Radio Shack TRS-80™ Microcomputer NEWS

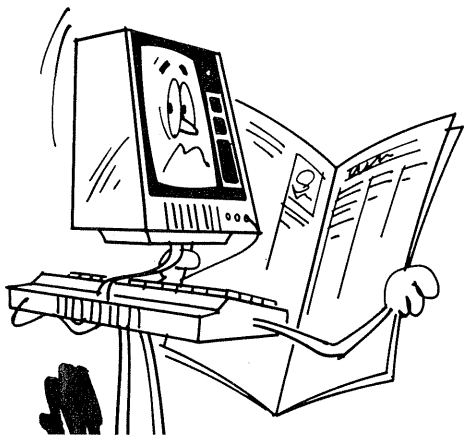
July, 1981

P.O. Box 2910, Fort Worth, Texas 76101

THE MICROCOMPUTER NEWSLETTER PUBLISHED FOR TRS-80 OWNERS

Volume 3, Issue 7

Newsletter Index In This Issue...



Radio Shack Helps Newspaper go Electronic

The Advertiser-Tribune in Tiffin, Ohio has selected Radio Shack equipment to help deliver an electronic edition of the newspaper. Early plans call for market research and system refinement this summer, limited marketing to existing home computer owners in the fall, with full scale promotional efforts to begin in early 1982.

A recent letter from The Advertiser-Tribune to Radio Shack outlined some of the goals of the electronic edition, as well as some of the newspaper's rationale in choosing Radio Shack equipment:

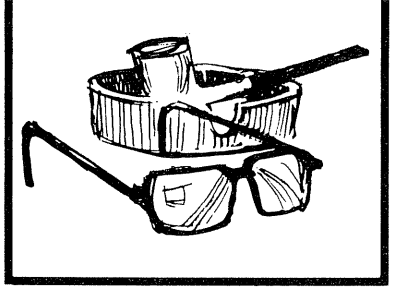
"Our goals in this project are several. We believe that some form of electronic newspaper will emerge as an important information medium in the near future. As a small newspaper in a relatively small newspaper group, we do not have the resources to pursue a major R&D effort, yet we want to be involved and have a hand in developing the future landscape. Most of the existing experiments are either too large in scope or too demanding of financial and human resources for a newspaper our size. Other experiments involving cable television delivery of character-generated news and information present limited learning opportunities.

"The Tandy/Radio Shack videotex host computer seems to present the perfect learning opportunity for us at this time. It offers some limited interactivity, graphics, and on-demand delivery of news and information to homes using "off the shelf" terminals which are locally available for a fairly modest price.

"Our prime goal is to develop and offer for sale an effective electronic data base which complements the contents of our newspaper. We hope to earn a reasonable profit on our efforts within a reasonable amount of time, and at the same time to learn a great deal about the development and delivery of an electronic data base."

A pair of Radio Shack 64K Model II's and Radio Shack's 8-line telephone multiplexer will be used by The Advertiser-Tribune to make the electronic edition available to home subscribers. The electronic edition of the newspaper will be formatted for use with either the Radio Shack Videotex Terminal or the Radio Shack Color Computer.

The Advertiser-Tribune is a member of the Buckner News Alliance, which includes newspapers in Tiffin, Ohio; York and Lewiston, Pennsylvania; Carlsbad, New Mexico; Pecos, Texas; and Fontana, California. The Advertiser-Tribune has a daily circulation of 11,500 copies.



View From the 7th Floor

by Jon Shirley, Vice President Computer Division

Last month I began telling you about RSBASIC, Radio Shack's new Compiler BASIC for Models I, II, and III. This month we look at disk files.

Back to basics (whoops) again. If you are still with me and have not written disk software read on and find out the real reason you need disk drives. Our interpreters allow two kinds of disk files, sequential and direct (also called random). A sequential file is just like a cassette tape file. You send out data from memory in a continuous stream that is saved on the disk. The procedure is to open a file, send it all the data, close the file. When you want some data from the file you must open it and read in the whole file. If you want to add data to the file you must read it all in, add the data, and write it all back. The problem is that you may have to search the entire file to find even one record.

