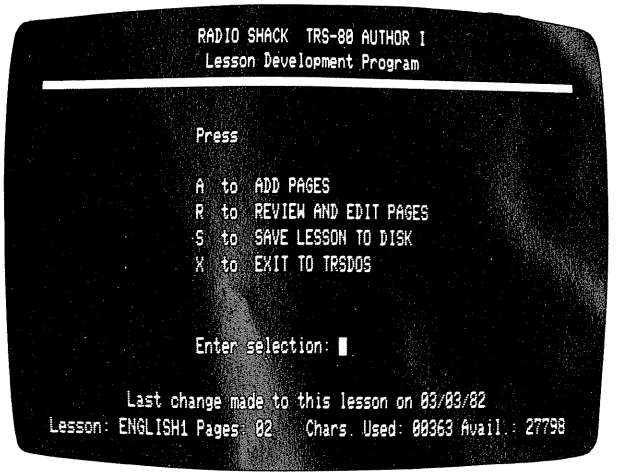
ORGANIZATION OF THE PROGRAM

So far we've only looked at part of the lesson-development segment of the AUTHOR I system, but the system actually includes much more. AUTHOR I is made up of a total of five modules, or five programs that allow you to work with a lesson in different ways.

The first of these five modules, the AUTHOR module of TRS-80 AUTHOR I, is the module that makes possible the creation of lessons. To create a lesson you load the AUTHOR module and enter a lesson name. The computer then allows you to begin constructing the lesson by adding pages or, if your lesson already exists, to select another AUTHOR module option, as shown in screen 4.

When you have finished adding pages to your lesson, you can select the Review/Edit option, which will allow you to go back and look over the pages you have just added to make corrections. When you have finished editing your lesson, you can save it to the diskette, and a lesson file distinct from the AUTHOR program will be set up.

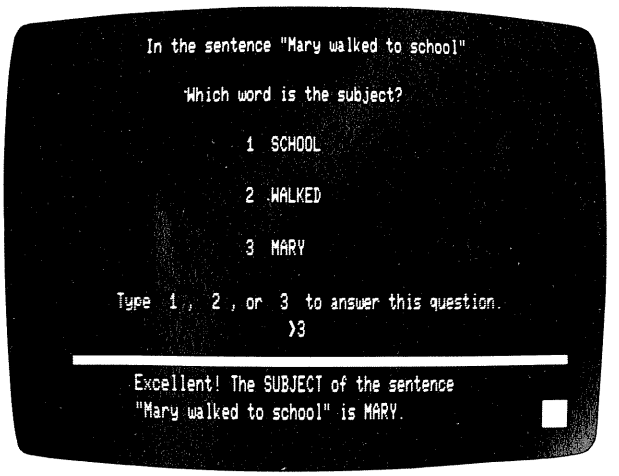
The PRINT/VERIFY module of AUTHOR I allows you to print out a hard copy (paper copy) of the computer-based lessons you have designed. Your printout will include both the lesson screens seen by the student, and information



Screen 4

displays like the Question Page described above. PRINT/VERIFY will also provide you with a cross-reference of all of the labels you have used in your lesson, whether these are labels marking particular parts of the lesson, or group or answer labels as described above.

AUTHOR I's TEACH module is the lesson presentation portion of AUTHOR I. To present a lesson the teacher or student loads the TEACH program and enters a lesson name. After that the system does the rest, presenting the lesson as you designed it. Screen Five shows a lesson question as the student would see it through TEACH. To answer this question, the student simply types the answer at the computer keyboard and presses ⟨**ENTER**⟩.



Screen 5

When the student has finished reading a text display or answering a question (that is, has answered the question correctly or has exhausted all available tries), he or she simply presses the space bar to continue to the next display. Pressing the space bar after an incorrect response when more tries are allowed returns the student to the original question screen to try again.

The STUDENT module of AUTHOR I enables you to set up student score files for your AUTHOR I-based lessons. Once you set up a lesson's score file and enroll students, the computer will record performance information

each time a rostered student completes the lesson. Both the student's total percentage of correct answers and the total amount of time the student took to complete the lesson are recorded automatically. Additional information that can be recorded includes sub-scores and sub-timings for selected parts of the lesson. One or two of the student's responses can also be recorded verbatim for your later review.

The fifth module of AUTHOR I, CHAIN, is not directly used by the lesson developer or teacher, but it exists on the AUTHOR I diskette to make it possible to branch students from within an AUTHOR I lesson to a location outside of that lesson. External branching can be from one AUTHOR I lesson to another, from an AUTHOR I lesson to a BASIC program, or from an AUTHOR I lesson to a program written in machine language.

AUTHOR I PACKAGES FROM RADIO SHACK

Three AUTHOR I or AUTHOR I-based packages are now available through your local Radio Shack store or Computer Center.

TRS-80 AUTHOR I (the complete lesson authoring system, catalog number 26-1727) contains all five of the AUTHOR I modules on diskette for use with TRS-80 Model III or Model I. It allows you to take advantage of all AUTHOR I features: you can create, edit, and administer lessons of all types for your school or district, and can keep student score files. The minimum system required for operation is a 32K TRS-80 with one disk drive, but 48K is recommended if you will be developing fairly long, complex lessons.

The TRS-80 AUTHOR I Lesson Presentation Package (cat. no. 26-2707) contains the TEACH and STUDENT modules of AUTHOR I.

This package allows you to present AUTHOR I-based lessons that you have purchased separately or that have been created by others in your school district, and it enables you to set up score files for all of the lessons. The minimum system required for use of the Lesson Presentation Package is a TRS-80 Model III or Model I 32K system with one disk drive. Some lessons that you purchase, however, may require the use of a 48K disk system and/or may run on Model III or Model I only.

The Radio Shack Computer Assisted Reading Development Program (C.A.R.D.), adapted from the Philadelphia Computer Assisted Reading Development Program, is the first of Radio Shack's lesson offerings for use with the TEACH and STUDENT modules of AUTHOR I. C.A.R.D. 1: Sentences (cat. No. 26-2603) requires a 48K Model III system with at least one disk drive. Either TRS-80 AUTHOR I (26-1727) or the TRS-80 AUTHOR I Lesson Presentation Package (26-2707) may be used to present the C.A.R.D. lessons and to set up lesson score files. For more information on the C.A.R.D. 1: Sentences program, read the article on C.A.R.D. also in this month's newsletter.

The listed price for TRS-80 AUTHOR I (26-1727) is $149.95. The TRS-80 AUTHOR I Lesson Presentation Package is listed at $64.95, and C.A.R.D. 1: Sentences is listed at $199.00. Prices, however, may vary at individual stores and dealers.

Introducing Radio Shack's C.A.R.D. 1: Sentences

A Staple Item for C.A.I. Reading

Radio Shack's new courseware package **C.A.R.D. 1: Sentences** (C.A.R.D. — Computer Assisted Reading Development) is the first package in a four-part reading program designed to run on a TRS-80 Model III 48K system with a minimum of one disk drive, and is itself a comprehensive, multi-lesson program that can play an important role in school C.A.I. reading curricula.

C.A.R.D. 1: Sentences was adapted to the TRS-80 from the Philadelphia Computer Assisted Reading Development Program. The original program was developed by Philadelphia City Schools educators and was tested, refined, and used with success in Philadelphia schools over a fourteen-year period.

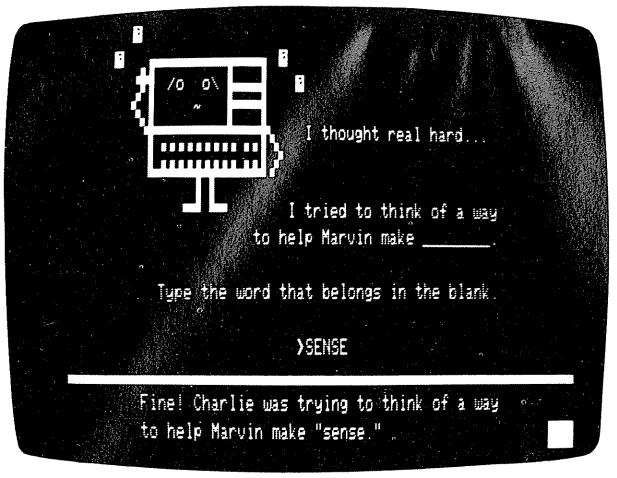
The program is directed at improving reading comprehension through tutorial reading development lessons in four skill areas: Sentence Recognition, Labeling, Sentence Relationships, and Ordering Sentences. C.A.R.D. 1 has been used with students in the fourth grade, in secondary school, and at all levels in between. It has been used in particular with intermediate and secondary students who have no serious word recognition problems but whose scores on standardized tests indicate that their reading comprehension is below grade level.

Each of the four topics (skill areas) in C.A.R.D. 1 is covered through a pretest, six or seven developmental lessons, and a posttest that duplicates the pretest in content. Topics are meant to be taken in order, as are lessons with each topic. Before the student begins a topic, he or she takes the pretest. If the student passes the pretest for the first topic, the pretest for the second topic is taken, and so on. If the student does not pass pretest, then he or she works through the lesson sequence for that topic. After taking the lessons, the student tackles the posttest for that topic and should be able to pass the posttest with a score of 75% or above before progressing to the next topic.

C.A.R.D. lessons are designed to interest students by presenting reading concepts in a clear, imaginative way. Early lessons capture the student's interest through an on-screen story about a visitor from Mars who needed to be taught to recognize sentences. Graphics are used to illustrate the concepts being discussed.

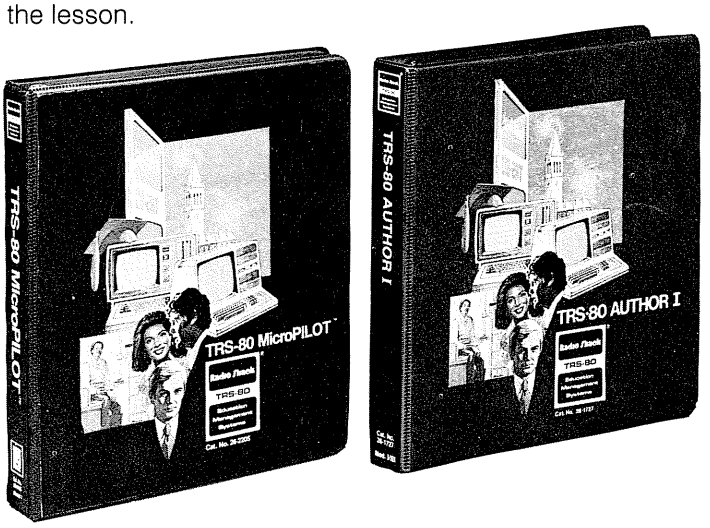
Later lessons hold the student's interest through devices like using example sentences whose subject matter includes school activities, sports, and hobbies. Mini-narratives in one lesson invite the student to think of sentence order as comparable to the order of books on a library shelf, to order in the student's closet at home, and to sequential order in the movies the student goes to see.

Numerous questions about the lesson material play an important part in C.A.R.D. 1 lessons. When the student



responds to a question, the computer provides immediate, constructive feedback, including positive messages for correct responses and hints and/or correction for incorrect responses. At the end of each lesson the student's score is displayed on the screen, along with an appropriate message.

Key questions within the lesson are used as criteria for "branching" students through different lesson paths. When the student answers one of these questions, the computer immediately evaluates the response and automatically sends that student along the lesson path appropriate to his or her performance. If the question was missed, the student will receive additional instruction. If the student answered the question correctly, he or she may be branched past remediation to a more advanced section of the lesson.



Lessons in C.A.R.D. 1 were adapted to the TRS-80 through the TRS-80 AUTHOR I system, and require either the TRS-80 AUTHOR I Lesson Presentation Package (cat. no. 26-2707) or TRS-80 AUTHOR I (cat. no. 26-1727) for operation. If you use the STUDENT module of either AUTHOR I package to set up score files for the C.A.R.D. 1 lessons, the computer will automatically record performance information each time a rostered student completes a lesson. The student's total score will be displayed, along with the total amount of time the student took to complete the lesson.

The general objectives for each of the four topics in C.A.R.D 1: Sentences are as follows. The passing criterion for each test or lesson is a score of 75%.

TOPIC ONE: SENTENCE RECOGNITION

Given a series of groups of words, identify those groups of words which are sentences and those groups of words which are not sentences.

TOPIC TWO: LABELING

Given a series of keywords (or given several words in a sentence) and given a selection of complete or partial topic labels, select the complete or partial topic label that best expresses the relationship between the keywords.

TOPIC THREE: SENTENCE RELATIONSHIPS

Given a series of sentences, identify those sentences which have common or related meanings and those which do not.

TOPIC FOUR: ORDERING SENTENCES

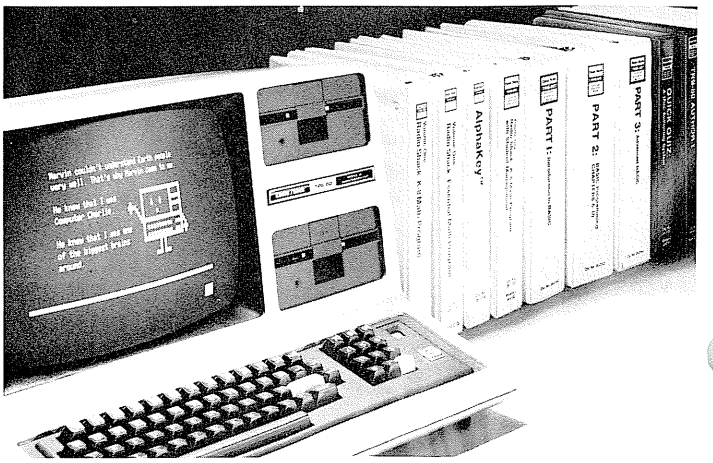
Demonstrate an understanding of sequence by ordering a series of sentences or concepts so that they logically expand or develop an idea.

A project report prepared by the Philadelphia School District testifies to the success of the program as implemented on timesharing systems in the Philadelphia schools. The report states that "CAI students made greater gains (in reading) than non-CAI students. . . . Teachers and administrators who were interviewed thought that the student profited not only from the individualized activities (provided by the program) but also from the motivation associated with CAI."

As originally used by the Philadelphia Schools, C.A.R.D. programs were run on teletype terminals accessing a large mainframe computer. Conversion of C.A.R.D. 1: Sentences to the Model III now adds convenience and greater accessibility to a high-quality, proven reading program.

Three other C.A.R.D. packages, "Paragraphs," "Directions" and "Comprehension" are scheduled for release after C.A.R.D. 1.

C.A.R.D. 1 is designed for use with the TRS-80 Model III 48K system with a minimum of one disk drive. The suggested retail price of C.A.R.D. 1: Sentences (cat. no. 26-2603) is $199.00, although prices may vary at individual stores and dealers. For further information, contact your local Radio Shack store or Computer Center.



Power-Up Sequence for the Model II

Many Model II owners have expressed confusion about the correct power-up sequence for the Model II when peripherals are attached.

TO TERMINATE OR NOT?

Beginning on September 24, 1981, Manufacturing began installing a different FDC (Floppy Disk Controller) board in the Model II. For the 32K Model II, the FDC board change began with serial #32001700. In the 64K Model II, the FDC board change began with serial #64036930.

Model IIs manufactured prior to September 28, 1981 include a terminator plug which **has** to be plugged into the Disk Expansion port if an expansion bay is not attached. If one of these early systems is configured with external drives, the disk bay must be turned on, even when the external drives are not being used. Neglecting to install the terminator plug (in the absence of a disk bay) or failure to turn on an attached expansion bay when operating the computer could render the data on the disk in drive 0 unreadable. If you have an early Model II, always be sure that the terminator plug is installed if you do not have an expansion bay, or if you have an expansion bay make sure that it is always on when operating the computer.

Later Model IIs (those beginning with serial #32001700 for 32K machines or serial #64036930 for 64K machines) do not require the use of a terminator plug and can be operated with the disk bay turned off without causing any loss of data in drive 0.

POWER UP PROCEDURE FOR MODEL IIs WITH PERIPHERALS

- Floppy diskettes should not be in any drive (0-3) in the system when the computer or the expansion bay is turned on or off. If diskettes are left in any drive during power-up or down procedures, they may be rendered unreadable.
- The Model II is turned on first except in systems that include hard disks. This is a general rule that would cover any peripheral except the hard disk.
- If you have a hard disk with your Model II, the hard disk has to be powered up before the CPU to enable the automatic boot software sequence. The hard disk drives require a one minute warm-up before the system can be booted. When you want to boot from and use floppy diskettes exclusive of the hard disk, you do not need to power-up the hard disk(s).
- If your system includes secondary hard disks, they should **never** be turned on before the primary hard disk drive. If they are turned on while the primary hard disk drive is off, data could be erased on the secondary hard disks.

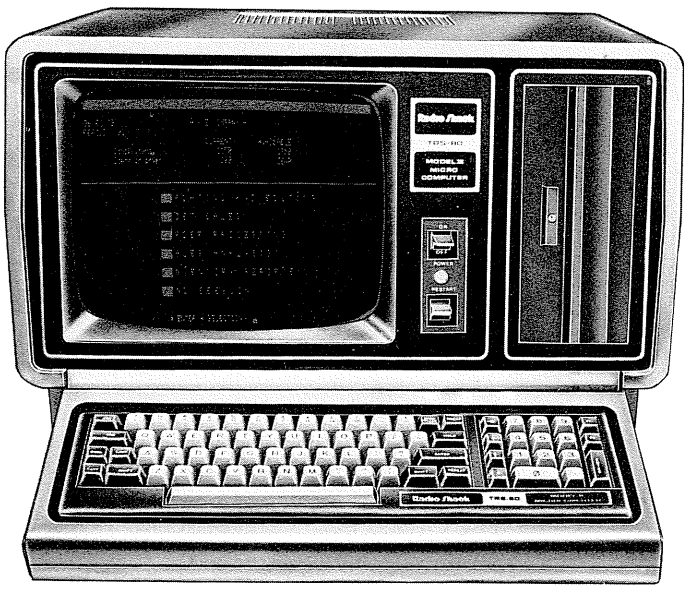
RECOMMENDED POWER-UP SEQUENCE

1. Check to make sure that there are no floppy diskettes in any of the drives and that all components of the system are turned off.
2. If you have the early Model II which comes with the terminator plug make sure that the terminator plug is installed or the disk bay is attached.
3. If you have hard disk drives and intend to use them
   - Turn on the primary drive
   - Next, turn on any secondary hard disk drive(s)
   - Allow one minute warm-up
4. Turn on the Model II.
5. Turn on any additional peripherals (including the disk bay on early Model IIs) that you intend to use.

RECOMMENDED POWER-DOWN SEQUENCE

1. Unload all floppy disk drives.
2. Turn off all peripherals including the expansion disk unit but not the hard disk drives.
3. If you have hard disk drives then:
   - Turn off the secondary drives, if applicable
   - Turn off the primary drive
4. Turn off the Model II

If followed, these power-up and power-down sequences should allow for the reliable operation of the Model II.



Model II Bugs, Errors and Fixes

Accounts Payable (26-4505)

Below are two sets of corrections for problems with Accounts Payable (26-4505).

Correction 1 — Accounts Payable version 1.0 does not allow the editing of YTD purchases/payments.

This is corrected by following the steps listed below.

1. Backup the diskette(s) and make the changes on the Backup Copy of the program.
2. In BASIC, load the program by typing LOAD "APS/BAS".
3. Make the following corrections:

CHANGES (Retype the line or refer to the Edit section of the owners manual)

Old Line:
```
200 IFW=1THEN230ELSEIFW$=CHR$(8)
    THENPRINTW$;FL$;B1$;:W=W-1:
    MID$(IN$,W,1)=FL$:GOTO140
```

New Line:
```
200 IFW=1THEN220ELSEIFW$=CHR$(8)
    THENPRINTW$;FL$;B1$;:W=W-1:
    MID$(IN$,W,1)=FL$:GOTO140
```

Old Line:
```
1800 GOSUB8500:PRINT@(23,42),;:
     FL=1:GOSUB1000:IFCF=1THEN1000
     ELSEIFCF=5THEN1800
```

New Line:
```
1800 GOSUB8500:PRINT@(23,42),;:
     FL=1:SZ=1:GOSUB1000:SZ=0:IFCF=1
     THEN1000ELSEIFCF=5THEN1800
```

ADD the following new lines.

```
222 IFSZ<>1THEN230ELSEMI#=A1#-A2#
    :IFASC(W$)=19THENPRINT@(22,20),
    "YTD PURCHASES & CREDITS ";:
    INPUTA1#:N#=A1#:GOSUB5000:
    AA$=V$:GOTO226

224 IFASC(W$)=16THENPRINT@(22,20),
    "YTD PAYMENTS & DEBITS ";:
    INPUTA2#:N#=A2#:GOSUB5000:
    AB$=V$ELSE230

226 GOSUB9040:PRINTN$;:GOSUB8400:
    GOSUB9200:IN$="X":RETURN

5000 CP#=CP#-MI#:W#=ABS(N#)*100+.1
    :X=W#/D1#:W#=W#-X*D1#
    :V$=CHR$(X-(N#<0)*128)
    :X=W#/D2#:W#=W#-X*D2#
    :V$=V$+CHR$(X):X=W#/D3#
    :W#=W#-X*D3#
    :V$=V$+CHR$(X)+CHR$(W):RETURN
```

4. Type SAVE"APS/BAS" to save the changes in the program.
5. Run the program using:
   CTRL S TO Edit PURCHASES
   CTRL P TO Edit PAYMENTS
   From VENDOR FILE maintenance menu.
6. This change is made especially to allow you to correct balances which are off by one cent due to rounding errors.
7. At TRSDOS READY, make a backup copy of the corrected diskette.

Correction 2 — The Accounts Payable (26-4505) version 1.0 program allows the user to delete any invoice at any time even when the invoice is Selected or Held. The program should not allow this to happen. The operator can change the variable by looking at line 1000 at APS/BAS. This is a probable cause for Error Code 61 in this program.

The problem is corrected by following the steps listed below.

1. Backup the diskette(s) and make the changes on the Backup copy of the program.
2. In BASIC, load the program by typing LOAD"APINVCE/BAS".
3. Make the following corrections:

CHANGES (Retype the line or refer to the Edit section of the owners manual)

VERSIONS PRIOR TO 2.0 (MODEL II)

Old Line:
```
795 K=KK:I(K)=0:I=I-1:IT=IT-1
    :ID=ID+1:PL=1635:W1$="DELETED"
    :GOSUB430:GOTO130
```

New Line:
```
795 K=KK:IS=IS+(I(K)<0):I(K)=0
    :I=I-1:IT=IT-1:ID=ID+1:PL=1635
    :W1$="DELETED":GOSUB430: GOTO130
```

4. Type SAVE"APINVCE/BAS" to save the changes in the program.
5. At TRSDOS READY, make a backup copy of the corrected diskette.

Business Mailing List (26-4506)

If you run SETUP and choose the option for a revision, the revised information will not be stored under the appropriate filename in version 2.0 of Business Mailing List.

The following routine can be used to change the name you assigned to the Mailing List system (MAILING LIST 26-4506). With a BACKUP of your Mailing List in drive 0, RUN the following program while in BASIC:

```
10 CLEAR 500
    : DEFINT A-Z
    : DIM C$(7)
    : CLS
    : PRINT@1525, "This program allows you to
    change the Mailing List system name as it is
    stored in the system information data file.
    You must rename all three Mailing List files
    after running this routine."
20 PRINT@410,;
    : INPUT "ENTER THE CURRENT MAILING LIST
    SYSTEM NAME"; F0$
    : GOSUB 90
30 PRINT@490,;
    : INPUT "ENTER THE NEW MAILING LIST SYSTEM
    NAME"; SN$
    : GOSUB 50
40 PRINT@985, "CHANGE COMPLETE"
    : PRINT@1840,;
    : END
50 OPEN "O", 1, F0$
60 PRINT#1, SN$
    : PRINT#1, NA$
    : PRINT#1, TL$
    : PRINT#1, AD$
    : PRINT#1, CSZ$
    : PRINT#1, FM$
70 PRINT#1, CP; M; M0; M1; M3; M4; M5; M6; M7; M8
80 FOR I=0 TO 7
    : PRINT#1, C$(I)
    : NEXT
```

```
     : CLOSE 1
     : RETURN
9Ø FØ$=FØ$+"Ø:Ø"
     : OPEN "I", 1, FØ$
1ØØ LINEINPUT#1, SN$
     : LINEINPUT#1, NA$
     : LINEINPUT#1, TL$
     : LINEINPUT#1, AD$
11Ø LINEINPUT#1, CSZ$
     : LINEINPUT#1, FM$
12Ø INPUT#1, CP, M, MØ, M1, M2, M3, M4, M5, M6,
    M7, M8
13Ø FOR I=Ø TO 7
     : LINEINPUT#1, C$(I)
     : NEXT
     : CLOSE 1
     : RETURN
```

NOTE

After completion of the above you must rename the three Mailing List data files. At TRSDOS READY type

RENAME filename0 TO newname0 ENTER
RENAME filename1 TO newname1 ENTER
RENAME filename2 TO newname2 ENTER

where 'filename' corresponds to the current Mailing List system, and 'newname' is the name you are changing it to.

SCRIPSIT (26-4531)

Below are two sets of PATCHes to correct two different problems on SCRIPSIT 2.0.

Patch 1 — These PATCHes were previously published in the February, 1982 Microcomputer News (page 35) with unintentional errors. The PATCHes below contain the necessary corrections.

When a large outline format like the one shown in the example is used, SCRIPSIT will not format text correctly.

Example:

```
--------o ------------------------------------------ {-------}------------"
```

NOTE: This is the second PATCH to SCRIPSIT 2.0 and should not be applied until after the 10/16/81 PATCH has been applied. If you do not have the 10/16/81 PATCH, contact either a Radio Shack Computer Center or Computer Marketing Representative at a Radio Shack store to get it.

The following PATCHes will correct this problem:

1) At TRSDOS READY apply these PATCHes on a backup copy of SCRIPSIT 2.0.

```
PATCH SCRIPSIT A=9EF8 F=FD7571 C=227491Ø
PATCH SCRIPSIT A=9F5D F=7DFD9671 C=AFC3DØD6
PATCH SCRIPSIT A=9FBE F=8Ø47 C=ØØØØ
PATCH SCRIPSIT A=D6DØ F=ØØØØØØØØØØØØØØØØ
                       C=D5ED5B749ØED527D
PATCH SCRIPSIT A=D6D8 F=ØØØØØØØØØØØØ
                       C=D1E1E5C3619F
PATCH STARTUP A=E4ØE F=B1BØAFB1B6AFB1B9B8B1
                      C=B1B1AFB1B1AFB1B9B8B1
```

2) When these PATCHes are made the SCRIPSIT initialization screen will show the date 11/11/1981 at the bottom.

Patch 2 — SCRIPSIT will occasionally increment a character during printing if the character follows a tab which is followed by a space.

The following PATCHes will correct this error:

1) At TRSDOS READY apply these PATCHes on a backup copy of SCRIPSIT 2.0.

```
PATCH SCRIPSIT A=D6CØ F=ØØØØØØØØØØØØØØØØ
                       C=7EFE2ØCA51F8FEØ9
PATCH SCRIPSIT A=D6C8 F=ØØØØØØØØØØØØ
                       C=CA51F8C33FF8
PATCH SCRIPSIT/SYS R=158 B=81 F=BE2812
                                C=C3CØD6
PATCH STARTUP A=E4ØE F=B1B1AFB1B1AFB1B9B8B1
                      C=B1B1AFB2B4AFB1B9B8B1
```

2) When these PATCHes have been made, the SCRIPSIT initialization screen will show the date 11/24/1981 at the bottom. **The 10/16/1981 PATCHes must be made before these PATCHes are made. See note above.**

3) You are now ready to resume use of SCRIPSIT 2.0.