A direct file is a lot neater. Each record has a record number and you can write just one record or read just one record as long as you keep track of which number is attached to which record. Usually this is done by keeping another file called an index file which contains the record numbers in a sorted sequence. For example if you have a file of names and addresses, they would be put on the disk as they are entered, in no particular order, with an index number assigned to each one. If you wanted to print out your list in order you would sort the names alphabetically and sort the numbers at the same time. Thus aardvark, with record number 47 would be the first number in the index file. Sound like work? It is!

RSBASIC adds the third type of disk file and as far as we know it is the only micro BASIC that has it. "It" is called ISAM (Index Sequential Access Method) and is a way to store and find disk records one at a time like direct access without having to create index files. Sound neat — it is, and its a feature of virtually all COBOLs. Our RSBASIC allows a single key to find the records (our COBOL allows multiple keys). So what is a key? Let's go back to our file of names and addresses. With ISAM we can tell the file that the last name in each record is the key for that record. When we want to find aardvark we simply tell the program to find aardvark. Magic? A little.

The way this works is that the program builds its own index on the disk and does its own searching to find the exact record you want. It does take up disk space, there are no free lunches, but it does make programming very easy. The key can be a part of the stored record or it can be an entirely different variable. By single key ISAM we mean there can only be one key for that file, if your key is last names you can only find records by last names. With our COBOL and its multiple keys you could have last name, zip code, and state all as keys. This can get out of hand and run you out of disk space fast so back to RSBASIC.

Now if you have a disk system, and have tried your hand at direct files, you are probably aware that it is very difficult to code direct access files in our interpreter. Just in case you have forgotten, here is how it's done:

1. OPEN the file
2. FIELD the buffer
3. convert numeric data with MKD$,MKI$,MKS$
4. LSET or RSET the values into the fields
5. PUT the record
6. CLOSE the file

If you can remember that without a manual close by you are a better person than I.

With RSBASIC all three file types (sequential, random and ISAM) use the same set of commands.

1. OPEN the file (and the open statement defines what kind of file it will be).
2. PRINT the file.
3. CLOSE the file.

That's all there is to it. You have three choices for printing the file, PRINT, PRINTUSING or WRITE (which stores the data on the disk in a space saving binary format).

A typical ISAM file write looks just like this:

```
30 OPEN #1, "LIST/DAT",MODE=W,TYPE=I,LENGTH=32,KEY=20
40 PRINT #1, KEY=NAME$; NAME$, PAYRAT, EXEMPT
```

To find this record later all you do is open the file and INPUT with the key (NAME$) set to the name you want. To get the record on Smith, A.B. set the key to KEY = "Smith, A.B.". Easy, right?

If you remember last month's column you will recall I said that RSBASIC will allow you to write a program that will run on all three disk TRS-80's. The key to this is that run-time module I mentioned earlier. It converts the compiled code to the final machine code and is different for each TRS-80 model. The compiled object code is not different so you could transfer an object file from a Model III to a Model II (by RS-232) and the Model II run-time would be able to go ahead and run the program. Of course the screen layout is different but if you were really clever you could . . . but that's another article.

That's not all there is to RSBASIC but I think you can see why we choose it over the others and perhaps why you should visit your nearest Computer Center and get a look in person. It comes in a huge binder because it has a very complete manual with lots of program examples for every command and a very complete index. Our manual alone puts it above the competition.

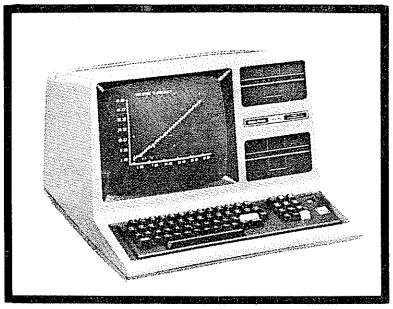
Until next month.