RSCOBOL Compiler (26-4703)

Version 1.3b of RSCOBOL does not produce a null (zero or empty) file when an open-output, followed by an immediate close is performed.

The following PATCH will correct this problem.

```
PATCH RUNCOBOL A=794F F=FD366EØØDD7E1ØB72ØØ9
                       C=DD7E1ØB72ØØDFD366EØØ
```

BASIC Compiler (26-4705)

When transferring RSBASIC (BASIC Compiler version 2.4) to a TRSDOS 4.0 Hard Disk system, RSBASIC will not recognize drive numbers 4 through 7.

The following PATCH will correct this problem.

1. At TRSDOS READY apply this PATCH to a backup copy of your 4.0 TRSDOS Diskette.

```
PATCH RSBASIC/OLF R=27 B=53 F=12DB C=CEØØ
```

Bisync 3270 (26-4715)

The following PATCH for Bisync (26-4715 version 1.0) allows the Model II, when operating in a 3275 polled environment, to respond to EOT's. This will allow multiple Model II's, emulating 3275's, to operate on the same line with specific or general polling, as long as the Control Unit Addresses are unique.

At TRSDOS READY, apply the following PATCH.

```
PATCH BIS327Ø A=7D4Ø F=CC6Ø7D C=ØØØØØØ
```

Bisync 3780 (26-4716)

The following PATCH for Bisync (26-4716 version 1.0) corrects problems with form feeds. The problem is characterized by invalid 'line skips' and 'page ejects.'

The following PATCH corrects the problem and should result in properly printed documents.

1. Make a backup copy of your Bisync 26-4716 Package.
2. At TRSDOS READY, apply the following PATCH to your diskette.

```
PATCH BIS378Ø A=4BDD F=C3EC C=C3BE
```

HARD DISK Interfacing Applications

To interface the following packages, certain precautions and techniques are necessary. Before attempting to interface the following packages ON HARD DISK note the following.

NOTES FOR:

26-4601 General Ledger
26-4604 Accounts Receivable
26-4605 Accounts Payable
26-4607 Order Entry/Inventory
26-4608 Sales Analysis

If you intend to run two or more of the above programs on Hard Disk you need to be aware that:

1. The Main Menu of each program above causes a copyright notice screen to appear the first time you bring up the menu. The first line of the copyright notice is the name of the program, e.g. GENERAL LEDGER. With multiple programs, however, the copyright notice will be from the last version of the program CPYRIGHT/COB that was FCOPYed to Hard Disk. This will have no adverse effect on the operation of your programs.
2. Three of the above programs, G/L, A/R, and A/P, all maintain the same company file. For instance in G/L, selection 1 from the RUNGL2 program will cause the company file maintenance program (COMPMNT/COB) to run. The program and the company information created is identical for each of the three programs. With multiple programs you will still be able to maintain this file from each of the three programs. However, when you exit the maintenance program of any of the three programs—G/L, A/R, and A/P, you will return to the menu of the program that was last moved to Hard Disk. When this happens, simply press the TAB key to return to TRSDOS READY and then restart your desired program. It should also be noted that if for some reason you had different company information for each of the above programs on your Floppy disk it will no longer be possible on Hard Disk.
3. As a general rule — whenever you exit from Company Maintenance function, return completely to TRSDOS READY and restart your desired program.

Example: You first move G/L to Hard Disk and then move A/R to Hard Disk (both with FCOPY). If you RUN A/R and perform the Company Maintenance function from it, you will be returned to the proper point within A/R at the completion of your changes. If, however, you RUN G/L and select the Company Maintenance function, at its completion you will be returned not to G/L but to A/R.

Chaining Subroutine for FORTRAN

John L. Montgomery
3010 Barcody Road
Huntsville, AL 65002

I am teaching an Entry Level Course in FORTRAN at a local university using a Model II Computer. In order to provide the students with the ability to chain FORTRAN jobs together I developed a chaining subroutine which, if called from FORTRAN, will load and execute another program. The program can be any program that is executable at the TRSDOS READY level or BASIC if the program name, files and memory size are attached.

The subroutine called FCHAIN('PROG',N) performs the chaining task. Another subroutine called SYSTEM('SYSCMD',N) performs exactly like the BASIC statement of the same name (i.e. go to the system, perform what is called for, and return to caller). Using these subroutines along with Mr. Robert G. Minty's method for preserving common data areas described in the July 1981 issue of Microcomputer News will provide nearly all of the flexibility inherent in TRSDOS to the FORTRAN programmer.

I have included some other incidental screen handling subroutines which were used in the demonstration program (Program One). The demonstrations should be self explanatory. Program One calls Program Two which performs several system functions and quits.

PROGRAM ONE

```
ØØ1ØØ C     PROGRAM ONE--DEMO FCHAIN SUBROUTINE
ØØ1Ø5 C     CALL AND ACTION
ØØ2ØØ C
ØØ3ØØ       PROGRAM ONE
ØØ4ØØ C     SETUP TIME DATE BUFFER
ØØ5ØØ       BYTE TBUF(26)
ØØ6ØØ C     CLEAR SCREEN
ØØ7ØØ       CALL CLS
ØØ8ØØ       CALL TIME(TBUF)
ØØ9ØØ C     DISPLAY THE TIME AND DATE BUFFER  DATA
Ø1ØØØ       WRITE(1,1ØØ) TBUF
Ø11ØØ 1ØØ   FORMAT(1X,26A1)
Ø12ØØ C     DELAY EXECUTION 3 SECONDS
Ø13ØØ       CALL DELAY(3)
Ø14ØØ C     CALL PROGRAM TWO A FORTRAN PROGRAM
Ø15ØØ       CALL FCHAIN('TWO',3)
Ø16ØØ       END
```

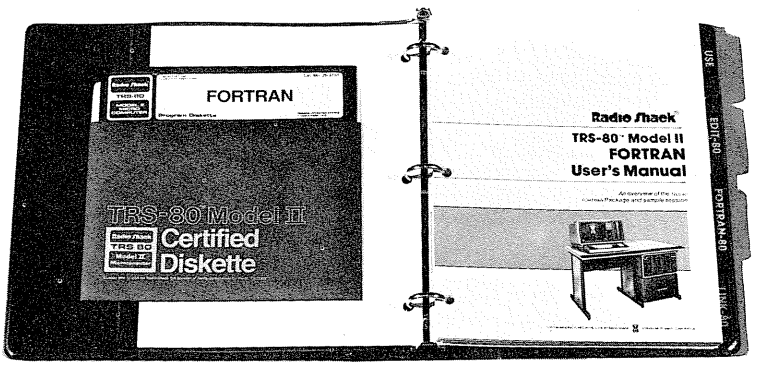
PROGRAM TWO

```
ØØ1ØØ C     PROGRAM TWO
ØØ2ØØ C     CALLED FROM PROGRAM ONE
ØØ2Ø5 C     DEMONSTRATES THE SYSTEM SUBROUTINE
ØØ3ØØ C     CALL AND ACTION
ØØ4ØØ C
ØØ5ØØ       PROGRAM   TWO
ØØ6ØØ C     CLEAR SCREEN
ØØ7ØØ       CALL CLS
ØØ8ØØ C     DISPLAY THE DIRECTORY SYSTEM COMMAND
ØØ9ØØ       CALL SYSTEM('DIR',3)
Ø1ØØØ C     DISPLAY THE ANALYSE SYSTEM COMMAND
Ø11ØØ       CALL SYSTEM('ANALYZE',7)
Ø12ØØ C     RENAME THIS PROGRAM TO THREE
Ø13ØØ       CALL SYSTEM('RENAME TWO TO THREE',19)
Ø14ØØ C     RENAME THREE BACK TO TWO
Ø15ØØ       CALL SYSTEM('RENAME THREE TO TWO',19)
```

```
Ø16ØØ C     LOAD THE BASIC INTERPRETER AND SET
Ø16Ø5 C     FILES AND MEM SIZE
Ø17ØØ       CALL FCHAIN('BASIC -F:3-M:6ØØØØ',18)
Ø18ØØ C     BASIC OVERWRITES US SO WE USE
Ø18Ø5 C     FCHAIN AND WE ARE NOW GONE
Ø19ØØ       END
```

ASSEMBLY LANGUAGE CHAINING SUBROUTINE

```
ØØ1ØØ ;       ASSEMBLY LANGUAGE SUBROUTINES FOR
ØØ1Ø1 ;       USE WITH FORTRAN 8Ø
ØØ11Ø ;       TIME SUBROUTINE FOR FORTRAN
ØØ12Ø ;       PARAMETER IN HL IS ADDRESS OF
ØØ121 ;       TIME/DATE BUFFER
ØØ13Ø ;       PASSED FROM FORTRAN, THE BUFFER IS
ØØ131 ;       FILLED
ØØ14Ø ;       AS 'SATAPR1519818Ø513.2Ø.12Ø45' REF
ØØ141 ;       PAGE 4-68
ØØ15Ø ;       FORTRAN MUST HAVE ASSIGNED 26 BYTES
ØØ151 ;       FOR THE BUFFER
ØØ16Ø         ENTRY   TIME
ØØ17Ø TIME:   LD      B,Ø
ØØ18Ø         LD      A,45
ØØ19Ø         RST     8
ØØ2ØØ         RET
ØØ21Ø ;       SCROLL PROTECT FOR FORTRAN
ØØ22Ø ;       CALLED BY FTN WITH HOW MANY LINES
ØØ23Ø ;       TO PROTECT PASSED AS PARAMETER PASSED
ØØ231 ;       IN HL
ØØ24Ø         ENTRY   SCROLL
ØØ25Ø SCROLL: LD      A,(HL)    ;GET HOWMANY
ØØ26Ø         LD      B,A       ;TO B
ØØ27Ø         LD      A,27
ØØ28Ø         RST     8
ØØ29Ø         RET
ØØ3ØØ ;       DELAY SUBROUTINE FOR FORTRAN
ØØ31Ø ;       PASS AN INTEGER VALUE OF SECONDS TO
ØØ311 ;       DELAY
ØØ32Ø ;       USE A VARIABLE NOT A CONSTANT IN
ØØ321 ;       FTN SINCE THIS VALUE
ØØ33Ø ;       IS CHANGED HERE PARAMETER IS PASSED
ØØ331 ;       IN HL
ØØ34Ø         ENTRY   DELAY
ØØ35Ø DELAY:  LD      E,(HL)    ;SAVE ENTRY VALUE
ØØ36Ø         SLA     (HL)      ;MPY INPUT BY 2
ØØ37Ø BAK:    LD      BC,ØFFFFH
ØØ38Ø         LD      A,6
ØØ39Ø         RST     8
ØØ4ØØ         DEC     (HL)
ØØ41Ø         JR      NZ,BAK
ØØ42Ø         LD      (HL),E    ;REINSTATE ENTRY
ØØ421 ;                          VALUE
ØØ43Ø         RET
ØØ44Ø ;       CLSCROL/CLS
ØØ45Ø ;       CLEAR SCREEN FROM FORTRAN
ØØ46Ø ;       CLEARS AREAS NOT SCROLL PROTECTED
ØØ47Ø ;       ACCOMPLISHED BY HOMING CURSOR
ØØ48Ø ;       TO FIRST NON SCROLL PROTECTED
ØØ49Ø ;       POSITION AND ERASING TO END OF
ØØ491 ;       SCREEN
ØØ5ØØ ;       USES CHARACTERS 14H TO HOME
ØØ51Ø ;       USES CHARACTER 18H ERASE TO END OF
ØØ511 ;       SCREEN
ØØ52Ø ;       CURSOR IS LEFT AT HOME POSITION
ØØ53Ø         ENTRY   CLSCRL
ØØ54Ø CLSCRL: LD      B,14H     ;HOME TO NON
ØØ541 ;                          SCROLL POS
ØØ55Ø         LD      A,8
ØØ56Ø         RST     8
ØØ57Ø         LD      B,18H     ;ERASE TO END OF
ØØ571 ;                          SCREEN
ØØ58Ø         LD      A,8
ØØ59Ø         RST     8
ØØ6ØØ         RET
ØØ61Ø ;       CLS     FULL SCREEN CLEAR CLEARS
ØØ611 ;               SCROLL PROTECTED AREAS
ØØ62Ø ;
ØØ63Ø ;
ØØ64Ø         ENTRY   CLS
ØØ65Ø CLS:    LD      B,1BH     ;FULL SCREEN CLS
ØØ66Ø         LD      A,8
ØØ67Ø         RST     8
ØØ68Ø         RET
ØØ69Ø ;       CURSOF/CURSON
ØØ7ØØ ;       TURN CURSOR OFF
ØØ71Ø ;       NEXT CLS OR CURSON WILL RESTORE IT
ØØ72Ø ;
ØØ73Ø         ENTRY   CURSOF
ØØ74Ø CURSOF: LD      B,Ø
ØØ75Ø         LD      A,26
ØØ76Ø         RST     8
ØØ77Ø         RET
ØØ78Ø ;       CURSON  TURN CURSOR ON
ØØ79Ø ;
ØØ8ØØ         ENTRY   CURSON
ØØ81Ø CURSON: LD      B,1
ØØ82Ø         LD      A,26
ØØ83Ø         RST     8
ØØ84Ø         RET
ØØ85Ø ;       FORTRAN CHAIN BY ASSEMBLY
ØØ86Ø ;       HL--->TRSDOS COMMAND IE FTN PROG
ØØ87Ø ;       DE--->LENGTH OF COMMAND
ØØ88Ø ;       FORTRAN CALL-->CALL('FTRANPRG',8)
ØØ89Ø ;       MAKE THE PARAMETER 8CHARS LONG
ØØ9ØØ         ENTRY   FCHAIN
ØØ91Ø FCHAIN:
ØØ911         EX      DE,HL
ØØ92Ø         LD      B,(HL)
ØØ921         EX      DE,HL
ØØ93Ø         LD      A,37
ØØ94Ø         RST     8
ØØ95Ø         RET
ØØ96Ø ;
ØØ97Ø ;       FORTRAN SYSTEM CALL
ØØ98Ø ;       HL ---->TRSDOS SYSTEM COMMAND LINE
ØØ99Ø ;       DE----->LENGTH OF COMMAND
Ø1ØØØ ;       FORTRAN:
Ø1ØØ1 ;    CALL('RENAME TEXT/OLD TO TEXT/NEW',27)
Ø1Ø1Ø          ENTRY   SYSTEM
Ø1Ø2Ø SYSTEM:
Ø1Ø21          EX      DE,HL
Ø1Ø3Ø          LD      B,(HL)
Ø1Ø31          EX      DE,HL
Ø1Ø4Ø          LD      A,38
Ø1Ø5Ø          RST     8
Ø1Ø6Ø          RET
Ø1Ø7Ø ;
Ø1Ø8Ø          END
Ø1Ø9Ø ;********************************
Ø11ØØ ;       END OF SUBS
Ø111Ø ;********************************
```



ADDSPACE/BAS

Henry C. Brown
2740 North Ferry St.
Anoku, MN 55303-1619

After reading the January Fort Worth Scene, I realized that I had developed a utility program for BASIC that might be of use to you and/or your readers.

The program, ADDSPACE/BAS, adds spaces around key words in a compressed BASIC program. The program to be modified must have been saved in 'A,' ASCII Format.

Make sure that you OPEN two files before going into BASIC. Answer the File Spec Prompt with the name of the program to be changed. Then "New File Spec" which is the name of the new BASIC program with spaces added.

The printer should be on, so if New Program Line exceeds 255 characters that line is not changed. The old line is sent to the printer, so if spaces are still wanted, there is a record of the offending lines.

I chose not to add a space between GOSUB and Line Number. This is handled in Routine "H." To add space, use Routine "C."

AN EXAMPLE USING ADDSPACE

The example below, TEST (in compressed format), is used to demonstrate how ADDSPACE works to uncompress a BASIC program. TEST2 is the uncompressed program that results after ADDSPACE has been run on TEST.

TEST

```
10 ' THIS IS A PROGRAM TO
20 ' DEMONSTRATE THE PROGRAM
30 '       ADDSPACE
40 FORN=1TO5:READN$(N):NEXTN
50 FORX=1TO5:PRINTN$(X):IFN$(X)="END"THEN
     PRINT"END OF DATA"ELSENEXTX
60 DATAJOHN,MARY,SHIRLEY,HANK,END
```

TEST2

```
10 ' THIS IS A PROGRAM TO
20 ' DEMONSTRATE THE PROGRAM
30 '       ADDSPACE
40 FOR N=1 TO 5:READ N$(N):NEXT N
50 FOR X=1 TO 5:PRINT N$(X):IF N$(X)="END" THEN
     PRINT"END OF DATA" ELSE NEXT X
60 DATA JOHN,MARY,SHIRLEY,HANK,END
```

For those of us who, for whatever reason, have to uncompress lengthy BASIC programs to display them in a more readable format, ADDSPACE can be a very useful program.

```
1 ' ADDSPACE/BAS IS PUBLIC DOMAIN -CREATED BY
     HENRY C. BROM 12/31/1981
2 ' PROGRAM ADDS SPACES AROUND KEY WORDS (I.E.
     UNCOMPRESSES PROGRAMS)
3 ' IF SPACES CREATE A LINE LONGER THAN 255
     CHARACTERS THAT LINE NOT CHANGED.
4 '
10 CLS
     : CLEAR 1000
     : DEFINT A-Z
     : ON ERROR GOTO 550
20 DEF FNRW%(A1$, A2$, A3%)=(INSTR(A1$,
     LEFT$(A2$+STRING$(A3%, "^"), A3%))-1)/A3%+1
30 '
100 PRINT "ADD SPACES AROUND KEY WORDS"
110 INPUT "FILE SPEC "; PROG$
     : INPUT "NEW FILE SPEC "; N1$
     : IF PROG$=N1$ THEN 10
120 OPEN "I", 1, PROG$
     : OPEN "O", 2, N1$
130 ' KEYWORDS & LINE CODES FOR HANDLING EACH
140 R6$="RETURN^RESUME^DEFINT^DEFSGN^DEFDBL
     ^DEFSTR^"
150 '      A,      A,      A,      A,      A,      A
160 R5$="PRINT^INPUT^GOSUB^FIELD^CLOSE^ERROR
     ^CLEAR^"
170 '      B,      B,      H,      A,      A,      A,      A
180 R4$="THEN^ELSE^READ^DATA^RSET^LSET
     ^SWAP^NEXT^STEP^KILL^"
190 '      C,    C,    A,    B,    A,    A,    A,    C,
     C,    B
200 R3$="FOR^AND^NOT^PUT^GET^DIM^DEF^LET^"
210 '      A,  C,  C,  A,  A,  A,  D,  A
220 R2$="IF^OR^TO^ON^AS^"
230 '      A, E, F, A, G
240 '
250 IF EOF(1) THEN 550 ELSE LINE INPUT #1, L$
260 N=LEN(L$)
     : N$=""
     : F4=0
     : F1=0  'QUOTE & REM FLAGS
270 FOR P=1 TO N
280  C$=MID$(L$, P, 1)
     : N$=N$+C$
290  IF C$=CHR$(34) AND F4=1 THEN F4=0
     : GOTO 310
300  IF C$=CHR$(34) AND F4=0 THEN F4=1
310  IF C$="'" AND F4=0 THEN F1=1 'START OF
     REMARK
320  IF F4=1 OR F1=1 THEN 450 'SKIP QUOTED
     STRINGS & "'" REM STATEMENTS
330  IF C$<"A" THEN 450 ' ALL KEYWORDS ALPHA
340  ' LOOK FOR THE LONGEST KEY WORDS FIRST
350  L=6
     : R%=FNRW%(R6$, RIGHT$(N$, L), L+1)
360  ON R% GOTO 1010, 1010, 1010, 1010, 1010,
     1010
370  L=L-1
     : R%=FNRW%(R5$, RIGHT$(N$, L), L+1)
380  ON R% GOTO 2010, 2010, 8010, 1010, 1010,
     1010, 1010
390  L=L-1
     : R%=FNRW%(R4$, RIGHT$(N$, L), L+1)
400  ON R% GOTO 3010, 3010, 1010, 2010, 1010,
     1010, 1010, 3010, 3010, 2010
410  L=L-1
     : R%=FNRW%(R3$, RIGHT$(N$, L), L+1)
420  ON R% GOTO 1010, 3010, 3010, 1010, 1010,
     1010, 4010, 1010
430   L=L-1
     : R%=FNRW%(R2$, RIGHT$(N$, L), L+1)
440   ON R% GOTO 1010, 5010, 6010, 1010, 7010
450 NEXT
500 '===========
510 N2=LEN(N$)
     : IF N2>255 THEN PRINT"NEW LINE TOO LONG
     CAN'T CHANGE"
     : PRINT L$
     : LPRINT L$
     : N$=L$
520 PRINT #2, N$
530 IF N2<>N THEN PRINT L$; "<--old"
     : PRINT N$
540 GOTO 250
550 CLOSE
     : PRINT "CLOSE", PROG$, N1$
560 END
1000 ' A ================================
1010 IF P=N THEN 450 ELSE X$=MID$(L$, P+1, 1)
     'LOOK AT NEXT CHAR
```

```
1Ø2Ø IF X$=" " OR X$=":" THEN 45Ø ELSE N$=N$+" "
     : GOTO 45Ø 'ADD SPACE
2ØØØ ' B =============
2Ø1Ø X$=MID$(L$, P+1, 1)
2Ø2Ø IF X$="@" OR X$=CHR$(34) OR X$=":" OR X$=" "
     THEN 45Ø ELSE N$=N$+" "
     : GOTO 45Ø
3ØØØ ' C ==================
3Ø1Ø X$=MID$(N$, LEN(N$)-L, 1)
3Ø2Ø IF X$=" " OR X$=":" THEN 1Ø1Ø ELSE
     T2$=LEFT$(N$, LEN(N$)-L)
     : T3$=RIGHT$(N$, L)
3Ø3Ø N$=T2$+" "+T3$
     : GOTO 1Ø1Ø
4ØØØ ' D =============
4Ø1Ø X$=MID$(L$, P+1, 3)
4Ø2Ø IF X$="INT" OR X$="SGN" OR X$="DBL" OR
     X$="STR" THEN 45Ø ELSE 1Ø1Ø
5ØØØ ' E =============
5Ø1Ø X$=MID$(L$, P-2, 3)
5Ø2Ø IF X$="XOR" THEN L=3
     : GOTO 3Ø1Ø ELSE GOTO 3Ø1Ø
6ØØØ ' F =============
6Ø1Ø X$=MID$(L$, P-3, 4)
6Ø2Ø IF X$="GOTO" THEN L=4
     : GOTO 3Ø1Ø ELSE GOTO 3Ø1Ø 'CHECK FOR SPACE
7ØØØ ' G =============
7Ø1Ø X$=MID$(L$, P+1, 1)
7Ø2Ø IF X$="C" OR X$="$" THEN 45Ø ELSE 3Ø1Ø
8ØØØ ' H ==================
8Ø1Ø X$=MID$(N$, LEN(N$)-L, 1)
8Ø2Ø IF X$=" " OR X$=":" THEN 45Ø ELSE
     T2$=LEFT$(N$, LEN(N$)-L)
     : T3$=RIGHT$(N$, L)
8Ø3Ø N$=T2$+" "+T3$
     : GOTO 45Ø
9999 '************** END *****************
```

Model I/III Bugs (From Page 30)

```
ØØ1ØØ IDENTIFICATION DIVISION.
ØØ11Ø PROGRAM-ID.
ØØ12Ø      TEST-ISAM.
ØØ13Ø
ØØ14Ø ENVIRONMENT DIVISION.
ØØ15Ø CONFIGURATION SECTION.
ØØ16Ø SOURCE-COMPUTER.          RMC.
ØØ17Ø OBJECT-COMPUTER.          RMC.
ØØ18Ø INPUT-OUTPUT SECTION.
ØØ19Ø FILE-CONTROL.
ØØ2ØØ      SELECT TST-FIL ASSIGN TO RANDOM, TSTFIL-NAME;
ØØ21Ø           ORGANIZATION IS INDEXED;
ØØ22Ø           ACCESS IS RANDOM;
ØØ23Ø           RECORD KEY IS TEST-KEY-VAL.
ØØ24Ø
ØØ25Ø DATA DIVISION.
ØØ26Ø FILE SECTION.
ØØ27Ø FD   TST-FIL
ØØ28Ø      RECORD CONTAINS 256 CHARACTERS
ØØ29Ø      LABEL RECORD IS STANDARD.
ØØ3ØØ Ø1   TST-FIL-REC.
ØØ31Ø      Ø5   TEST-KEY-VAL        PIC X(1Ø).
ØØ32Ø      Ø5   SOME-SPACES         PIC X(246).
ØØ33Ø WORKING-STORAGE SECTION.
ØØ34Ø 77   TSTFIL-NAME              PIC X(6) VALUE "TSTFIL".
ØØ35Ø 77   END-VAL                  PIC X(3) VALUE "NO ".
ØØ36Ø 77   USER-INPUT-KEY           PIC X(1Ø).
ØØ37Ø Ø1   MESSAGE-1.
ØØ38Ø      Ø2   FILLER              PIC X(28)
ØØ39Ø             VALUE "INPUT 1Ø IDENTICAL LETTERS: ".
ØØ4ØØ
ØØ41Ø PROCEDURE DIVISION.
ØØ42Ø
ØØ43Ø FOR-OPENERS-PARAGRAPH.
ØØ44Ø      OPEN OUTPUT TST-FIL.
ØØ45Ø      MOVE SPACES TO SOME-SPACES.
ØØ46Ø
ØØ47Ø DRIVER-PARAGRAPH.
ØØ48Ø      PERFORM INPUT-PARAGRAPH.
ØØ49Ø      IF END-VAL = "YES" GO TO END-RUN.
ØØ5ØØ      PERFORM FILE-WRITE-PARAGRAPH.
ØØ51Ø      GO TO DRIVER-PARAGRAPH.
ØØ52Ø
ØØ53Ø********** NOTE USE OF "TAB" BELOW **********
ØØ54Ø
ØØ55Ø INPUT-PARAGRAPH.
ØØ56Ø      DISPLAY MESSAGE-1, LINE 2, ERASE.
ØØ57Ø      ACCEPT USER-INPUT-KEY, POSITION Ø, PROMPT,
ØØ58Ø           ECHO, TAB.
ØØ59Ø      IF USER-INPUT-KEY = "ZZZZZZZZZZ"
ØØ6ØØ           MOVE "YES" TO END-VAL.
ØØ61Ø
ØØ62Ø FILE-WRITE-PARAGRAPH
ØØ63Ø      MOVE USER-INPUT-KEY TO TEST-KEY-VAL.
ØØ64Ø      WRITE TST-FIL-REC.
ØØ65Ø
ØØ66Ø END-RUN.
ØØ67Ø      DISPLAY "SUCCESSFUL END-OF-PROGRAM.".
ØØ68Ø      CLOSE TST-FIL.
ØØ69Ø      STOP RUN.
```

Run this program without the PATCH below applied. Type in "AAAAAAAAAA" when prompted. The program should halt abnormally when the user-response-field is filed with "A's," due to lines 570-580 of the above program.

Now apply the PATCH and run this program again. Type in "AAAAAAAAAA", then "BBBBBBBBBB", "CCCCCCCCCC", etc. The program should continue to prompt after each string of 10 characters is typed in. Halt the program when desired by typing "ZZZZZZZZZZ".