New Characters

John Feher Allentown, Pennsylvania

I have recently had (the Radio Shack) Lower Case Kit cat. #26-1104 installed in my Model I 32K Business System with Line Printer II. When I was testing to see what all the characters looked like, I was able to generate six new additional characters by POKEing their code into Video Memory. The only drawback is that it will only do this by POKE's and not the CHR$ command. I have supplied a little routine below for the benefit of other Model I Upper/Lower Case users.

```
5 CLS
10 DATA 96, 123, 124, 125, 126, 127
20 A=15361
30 FOR B%=1 TO 6
    : READ CH
    : POKE A, CH
    : PRINT@A-15360+63, CH;
    : A=A+5
    : NEXT
40 PRINT CHR$(10)
    : END
```



Model I/III

Product Line Manager's News

Here is some great news for both Model I and Model III owners! This month we announced new low prices on Model I Expansion Interfaces (E/I's) and memory upgrades. The 16K E/I (26-1141) now has a suggested retail price of $398.00, down from $429.00. We've also reduced the price of the 32K E/I (26-1142) from a suggested retail price of $559.00 to a low $497.00. Salt away the savings for all of those "necessities" that can be added once you have acquired the E/I. Necessities like Model I disk drives. The E/I also provides for the use of a printer, a second cassette recorder and an RS-232 so you can go full duplex on CompuServe. Don't miss out on the break on memory upgrades either. The Model I 16K RAM and keypad upgrade (26-1101 — installation extra) is now only $180, down from $200. We've also passed along a price break for owners on the 16K RAM upgrade (26-1102). That upgrade is now only $99 (again this price does not include installation), a savings of $20 from the previous price of $119.

Speeding Bullets, and other topics . . .

Have you ever wondered if there were a "better" or faster way to get something done using your TRS-80? If you are like many of us, the answer to that is YES, especially if you have tried writing a routine to sort 500 items in a string array, or if you have tried to animate graphics, or . . . The list can be endless.

You may have heard or read a little about a mystery language which makes your TRS-80 faster than a speeding bullet, able to leap buildings in a single bound, etc. The language is machine language or assembly language (they are both talked about). What is machine language? Machine language for the TRS-80 Models I and III is the specific instructions used by the Z-80 microprocessor which is the heart of your TRS-80.

Well, you say, I thought that was BASIC. Actually, inside the TRS-80, tucked away in the lower reaches of memory, is an interpreter. The function of this interpreter is to change the programs you write in BASIC into machine language that the Z-80 can understand and execute. The interpreter does a very good job, is reasonably efficient, and seldom gets in your way. It is also somewhat slow. A sort routine which takes BASIC 45 minutes to complete can be easily finished by a machine language program in 9 seconds. Quite a difference, right? This is just one example of why it's worth your time to delve into the "secrets" of machine language programming.

I think I know your next question. If machine language is so much faster than BASIC, why do we use BASIC? It's because BASIC is reasonably easy to learn and understand. BASIC uses commands and instructions like PRINT, RUN and LIST which are English words that easily convey (to the person reading them) the intent or purpose of what we are asking the computer to do. If we tell the computer to PRINT 5 + 5, using BASIC it is easy to see that we want the TRS-80 to display the sum of 5 + 5 on the video display.

Well, you say, I am willing to suffer a little to gain that much speed. OK, here is a section of machine language code (don't ask me to interpret it, I am just passing it along) — 1110110110110000. Did you get that? It's a command which tells the computer to move a block of information from one part of memory to another part. We could convert this binary (ones and zeros) information into a slightly more readable (from a people standpoint) form by using hexadecimal notation — EDB0. Is that better? No?

I think you can see why most of us don't program in machine language. Relax — you don't need to get a headache looking at those ones and zeros. You also don't have to remember all of the possible combinations. The trick is to use assembly language. Assembly language is a "low level" language which is easier to understand and use than the ones and zeros of machine language, but which requires a little more work from you than BASIC requires. Remember our machine language code? In assembly language, it becomes LDIR. Still doesn't make sense? Think of LDIR as being an abbreviation for LoaD, Increment and Repeat, which describes what the command does, it LoaDs information into a new memory location, Increments a counter by −1, and then Repeats the operation until the counter is zero.