THE PATCH:

At TRSDOS READY type in the following line:

```
PATCH RUNCOBOL/CMD (ADD=AE61,FIND=C3F6A9,CHG=AFC9ØØ)
```

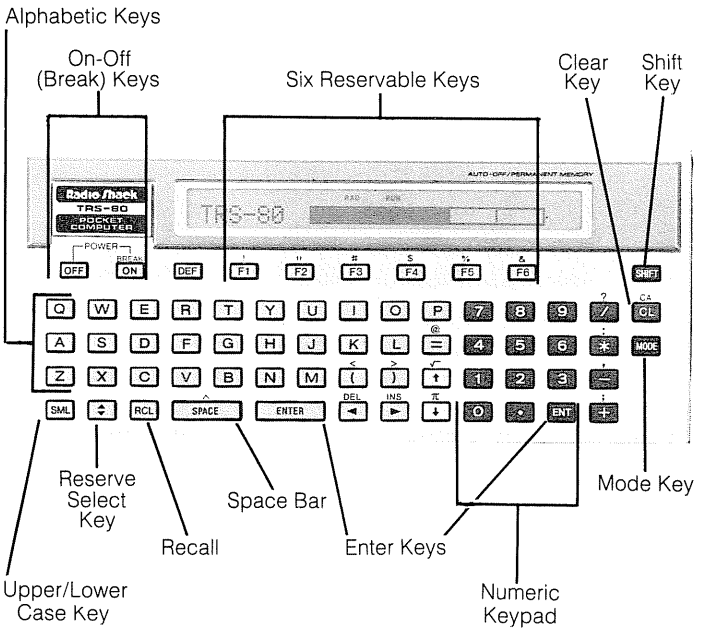
Pocket Computer Model PC-2

THE HIGHLIGHTS

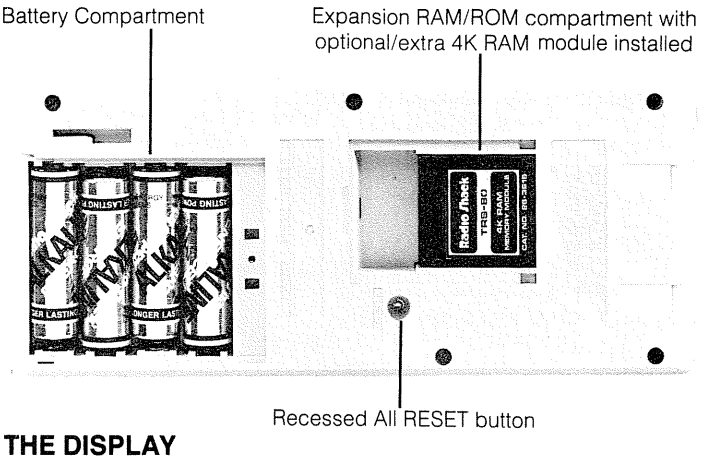
- Eight-bit custom microprocessor operating at a clock speed of 1.3 MHz.
- Powerful BASIC Interpreter with 42 Statements, 34 Functions and 6 Commands
- FULL String handling ability with up to 80 characters per string
- Two Dimension Arrays
- 26 Character Alphanumeric Display with Upper and Lower case characters
- Fully addressable 7 x 156 Dot Matrix LCD Graphics
- Audio tone programmable for number of repetitions, tone and duration
- 2640 Byte Memory, expandable with plug-in RAM or ROM modules
- Built-In Real Time Clock
- 60-pin Input/Output Interface Connector

DETAILS

There are so many new and interesting features on the PC-2 that it is difficult to know where to get started. Perhaps the best place to start is by showing you the machine and pointing out some of the keyboard and case highlights:

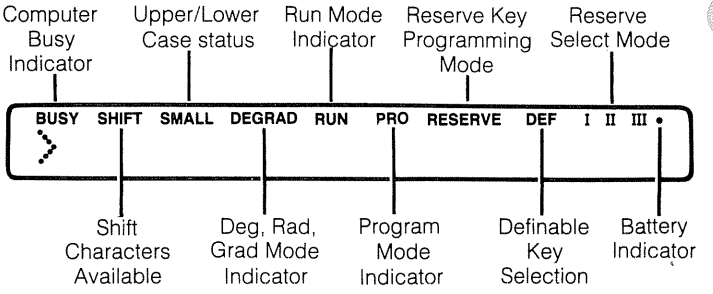


In addition to the highlighted features, PC-2 has a 60-pin I/O connector on one end, and also a connector for an optional/extra power supply. Further, when PC-2 is connected to the PC-2 4-color Printer/Plotter, the power for PC-2 comes from the printer's power source (battery or AC). This helps to extend the life of the PC-2's internal batteries.



THE DISPLAY

The PC-2 display provides a wealth of information about the current status of the computer, as you can see from the labeled drawing of the LCD display:



Let's look at some of the features we have pointed out on the photos and drawings:

SHIFT Key — The ⟨**SHIFT**⟩ key is used to select the characters or symbols which appear above the non-alphabetic keys. An example is the ^ which is above the Space Bar. Pressing ⟨**SHIFT**⟩⟨**SPACE BAR**⟩ will give you the caret (^) not a space. The shift key is used with the alphabetic keys to produce upper or lower case characters. The "normal" keyboard mode is to have upper case characters when you press an alphabetic key, and lower case characters when you press ⟨**SHIFT**⟩ and then an alphabetic key. You use the ⟨**SML**⟩ key to switch this to unshifted lower case and shift for upper case. The LCD indicator "SMALL" is displayed when the keyboard is in the shift for upper case mode.

The ⟨**MODE**⟩ key normally toggles the computer (switches the computer back and forth) between the RUN and PROgram modes. The RUN mode is used when you want to execute a program which is in memory, or when you want to use the PC-2 as a sophisticated calculator. The PROgram mode is used to enter your BASIC programs.

Using ⟨**SHIFT**⟩⟨**MODE**⟩ puts you into the RESERVE key programming mode. Here you can program the special function keys (⟨**F1**⟩-⟨**F6**⟩).

The Numeric Keypad is laid out to facilitate arithmetic operations. The easy availability of the arithmetic operator keys (/ for division, * for multiplication, - for subtraction and + for addition) make numeric data entry very easy. We provided two Enter Keys, a large ⟨**ENTER**⟩ key under the alphabetic keys, and a smaller ⟨**ENT**⟩ key with the numeric keypad, to make data entry as convenient as possible.

There are six Reservable function keys available at the top of the PC-2 keyboard. Each key can be programmed in three separate reserve key programming modes, giving you a total of 18 programmable functions. The mode you are in is indicated by a Roman I, II or III on the LCD display. You switch between these modes by using the ⟨ ▲▼ ⟩ key. In addition to providing you with 18 possible reserved functions, PC-2 also gives you the ability to create three reserve key menus (one for each of the modes). These menus can be used to remind you of the function reserved for each key in each of the possible modes.

The ⟨**RCL**⟩ or recall key lets you toggle the display between the current reserve key menu and the most recently displayed information. If you had entered 5+5 from the keyboard and then wanted to check the function keys, press ⟨**RCL**⟩ to see the reserve menu, and ⟨**RCL**⟩ to return to the display which had 5+5.

The ⟨**DEF**⟩ key is used to select previously defined sections in your program, or to select one of the 10 permanently DEFined functions. The alphabetic keys Q, W, E, R, T, Y, U, I, O and P have been permanently defined to give you two key entry of the following commonly used commands: INPUT, PRINT, USING, GOTO, GOSUB, RETURN, CSAVE, CLOAD, MERGE and LIST. Pressing ⟨**DEF**⟩ and one of these keys will enter and display the appropriate command word. Note: if you do not have the optional printer/cassette interface attached, the CSAVE, CLOAD and MERGE commands (as well as all other commands peculiar to cassette or printer operations) will be displayed as a ˜ to indicate that this function is not available. The ⟨**DEF**⟩ key is used with the remaining alphabetic keys to give you quick access to labeled portions of BASIC programs. This gives you the ability to have more than one BASIC program in memory at the same time.

The LCD Busy indicator tells you when the computer is actively running a program or other operation. This can be valuable when the PC-2 is executing a lengthy program or operation.

PC-2 can be set to do angle/trigonometry calculations in degrees, radians, or grads. Easy to use commands are used to select the appropriate system.

PC-2 will accept a single RAM, ROM or combination RAM/ROM expansion module. These modules can be used to simply increase the amount of memory which is available to the user, or ROM packs can be designed for particular applications. Radio Shack currently offers an optional/extra 4K RAM expansion module.

The deeply recessed All RESET button is available should you ever need to do a total system reset.

VARIABLES

PC-2 variable names can be one or two characters. The name can be a single letter (A-Z) or it can be a letter followed by either another letter or a single digit (0-9).

Variables can be one of two types, numeric for storing numbers, or string for storing characters. A and A$ are two separate variables, A for numbers and A$ for strings. Both A and A$ can be used in the same program.

Unless otherwise stated by use of the DIM command, string variables can hold up to 16 characters. Use of the DIM statement can reduce a string's storage capacity to a single character, or expand the capacity to 80 characters.

Variables retain their most recently assigned values until:

1. A new value is assigned.
2. A NEW or CLEAR command is given.
3. A program is run using the RUN command.
4. The computer's batteries are changed.

Variables retain their values even with PC-2 turned off.

Although all variables of the same type (numeric or string) are utilized in the same manner, they are not treated the same by PC-2. PC-2 includes a "fixed memory" area with enough storage space for 26 numeric variables (A-Z) and 26 string variables (A$-Z$) with a string size of up to 16 characters. As a result, variables with these names are always stored in this area.

All other variables, including those with two character names and DIMensioned variables, are allocated space within main memory. This means that main memory is shared by programs and variables, with the program building from the bottom of memory toward the top, and variables building from the top of available memory toward the bottom. It is possible for program and variable storage to overlap, resulting in an error code in the range from 177 to 181.

During normal operations, fixed variable memory is cleared or reset only by an explicit CLEAR command. Variables in the main memory area are cleared by either CLEAR or RUN. Both memory areas are cleared when a NEW command is issued.

ARRAYS

An array is simply a group of consecutive storage areas, or "locations," with a single name. Each storage area can hold a single number or character string. In an array, all storage areas are either numeric or string.

All variables in fixed memory may be used as elements in a numeric or string array, as appropriate for the variable type. To use fixed memory variables in this way, a DIMension statement is not needed. The numeric fixed array is @ and the string array is @$. Thus, the designation @(1) represents the same storage location as the numeric variable A, and @(26) is the same location as the numeric variable Z. The designation @$(5) refers to the same variable storage location as E$, and @$(20) is the same as T$. The smallest subscript in the @ or @$ array is 1, and the largest is 26.

To define arrays other than @ and @$, or if you need a two dimension array, the DIM statement (short for dimension) is used. Arrays must always be "declared" (dimensioned) before they can be used.

A numeric array can be DIMensioned with one or two dimensions. Element values for each dimension are values from 0 (for a single element array) to 255 (for an array with 256 elements). Obviously, the larger you dimension an array, the more memory it will require for storage. For-

example, an array with 25 rows and 35 columns requires 875 storage locations! If you have a variable called X, and an array called X these are separate and distinct variable storage areas.

String arrays can also be one or two dimensional with up to 255 elements in each dimension. In addition to specifying the size of the array, you may also, optionally, define the maximum string length for each element. Allowable string lengths range from 1 to 80. If string length is not specified, the default value of 16 is used.

POCKET BASIC

The following information will give you only a few of the special features of PC-2 BASIC:

CONTROLLING THE DISPLAY

The PC-2 has two ways of using the LCD display — text and graphics. Both text and graphics can be displayed at the same time on the LCD. Some of PC-2's display commands include:

CURSOR — When used with a parameter, CURSOR positions the cursor to a specified text position (0-25). When used with no parameter, CURSOR cancels the previously specified position.

USING — USING allows PAUSE or PRINT information to be formatted before being displayed. The format of a USING statement will be used for all PRINT or PAUSE statements following it until a new USING statement is found. A single statement can have multiple USING formats. The PC-2 USING edit characters are:

- # — Numeric field
- * — Fill spaces in numeric fields with asterisks
- . — Decimal point in numeric fields
- , — Comma insertion every three digits to the left of the decimal point in numeric fields
- ^ — Scientific notation
- + — Force printing of sign
- & — Specifies a character field

WAIT — The WAIT command is used to determine how long information PRINTed on the LCD will remain. WAIT by itself sets the display time to "infinity." If WAIT is used with a parameter, the WAIT parameter value can range from 0 (no delay, flash the information as fast as possible) to 65535 (hold the information on the display for approximately 17 minutes.) WAIT 64 will delay for about one second, and WAIT 3840 about one minute. The WAIT command has no affect on the PAUSE statement.

GCURSOR — Positions the cursor to one of the 156 7-dot columns. This command is usually used with GPRINT to place graphic information at a specific column on the display. GCURSOR can also be used with PRINT to create special effects with non-graphics information.

GPRINT — The GPRINT statement provides direct, programmable control over the individual dots in the LCD. The parameters used with GPRINT may be decimal values (0-127), hexadecimal (hex) values (&00-&7F) or a string containing hex digits. In a hex value the first digit (0-7) represents the pattern of the 3 bottom dots in the column and the second hex digit (0-F) represents the pattern of the top four dots. More than one value may be specified in a single GPRINT statement. If decimal or hex values are used, each value is separated from other values by a comma or a semicolon. The use of a comma between values will result in a blank column being printed between the columns represented by the given values. A semicolon will print the information without an intervening blank column.

POINT — The POINT command is used to test the dot pattern in a particular column. The value returned ranges from 0 for a column with no dots set to 127 for a column with all dots set.

LOGICAL OPERATIONS

In addition to the relational operators provided by ⟨, ⟩, = and combinations of these characters, PC-2 also provides the AND, NOT and OR logical operators.

STRING FUNCTIONS

PC-2 has a full complement of string functions, which operate in the same manner as the corresponding commands in Model I, II, III or Color BASICs. The PC-2 String functions are: ASC, CHR$, INKEY$, LEFT$, LEN, MID$, RIGHT$, STR$ and VAL.

CONTROL STATEMENTS AND COMMANDS

AREAD — The AREAD statement allows the user to enter a single value into a program using ⟨**DEF**⟩ and a labeled section of the program. INPUT is not needed to get the value into the program. The AREAD statement must be the first statmeent in the labeled program segment.

ARUN — The ARUN statement is used to automatically begin executing a program as soon as PC-2 is turned on. ARUN must be the first statement in the program, and PC-2 must have been in the RUN mode when the computer was turned off.

BEEP — The BEEP statement has two forms. The first form is used to turn the beep or sound function on or off. If BEEP OFF has been executed, all BEEP commands will be ignored until a BEEP ON command is received. One use of the BEEP OFF statement is to turn off the sound during cassette operations.

The second form of the BEEP statement is used to create tones which can be used for game playing, error signaling and other applications. In this form of the BEEP statement, there is one required parameter and two optional parameters. The required parameter tells PC-2 how many times to beep. Acceptable values range from 0 (for no beep) to 65535. The second parameter, if used, specifies the frequency of the tone. Acceptable values range from 0 (about 7KHz) to 255 (about 230Hz). The third parameter specifies the duration of each tone. Acceptable values for duration range from 0 (no sound) to 65279 (very long).

LOCK — The LOCK statement is used to lock PC-2 into the current operating mode (RUN, PROgram or RESERVE). Locking the computer into a single mode prevents the operating mode from being changed by accidentally hitting the ⟨**MODE**⟩ key.

STATUS — STATUS is used to tell you the current status of PC-2 memory:

STATUS 0 — number of program steps available
STATUS 1 — number of program steps used

STATUS 2 — Address +1 of the end of the current program

STATUS 3 — Address −1 of the beginning of variable storage

STATUS 4-255 — program line number being executed when execution was halted

UNLOCK — UNLOCK is used to unlock PC-2 after a LOCK command has been issued. Once UNLOCK has been executed, the computer's operating mode can again be changed by using the ⟨**MODE**⟩ key.

MATHEMATICAL FUNCTIONS

The mathematical functions which are available in PC-2 are: ABS, ACS, ASN, ATN, COS, DEG, DMS, EXP, INT, LOG, LN, PI, RANDOM, RND, SGN, SIN, SQR and TAN.

SPECIAL FEATURES

TIME — PC-2 has an internal, real-time clock. The TIME statement lets you set or read the current date and time.

To read the time, use TIME without a parameter. The result will be in the form of MMDDHH.mmss, where MM is a one or two digit value (1-12) for the current month, DD is two digits for the current day (01-31), HH is a two digit 24 hour value for the current hour (00-23). In the decimal portion, mm is a two digit number for minutes (00-59) and ss is two digits for seconds (00-59).

To set the current date and time, enter TIME and a numeric value which represents the current date and time information. The format is the same as the value given when reading date and time.

ERROR MESSAGES

PC-2 has more than 39 error messages for normal operations. When an error occurs, a message such as ERROR 28 or ERROR 32 IN 110 will appear. The PC-2 owners manual has a complete listing of the meaning of each of these error messages.

ABBREVIATIONS

Most commands, statements and functions can be abbreviated to one, two or three characters for easy program and command entry. Use of abbreviations does not reduce the memory required to store your program.

MULTIPLE STATEMENT LINES

PC-2 permits multiple program statements to be placed in a single line. Each of the statements in the line is separated from the others by a colon (:). One advantage of multiple statement lines is that they help reduce the amount of memory required for program storage by reducing the number of line numbers in the program (it takes less memory to store a colon, than to store a line number.)

READ, DATA and RESTORE

PC-2 implements BASIC's READ, DATA and RESTORE commands. Unlike other TRS-80s, the PC-2 RESTORE command can be used to restore to a specific DATA line number as well as the more usual restore to top of data.

COMPUTED PROGRAM TRANSFER

PC-2 supports ON . . . GOTO, ON . . . GOSUB and ON ERROR GOTO to make programming easier. The ON ERROR GOTO command is used by the programmer to control error handling rather than have the computer stop program execution.

PLOTTER/PRINTER/CASSETTE INTERFACE

The Printer/Plotter/Cassette Interface is such a dynamic device that we will discuss it in the June Microcomputer News graphics issue.

LANGUAGE COMPARISON — PC-1 TO PC-2

Instructions common to PC-1 and PC-2

Functions

ABS	DEG	LOG	SIN
ACS	DMS	LN	√‾ (Square Root)
ASN	EXP	PI (π)	TAN
ATN	INT	SGN	∧ (Exponentiation)
COS			

Statements

AREAD	END	GRAD	MEM	RADIAN	THEN
BEEP	FOR-TO-STEP	IF	NEXT	REM	USING
CLEAR	GOSUB	INPUT	PAUSE	RETURN	
DEGREE	GOTO	LET	PRINT	STOP	

Commands

CONT	LIST	NEW	RUN

Cassette Commands (Available With 4 Color Plotter/Printer/ Cassette Interface 26-3605)

CHAIN	CLOAD	CLOAD?	CSAVE	INPUT#	PRINT#

Commands Unique to PC-2

Functions

AND	LEFT$	OR	STATUS
ASC	LEN	POINT	STR$
CHR$	MID$	RIGHT$	TIME
INKEY$	NOT	RND	VAL

Statements

ARUN	GCURSOR	LOCK	POINT	TROFF
BEEP (Expanded)	GPRINT	ON ERROR	RANDOM	TRON
CLS	DATA	ON GOSUB	READ	UNLOCK
CURSOR	DIM	ON GOTO	RESTORE	WAIT

Cassette Instructions (Available with 4 Color Plotter/Printer/ Cassette Interface 26-3605)

MERGE	RMT OFF	RMT ON

Printer Instructions (Available with 4 Color Plotter/Printer/ Cassette Interface 26-3605)

COLOR	GRAPH	LINE	RLINE	TAB
CSIZE	LCURSOR	LLIST	ROTATE	TEST
GLCURSOR	LF	LPRINT	SORGN	TEXT

SPEED COMPARISON

The following PC-2 program (modified as needed for each machine) was used on the PC-1, PC-2 and a Model III to get some feeling for the speed differences between the machines.

```
10 A=TIME
20 WAIT 0
   : REM PC-2 ONLY
30 FOR I=1 TO 1000
40 PRINT I; I^2; I^3
50 NEXT I
60 B= TIME
70 PRINT A, B
```

PC-1 — Approx. 3 mins. 50 seconds
PC-2 — 44 seconds
Model III — 14 seconds

Extended Color BASIC Graphics and Color BASIC Sorts

This month, I thought it would be interesting to generate some pleasant designs using the graphics functions available in the various Color Computers that are in use out there.

Originally, I had intended to use the LINE command to generate various designs on the video. By using the design characteristics of the television, you will be able to create multiple symmetrical patterns on the screen. Because these programs require the use of Extended BASIC, I have also included a couple of "sorting" programs that should work on all of the machines (since they do not require Extended BASIC). These will also offer some dazzling graphic displays. So, on with the show. . .

FOR EXTENDED COLOR BASIC OWNERS

Before ENTERing and RUNning each of the following graphics programs, you need to type in and ENTER the following command:

```
PCLEAR 4 <ENTER>
```

If you have just powered up your Extended BASIC machine, this command has already been executed. If not, do it as a direct statement (without a program line number). This will set the stack pointer to the first available RAM location in which your program can be stored.

Now here's the first program:

```
1Ø CLS
2Ø PMODE 4,1
3Ø PCLS
4Ø SCREEN 1,1
5Ø COLOR 1,3
6Ø A=Ø
     : B=Ø
     : X=255
     : Y=192
7Ø FOR COUNTER = Ø TO 255 STEP 3
8Ø LINE (A,B)-(X,Y), PSET
9Ø A=A+3
     : X=X-3
1ØØ NEXT COUNTER
11Ø A=Ø
     : B=Ø
     : X=255
     : Y=192
12Ø FOR COUNTER = Ø TO 192 STEP 3
13Ø LINE (X,B)-(A,Y), PSET
14Ø B=B+3
     : Y=Y-3
15Ø NEXT COUNTER
999 GOTO 999
```

The next program uses the same idea and adds circles to the graphics. Here it is:

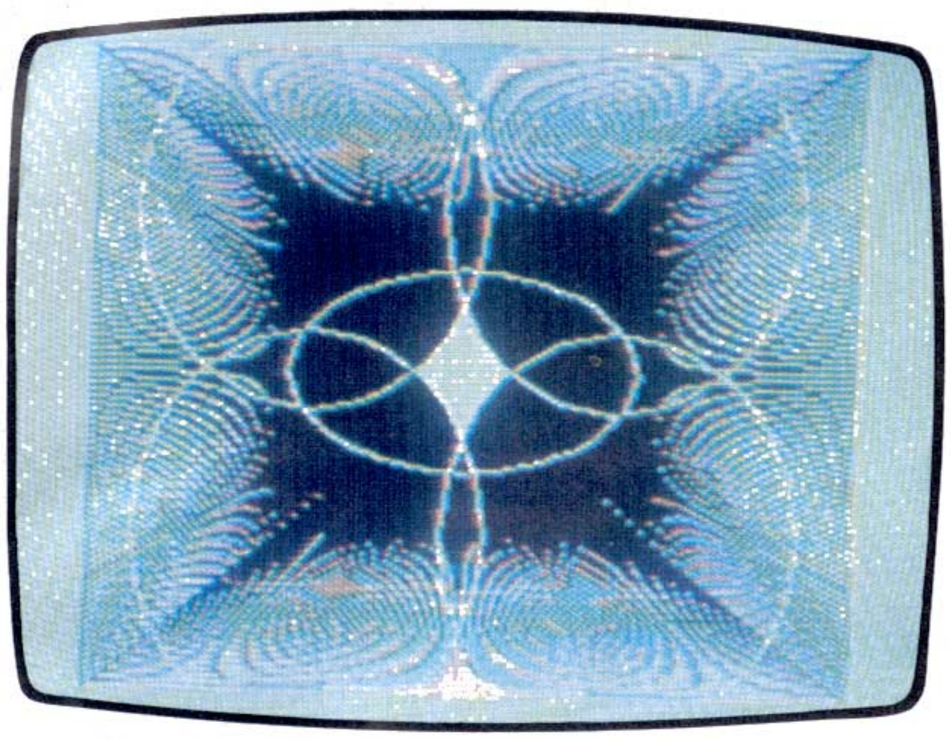
```
1Ø PMODE 4,1
     : COLOR 4,1
     : PCLS
     : SCREEN 1,1
2Ø FOR X=Ø TO 255 STEP 2
     : LINE (X,)-(255-X,192),PSET
     : NEXT X
     : FOR Y=Ø TO 192 STEP 2
     : LINE(255,Y)-(Ø,192-Y),PSET
     : NEXT Y
3Ø COLOR 1,4
     : FOR X=76 TO 18Ø STEP 1Ø4
     : FOR Y=48 TO 144 STEP 96
     : CIRCLE(X,Y),6Ø,1
     : NEXT Y,X
     : CIRCLE(128,96),6Ø,1,.5
4Ø PAINT(128,96),1,1
     : PAINT(76,48),1,1
     : PAINT(18Ø,48),1,1
     : PAINT(76,144),1,1
     : PAINT(18Ø,144),1,1
     : PAINT(1Ø4,96),1,1
     : PAINT(152,96),1,1
     : PAINT(128,72),1,1
     : PAINT(128,12Ø),1,1
5Ø SCREEN 1,Ø
999 GOTO 999
```

If you liked that one, then try this one on for size:

```
1Ø PMODE 4,1
     : COLOR 4,1
     : PCLS
     : SCREEN 1,1
15 FOR NUMBER = 7 TO 2 STEP -1
2Ø FOR X=Ø TO 254 STEP NUMBER
     : LINE(X,Ø)-(128,96),PSET
     : LINE(X+1,Ø)-(128,96),PRESET
     : NEXT X
3Ø FOR Y=Ø TO 191 STEP NU-1
     : LINE(255,Y)-(128,96),PSET
     : LINE(255,Y+1)-(128,96),PRESET
     : NEXT Y
4Ø FOR X=255 TO 1 STEP -NUMBER
     : LINE(X,192)-(128,96),PSET
     : LINE(X-1,192)-(128,96),PRESET
     : NEXT X
5Ø FOR Y=192 TO 1 STEP -NU+1
     : LINE(Ø,Y)-(128,96),PSET
     : LINE(Ø,Y-1)-(128,96),PRESET
     : NEXT Y
6Ø NEXT NUMBER
999 GOTO 15
```

And if that's not enough, here's another one to try:

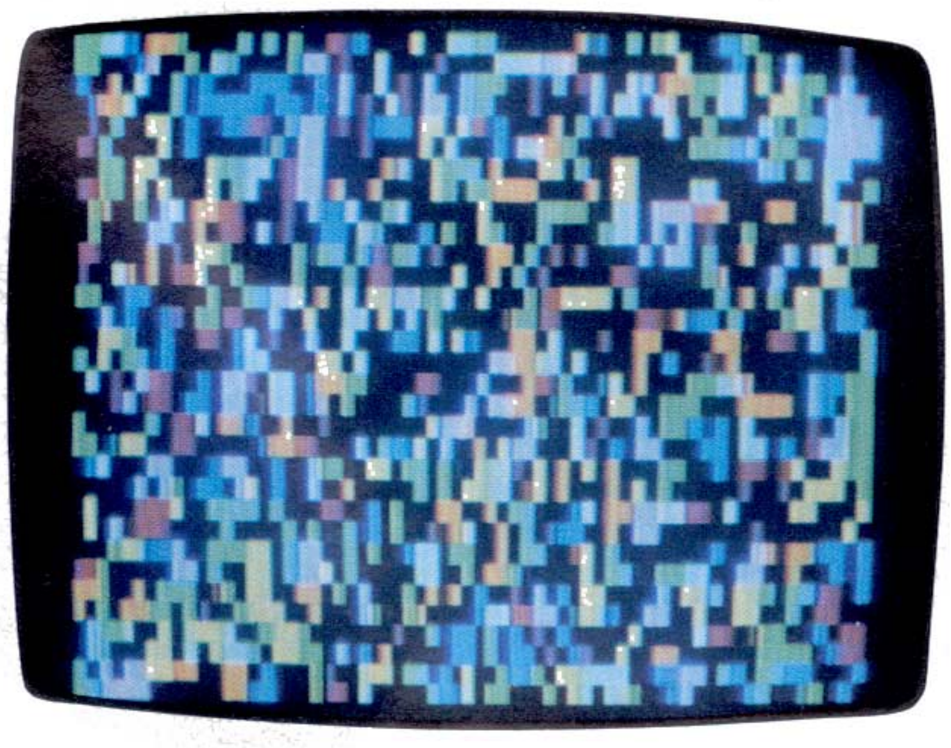
```
1Ø PMODE 4,1: COLOR 4,1
     : PCLS
     : SCREEN 1,1
15 FOR NUMBER = 8 TO 3 STEP -1
2Ø FOR X=Ø TO 254 STEP NU
     : LINE(X,Ø)-(Ø,192),PSET
     : LINE(X+1,Ø)-(Ø,192),PRESET
     : NEXT X
3Ø FOR Y=Ø TO 191 STEP NU-1
     : LINE(255,Y)-(Ø,192),PSET
     : LINE(255,Y+1)-(Ø,192),PRESET
     : NEXT Y
4Ø NEXT NUMBER
999 GOTO 15
```



ARE YOUR GRAPHICS OUT OF SORTS?