Assembly language programming requires that you, as the programmer, be somewhat aware of what is happening (or not happening) inside the computer. In BASIC you simply tell the computer that you want A = 5. You don't worry about where the computer puts A, or how it keeps track of what line number to return to after a subroutine, or how it multiplies two numbers together (as long as you get the correct result). In assembly language, you have to tell the computer where to store variables like A. You also have to write your own multiplication routine (or borrow one from ROM). The Computer knows how to add and subtract, but not how to multiply. Assembly language does do some things for you. For instance, it keeps track of the line to return to after a subroutine, just like BASIC does.

One of the reasons that assembly language is so much faster than the BASIC which comes with your TRS-80 is the fact that we finish two steps in creating the program before we ever run it. In BASIC, you create your program by typing it into the computer. When you are through, you can type RUN and the interpreter (remember the little guy back in the corner?) will first have to interpret the BASIC commands, and then execute the commands. In assembly language however, we finish two steps before executing the program. First you use an Editor to create what is called a "source" program or "source code." (When you write a BASIC program this is the only step you do.) The source code is then assembled into machine language using an Assembler. (The BASIC interpreter has to do this while it is running your program.) The result is a machine language program which can be run directly if it is a complete program, or you can interface it to Level II or Model III BASIC using a USR call. (USR calls may be the subject of a future article).

Assembly language can be relatively simple or extremely complex, depending on what you want to do, and how fancy you want to be doing it. If you want to write your company's Accounts Receivable program in assembly language, good luck. It can be done, it will take awhile, and it probably won't be much faster than a good A/R written in Compiler BASIC, or COBOL. If you want to write short routines to enhance your BASIC programs, or even complete machine language programs, there are some fantastic possibilities. Assembly language programs execute very rapidly, generally take up very little space, and teach you alot about programming, and computers.

If you are interested in serious assembly language programming, Radio Shack offers the Disk Editor/Assembler package (26-2202 Suggested Retail Price $99.95). We recommend it for experienced assembly language programmers. Disk Editor/Assembler contains advanced features like Macros, conditional

(Continued on Page 4)

Model I/III (From Page 3)

assembly, fully relocatable code, Z-80 or 8080 mnemonics, a Linking Loader, and much more. This program is available for 32K Model I disk systems only.

We have introduced our new Series-I Editor-Assembler. Series-I is an upgraded version of our original Tape Editor-Assembler (26-2002) which was for Model I tape based systems only. Series-I includes some enhancements to the Editor program, along with a new 250 page manual which we think will make writing assembly language programs much easier for beginners. A sample session and sample program help to guide you through the process of using the system.

Series-I Editor-Assembler is available in two versions, 26-2011 for tape based systems (suggested retail price $29.95) and 26-2013 for Disk systems (suggested retail price $34.95). Both versions are designed to work with either Model I or Model III. The tape based 26-2011 requires 16K of memory and will work in Level I (Model I or III, using the special tape which is included), Level II Model I's and Model III's with Model III BASIC. The disk based 26-2013 is Radio Shack's first low cost Editor-Assembler for Model I and III 32K disk users. All files are stored on disk for easy access and retrieval. We have also provided disk users with a special program which will let them move tape source files to disk, or to create tape object files from your disk object files.

If you are just getting started in assembly language, try our Series-I Editor-Assemblers. They are easy to use, and the documentation has been written to assist the first time assembly language programmer. Don't misunderstand, assembly language programming can be difficult, and you need to have access to assembly language programming manuals in addition to the Series-I Editor-Assembler. One book which we can highly recommend is "TRS-80 Assembly Language Programming" by William Barden, Jr. This book is available from Radio Shack (62-2006 $3.95). The book was written specifically for the Model I TRS-80, but is very useful with the Model III as well. Other books you may be interested in include Rodnay Zaks' "How to program the Z-80" (62-2066 $10.95), "Understanding Microprocessors" (62-2017 $2.95) and "Understanding Digital Computers" (62-2027 $4.95).