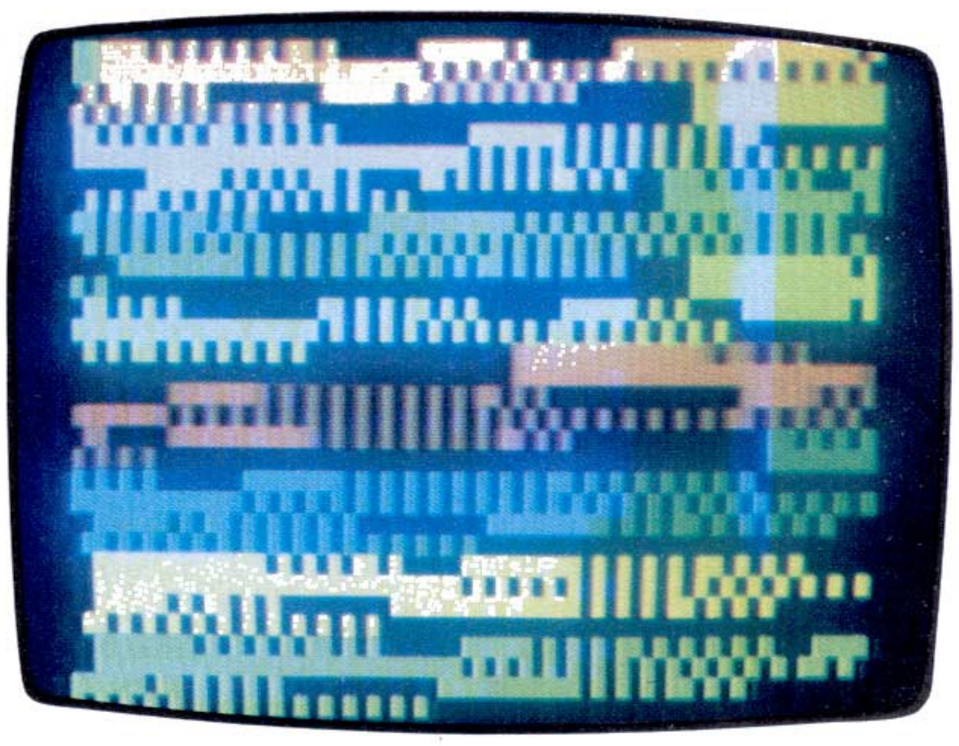
Now, for those of you who do not have Extended BASIC, I worked up a couple sort routines which generate random graphic characters, and in one case fills the screen with them, and then sorts them while you watch. Here are the routines:

```
1Ø CLS
2Ø INPUT "START";S
3Ø INPUT "END";E
4Ø INPUT "NUMBER";N
45 PRINT
     : PRINT"PRESS ENTER TO BEGIN"
46 A$=INKEY$
     : IF A$="" THEN A=RND(1ØØ)
     : GOTO 46
5Ø CLS
     : PRINT"CREATING RANDOM GRAPHICS"
     : DIM S(N)
6Ø FOR L=Ø TO N
7Ø S(L)=S+RND(E-S+1)-1
8Ø NEXT L
9Ø GOSUB 29Ø
1ØØ GOSUB 12Ø
11Ø GOTO 6Ø
12Ø M=N
13Ø M=INT(M/2)
     : SOUND 2ØØ,5
     : IF M=Ø THEN RETURN
14Ø J=Ø
     : K=N-M
15Ø I=J
16Ø L=I+M
165 CM=CM+1
```



```
18Ø IF S(I)<S(L) THEN 26Ø
19Ø T=S(I)
     : S(I)=S(L)
     : S(L)=T
2ØØ PRINT@Ø+I. CHR$(S(I)):
21Ø PRINT@Ø+L, CHR$(S(L));
22Ø SW=SW+1
23Ø I=I-M
24Ø IF I<1 THEN 26Ø
25Ø GOTO 16Ø
26Ø J=J+1
27Ø IF J>K THEN 13Ø
28Ø GOTO 15Ø
29Ø REM ... PRINTOUT ...
3ØØ FOR L=Ø TO N
31Ø PRINT@Ø+L, CHR$(S(L));
32Ø NEXT L
33Ø RETURN
```

This is a "shell-type" sort routine. The input format and printout formats are similar to those explained in the Color Computer Learning Lab software package. You will find that this routine will sort larger numbers faster than the "bubble-type" sort included in that package. For the inputs, use 128 for START, 255 for END, and 510 for NUMBER. This limits the items selected for sorting to only graphics characters and only as many characters as will fit on the screen without causing the display to scroll.



This next program will use a "bubble sort" that is placed into assembly code in the machine. You will find it very fast compared to the last one.

```
1Ø DATA 142, 4, Ø, 134, 96, 167, 128, 14Ø, 6, Ø,
     38, 249, 141, 51, 142, 4, Ø, 191, 6, 115,
     16, 142, 6, Ø, 166, 132, 161, 164, 47, 6,
     23Ø, 164
2Ø DATA 167, 164, 231, 132, 49, 63, 16, 188, 6,
     115, 46, 236, 48, 1, 191, 6, 115, 16, 142,
     6, Ø, 14Ø, 6, Ø, 45, 222, 173, 159, 16Ø, Ø,
     39, 2Ø4, 63
3Ø DATA 142, 4, Ø, 141, 1Ø, 138, 128, 167, 128,
     14Ø, 6, Ø, 38, 245, 57, 52, 4, 198, 23, 182,
     6, 117, 72, 184, 6, 117, 132, 64, 129, 64,
     118, 6, 118, 182, 6, 117, 7Ø, 132, 254
4Ø DATA 183, 6, 117, 9Ø, 38, 233, 182, 6, 118,
     53, 132, Ø, Ø, 85, 85
5Ø FOR X=&HØ6ØØ TO &HØ676
6Ø READ A
7Ø POKE X, A
8Ø NEXT X
9Ø EXEC &HØ6ØØ
```

Well, at least one of these programs should keep you watching your Color machine until next month.

Call or Visit the Radio Shack Computer Center Near You for More Information

ALABAMA
BIRMINGHAM 2428 Green Springs Hwy., (205) 945-0792
HUNTSVILLE 1400 N. Memorial Pkwy., (205) 536-1581
MOBILE 405 Bel-Air Blvd., (205) 471-1617
MONTGOMERY #24 Union Square S/C, (205) 271-1500

ARIZONA
PHOENIX 10233 Metro Pkwy. E., (602) 861-1124; 4301 N. 7th Ave., (602) 277-3031
SCOTTSDALE 2525 N. Scottsdale Rd., (602) 990-2241
TEMPE 83 E. Broadway, (602) 894-2065
TUCSON 5622 E. Broadway, (602) 748-0101

ARKANSAS
LITTLE ROCK Town & Country S/C, University & Asher, (501) 568-5694

CALIFORNIA
ANAHEIM 509 Katella, (714) 776-9540
BERKELEY 1922 Grove St., (415) 848-9170
BEVERLY HILLS 8500 Wilshire Blvd., (213) 659-8870
CANOGA PARK 8371 Topanga Canyon, (213) 347-9800
CARMICHAEL 6305 Fairoaks Blvd., (916) 484-6815
CHULA VISTA 1201 3rd Ave., (714) 420-3810
DOWNEY 8031 Florence Ave., (213) 927-7882
ESCONDIDO 347 W. Mission Ave., (714) 741-6032
FRESNO Princeton S/C, 2721 N. Blackstone Ave., (209) 225-5551
GLENDALE 236 N. Brand Blvd., (213) 246-9310
HAYWARD 20942 Mission Blvd., (415) 278-2888
LAKEWOOD 5830 Lakewood Blvd., (213) 920-9671
LA MESA 5346 Jackson Dr., (714) 460-3610
LONG BEACH 2119 Bellflower Blvd., (213) 597-3377
LOS ANGELES 740 S. Olive St., (213) 629-2455
MONTEREY 484 Washington St., (408) 375-8430
MOUNTAIN VIEW 1933 El Camino Real W., (415) 961-0542
PASADENA 575 S. Lake Ave., (213) 449-5424
RIVERSIDE 3844 La Sierra Ave., (714) 689-0340
SACRAMENTO 4749 J. St., (916) 454-3287
SAN BERNARDINO 764 Inland Center Dr., (714) 884-6871
SAN DIEGO 3062 Clairemont Dr., (714) 276-6050; 3902 El Cajon Blvd., (714) 280-0227
SAN FRANCISCO One Market Place, (415) 777-9810
SAN JOSE 1228 S. Bascom Ave., (408) 297-2603
SAN MATEO 3180 Campus Dr., (415) 573-8607
SANTA BARBARA 4141 State St. A-1, (805) 967-4538
SANTA FE SPRINGS 14138 East Firestone Blvd., (213) 921-0702
STOCKTON College Sq. S/C, 963 West March Lane, (209) 957-3676
TORRANCE 3840 Sepulveda at Hawthorne, (213) 373-0306
VENTURA 4005 E. Main St., (805) 654-0196
WEST COVINA 2516 E. Workman St., (213) 915-5791

COLORADO
BOULDER Arapahoe Plaza, 3550 Arapahoe, (303) 443-7142
COLORADO SPRINGS 4341 N. Academy, (303) 593-7500
DENVER 8000 E. Quincy, (303) 770-1362
LAKEWOOD 2099 Wadsworth Blvd., (303) 232-6277

CONNECTICUT
EAST HAVEN 51 Frontage Rd., (203) 467-8864
FAIRFIELD 1196 Kings Hwy. & Rt. 1, (203) 255-6099
MANCHESTER 228 Spencer St., (203) 649-8210
NORWALK Rt. 7-345 Main Ave., (203) 846-3418
ORANGE Woolco S/C; 538 Boston Post Rd., (203) 795-1291
WATERBURY 105 Bank St., (203) 573-8800
WATERFORD 122 Boston Port Rd., (203) 443-0716
WEST HARTFORD 39 S. Main St., (203) 523-4283

DELAWARE
WILMINGTON 3847 Kirkwood Hwy., (302) 999-0193

DISTRICT OF COLUMBIA
WASHINGTON 1800 M St. NW., (202) 822-3933

FLORIDA
ALTAMONTE SPRINGS 766 B. East Altamonte Dr., (305) 339-1313
CLEARWATER 2460-D US 19 North, (813) 797-3223
HOLLYWOOD 429 S. State Rd. #7, (305) 966-4382
JACKSONVILLE 8252 Arlington Exprwy., (904) 725-2594
LAUDERDALE LAKES 4317-25 N. State Rd. 7, (305) 486-2240
MIAMI 9459 S. Dixie Hwy., (305) 667-2316; 1601 Biscayne Blvd., (305) 374-6433; 15 SE. 2nd Ave., (305) 374-7310; 20761 S. Dixie Hwy., (305) 238-2518
N. MIAMI BEACH The Promenade, 1777 N.E. 163rd St., (305) 940-6887
ORLANDO 1238 E. Colonial Dr., (305) 894-0570
ST. PETERSBURG 3451 66th St. N., (813) 381-2366
SARASOTA 5251 S. Tamiami Tr. (Hwy.41), (813) 923-4721
TALLAHASSEE 2529 S. Adams, (904) 222-4440
TAMPA 4555 W. Kennedy, (813) 879-7470; 1825 E. Fowler Ave., (813) 971-1130
W. PALM BEACH 2271-A Palm Beach Lakes Blvd., (305) 683-3100

GEORGIA
AUGUSTA 3435 Wrightsboro Rd., (404) 738-5998
ATLANTA 2108 Henderson Mill, (404) 939-9888; 49 W. Paces Ferry, (404) 231-9604; Akers Mill S/C, 2937 Cobb Parkway NW., (404) 955-5235; 113 Peachtree St., (404) 223-5904
COLLEGE PARK 5309 Old National Hwy., (404) 761-3055
DORAVILLE 5697 Buford Hwy., (404) 458-2691
SAVANNAH Chatham Plaza, 7805 Abercorn St., (912) 355-6074

IDAHO
BOISE 691 S. Capitol Blvd., (208) 344-5450

ILLINOIS
AURORA 890 North Lake St., (312) 844-2224
CHICAGO 4355 S. Archer Ave., (312) 376-7617; CNA Plaza, 309 S. Wabash, (312) 922-0536
ELMWOOD PARK 7212 W. Grand Ave., (312) 452-7500
FAIRVIEW HEIGHTS #4 Market Place, (618) 398-6410
HOMEWOOD/GLENWOOD 329 Glenwood Lansing, (312) 758-0440
LaGRANGE One S. LaGrange Rd., (312) 482-3484
LIBERTYVILLE 1350 S. Milwaukee Ave., (312) 367-8230
LOMBARD 4 Yorktown Center, (312) 629-5350
NILES 8349 Golf Rd., (312) 470-0670
OAK LAWN 4815 W. 95th St., (312) 425-9130
PEORIA 4125 N. Sheridan Rd., (309) 685-7056
ROCKFORD North Town S/C, 3600 N. Main St., (815) 282-1001
SCHAUMBURG 651 Mall Dr., (312) 884-8600
SPRINGFIELD Sherwood Plaza, 2478 Wabash, (217) 787-3066

INDIANA
EVANSVILLE 431 Diamond Ave., (812) 426-1715
FT. WAYNE 747 Northcrest S/C, (219) 482-9547
GRIFFITH 208 W. Ridge Rd., (219) 838-3000
INDIANAPOLIS 6242 E. 82nd St., Castleton Plz., (317) 849-6896; Speedway Plaza, 6129 B Crawfordsville, (317) 244-2221; 10013 E. Washington St., (317) 898-4887

IOWA
DAVENPORT 616 E. Kimberly Rd., (319) 386-3457
DES MOINES 7660 Hickman Rd., Sherwood Forest S/M, (515) 270-0193

KANSAS
OVERLAND PARK 8619 W. 95th, (913) 642-1301
WICHITA 2732 Blvd. Plaza S/C, (316) 681-1212

KENTUCKY
FLORENCE 7727 Mall Rd., (606) 371-2811
LEXINGTON 2909 Richmond Rd., (606) 269-7321
LOUISVILLE 2900 Taylorsville Rd., (502) 459-9901

LOUISIANA
BATON ROUGE 7007 Florida Blvd., (504) 928-5260
METAIRIE 3750 Veterans Hwy., (504) 454-3681
NEW ORLEANS 327 St. Charles Ave., (504) 523-6408
SHREVEPORT 1545 Line Ave., (318) 221-5125

MAINE
BANGOR Maine Square, (207) 945-6491

MARYLAND
BALTIMORE 7942 Belair Rd., Putty Hill Plaza, (301) 882-9583; 115 N. Charles St. at Lexington, (301) 539-7251
CATONSVILLE One Mile West S/C, 6600 B Balt. Nat'l. Pike, (301) 788-3277
FREDERICK Shoppers World, Rt. 40W, (301) 695-8440
NEW CARROLLTON-LANHAM 7949 Annapolis Rd., (301) 459-8030
ROCKVILLE Congressional Plaza, 1673 Rockville Pike, (301) 984-0424
SALISBURY Shoppers World S/C, Rt. 50, (301) 546-9223

MASSACHUSETTS
BOSTON 730 Commonwealth Ave., (617) 739-1704
BRAINTREE South Shore Plaza, 250 Granite St., (617) 848-9290
BROCKTON 675 Belmont, (617) 583-2270
BURLINGTON Crossroads Plaza, Rt. 3 S., (617) 229-2850
CAMBRIDGE Harvard Square, 28 Boylston St., (617) 354-7694
CHESTNUT HILL 200 Boylston St., (617) 969-2031
NATICK 1400 Worcester Rd., (617) 875-8721
SAUGUS 343 Broadway, (617) 233-4985
SPRINGFIELD 1985 Main St., Northgate Plz., (413) 732-4745
WORCESTER Lincoln Plaza, (617) 852-8844

MICHIGAN
BIRMINGHAM 3620 W. Maple Rd., (313) 647-2151
DETROIT DWNTN 1559 Woodward Ave., (313) 961-6855
FLINT G3298 Miller Rd., Yorkshire Plaza, (313) 732-2530
GRAND RAPIDS 3142 28th St. SE., (616) 957-2040
KALAMAZOO 25 Kalamazoo Center, (616) 343-0780
LANSING 2519 S. Cedar St., (517) 372-1120
LIVONIA 33470 W. 7 Mile Rd., (313) 476-6800
ROSEVILLE 31873 Gratiot Ave., (313) 296-6210
SOUTHFIELD 17651 West 12 Mile Rd., (313) 569-1027
TROY Oakland Plaza, 322 John R. Rd., (313) 585-3900

MINNESOTA
BLOOMINGTON 10566 France Ave. S., (612) 884-1641
GOLDEN VALLEY Golden Valley S/C, 8016 Olson Memorial Hwy., (612) 542-8471
ST. PAUL 6th & Wabasha, (612) 291-7230

MISSISSIPPI
JACKSON 979 Ellis Ave., (601) 352-5001

MISSOURI
FLORISSANT 47 Florissant Oaks S/C, (314) 921-7722
INDEPENDENCE 1325 S. Noland Rd., (816) 254-3701
KANSAS CITY 4025 N. Oak Trafficway, (816) 455-3381
ST. ANN 10472 St. Charles Rock Rd., (314) 428-1400

NEBRASKA
OMAHA 3006 Dodge St., (402) 346-4003

NEVADA
LAS VEGAS Commercial Center, 953 E. Sahara #31-B, (702) 731-3956
RENO 3328 Kietzke Lane, (702) 826-6327

NEW HAMPSHIRE
MANCHESTER Hampshire Plaza, 1000 Elm St., (603) 625-4040

NEW JERSEY
BRIDGEWATER 1472 U.S. Highway 22 East, (201) 469-3232
E. BRUNSWICK 595 A Rt. 18, (201) 238-7142
E. HANOVER Rt. 10, Hanover Plaza, (201) 884-1200
LAWRENCEVILLE Rt. 1 & Texas Ave., (609) 771-8113
NEWARK 595 Broad, (201) 622-1339
PARAMUS 175 Rt. 17 S., (201) 262-1920
SPRINGFIELD Rt. #22 Center Isle, (201) 467-9827

NEW MEXICO
ALBUQUERQUE 2108 San Mateo NE., (505) 265-9587

NEW YORK
ALBANY Shoppers Pk., Wolf Rd., (518) 459-5527
BAYSHORE 1751 Sunrise Hwy., (516) 666-1800
BETHPAGE 422 N. Wantagh Ave., (516) 822-6403
BUFFALO 839 Niagara Falls Blvd., (716) 837-2590
FRESH MEADOWS 187-12 Horace Harding Exp., (212) 454-1075
JOHNSON CITY Giant Shopping Center, Harry L. Drive, (607) 729-6312
MELVILLE TSS Mall, Rt. 110, (516) 673-4646
NEW ROCHELLE 242 North Ave., (914) 636-0700
NEW YORK 385 Fifth Ave., (212) 889-1345; 139 E. 42nd St., (212) 953-6053
REGO PARK 97-77 Queens Blvd., (212) 897-5200
ROCHESTER 3000 Winton Rd., (716) 244-6400
SPRING VALLEY White House Center, 88 W. Rt. 59, (914) 425-2828
STATEN ISLAND 2409 Richmond Ave., (212) 698-3100
SYRACUSE 2544 Erie Blvd., (315) 446-3017
UTICA Riverside Mall, (315) 735-1933

NORTH CAROLINA
CHARLOTTE 3732 Independence Blvd., (704) 535-6320
GREENSBORO 3718 High Point Rd., (919) 294-5529
RALEIGH Townridge Sq., Hwy. 70 W., (919) 781-9380
WINSTON-SALEM 629 Peters Creek Pkwy., (919) 722-0030

OHIO
AKRON Fairlawn Plaza, 2727 W. Market St., (216) 836-9303
CANTON 5248 Dressler Rd. NW., (216) 494-7230; Mellet Plaza, 3826 W. Tuscarawas, (216) 478-1878
CENTERVILLE 2026 Miamisburg-Centerville Rd., (513) 435-5167
CINCINNATI 9725 Montgomery, (513) 793-8688
CLEVELAND 419 Euclid (Dwntwn), (216) 575-0800; 27561 Euclid Ave., (216) 289-6823
COLUMBUS 862 S. Hamilton, Great Eastern S/C, (614) 864-2806; The Patio Shop. Ctr., 4661 Karl Rd., (614) 888-8227
DAYTON Northwest Plaza, 3279 West Siebenthaler, (513) 277-6500
NORTH OLMSTED Great Northern S/C, (216) 734-2255
TOLEDO 5844 W. Central Ave., (419) 531-5797
YOUNGSTOWN Union Square Plaza, 2543 Belmont Ave., (216) 744-4541

OKLAHOMA
OKLAHOMA CITY 4732 SE 29th St., (405) 670-4561; Springdale S/C, 4469 NW 50th, (405) 943-8712
TULSA 7218 & 7220 E. 41st St., (918) 663-2190

OREGON
EUGENE 390 Coburg Rd., (503) 687-0082
PORTLAND 7463 SW Barbur Blvd., (503) 246-1157; 9131 SE Powell, (503) 777-2223

PENNSYLVANIA
ALLENTOWN Crest Plaza S/C, Cedar Crest Blvd. US 22, (215) 395-7155
BALA CYNWYD 67 E. City Line Ave., (215) 668-9950
ERIE 5755 Peach St., (814) 868-5541
HARRISBURG Union Deposit Mall, Union Deposit Rd. #17, (717) 564-6753
LANCASTER Park City Plaza, US 30, (717) 393-5817
MONROEVILLE 3828 Wm. Penn. Hwy., (412) 823-3400
MONTGOMERYVILLE Airport Sq., Rt. 309, (215) 362-1200
PHILADELPHIA 7542 Castor Ave., (215) 342-2217; 1002 Chestnut St., (215) 923-3080
PITTSBURGH 5775 Baptist Rd., Hills Plaza, (412) 831-9694; 303 Smithfield St., (412) 391-3150
SCRANTON 206 Meadow Ave., (717) 348-1801

RHODE ISLAND
E. PROVIDENCE 850 Waterman Ave., (401) 438-2860

SOUTH CAROLINA
COLUMBIA Old Sears Bldg., 1001 Harden St., (803) 799-2065
GREENVILLE N. Hills S/C, (803) 292-1835
N. CHARLESTON 5900 Rivers Ave., (803) 747-5580

TENNESSEE
CHATTANOOGA 636 Northgate Mall, (615) 870-1366
JOHNSON CITY Peerless Center, (615) 282-6829
KNOXVILLE Cedar Bluff S/C, 9123 Executive Park Dr., (615) 690-0520
MEMPHIS 4665 American Way, (901) 795-4963; 1997 Union Ave., (901) 278-7935
NASHVILLE 2115 Franklin Pike, (615) 298-5484; Rivergate Plaza, (615) 859-3414

TEXAS
ARLINGTON 2500 E. Randol Mill, Suite 113, (817) 274-3127
AUSTIN 8764 E. Research Blvd., (512) 459-4238
BEAUMONT 5330 Eastex Frwy., (713) 898-7000
CORPUS CHRISTI 1711 S. Staple St., (512) 887-8901
DALLAS 15340 Dallas Pkwy., Suite 1100, (214) 934-0275; 2930 W. Northwest Hwy., (214) 350-4144; 1517 Main St., (214) 760-8601; 2588 Royal Ln., (214) 484-9947
EL PASO 9515 Gateway West, (915) 594-8211
FT. WORTH 15 One Tandy Center, (817) 335-7198; 2801 Alta Mere, (817) 738-0251
HOUSTON 211C-FM 1960, (713) 444-7006; 10543 Gulf Fwy., (713) 943-9310; 5900 North Fwy., (713) 699-1932; 6813 SW Fwy., (713) 777-7907; 809 Dallas St., (713) 651-3002
HURST Northeast Mall, (817) 284-1518
LUBBOCK 3625 34th St., (806) 793-1467
ODESSA 1613 "A" East 8th Street, (915) 334-8355
RICHARDSON Fleetwood Sq. S/C, 202 W. Campbell Rd., (214) 669-1494
SAN ANTONIO 6018 West Ave., (512) 344-8792; 4249 Centergate, (512) 657-3958

UTAH
MURRAY 6051 S. State Ave., (801) 268-8978
SALT LAKE CITY 415 5th Ave., (801) 322-4893

VIRGINIA
ALEXANDRIA 4527 Duke St., Westend S/C, (703) 370-9000
FAIRFAX Westfair Center, 11027 Lee Hwy., (703) 273-6500
NORFOLK 5731 Poplar Hall Dr., (804) 461-0798
RICHMOND Willow Lawn S/C, 1617 Willow Lawn Dr., (804) 282-3453; 7728 Midlothian Turnpike, (804) 272-8803
ROANOKE Franklin Bldg., 3561 Franklin Rd. S.W., (703) 342-6335

WASHINGTON
BELLEVUE Crossroads Mall, North East 8th & 156 St., (206) 644-1804
FEDERAL WAY 33505 Pacific Hwy. South, (206) 838-6830
SEATTLE 18405 Aurora Ave. N., (206) 542-6184
SPOKANE 7702 N. Division, (509) 484-7000; E. 12412 Sprague, (509) 922-2800
TACOMA 7030 S. Sprague, (206) 473-7333
TUKWILA 15425 53rd Ave. S., (206) 248-3710
YAKIMA 1111 N. First St., (509) 248-9667

WEST VIRGINIA
HUNTINGTON 2701½ 5th Ave., (304) 523-3527

WISCONSIN
MADISON 57 West Towne Mall, (608) 833-6130
MILWAUKEE 6450 N. 76th St., (414) 353-6730
WEST ALLIS 2717 South 108th St., (414) 327-4240

Radio Shack®

TRS-80 Microcomputer News
P.O. Box 2910
Fort Worth, Texas 76113-2910

FIRST CLASS
U.S. POSTAGE
PAID
DALLAS, TX
PERMIT 2180

ADDRESS CHANGE
☐ **Remove from list** ☐ **Change as shown**
Please detach address label and mail to address shown above