Assembly Language can be fun and educational (also fast), and with our Series-I Editor-Assembler, it is also inexpensive. If you have ever wanted to break away from your BASIC interpreter, start with Series-I.

Base Conversion

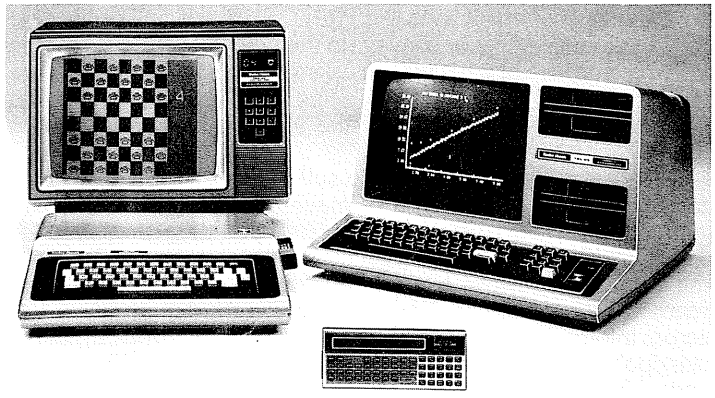
Gerald G. Noser Huntington Beach, California

This is a BASIC computer program that I have written for base conversion. It allows for the entry of a number in any base (2 through 16) and will calculate and print the equivalent value in a new base as specified by the user (again, 2 through 16). The allowable place holders (digits) of 0-9 and A-F are held in a dimensioned string variable for comparison of the data entered. A maximum value of 10,000,000 (base 10) has been set for the initial data entry.

All appropriate instructions, checks for the data entered and error messages are included within the program. It should be noted that bases larger than sixteen may be accommodated by adjusting the READ/DATA loop of lines 30 and 40 and also editing the checks for the maximum values of B1# and B2# in lines 120 and 150.

This program was written for a non-disk Level II Model I system.

```
10 REM BASE CONVERSION; SUBMITTED BY GERALD G. NOSER
20 CLEAR 500
    :DIM X$(16)
30 RESTORE
    :FOR X=1 TO 16
    :READ X$(X)
    :NEXT X
40 DATA 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
50 CLS
    :BT#=0
    :MV#=10000000
    :REM INIT BASE TEN AND MAX VALUES
60 PRINT@128, CHR$(30)"ENTER THE NUMBER TO BE CONVERTED"
70 PRINT@256, "(USE THE NUMERALS 0-9 AND/OR LETTERS A-F ON
    ENTRY.)"
80 N$=" "
    :PRINT@172,;
    :INPUT N$
    :PRINT
    :PRINT
90 IF N$=" " THEN 50
    :REM RE-ENTER N$
100 B1#=-1
    :PRINT@256, CHR$(30)"ENTER THE BASE OF THE NUMBER (2 TO 16)"
110 PRINT@300,;
    :INPUT B1#
120 IF B1#<2 OR B1#>16 OR B1#<>INT(B1#) THEN 100
    :REM RE-ENTER B1#
130 B2#=-1
    :PRINT@384, CHR$(30)"ENTER THE NEW BASE VALUE (2 TO 16)"
140 PRINT@428,;
    :INPUT B2#
150 IF B2#<2 OR B2#>16 OR B2#<>INT(B2#) THEN 130
    :REM RE-ENTER B2#
160 NM$=N$
    :Z=LEN(N$)
    :FOR X=1 TO Z
170 Y$=RIGHT$(N$,1)
    :FOR Y=1 TO 16
    :IF Y$=X$(Y) THEN 210 ELSE NEXT Y
180 REM COMPARE RIGHT-MOST CHARACTER TO ALLOWABLE PLACE HOLDERS
190 PRINT@512, "*** ILLEGAL CHARACTER ON ENTRY, RE-ENTER. ***"
200 FOR X=1 to 1000
    :NEXT X
    :GOTO 50
210 Y#=Y-1
    :IF Y#>=B1# THEN 190
    :REM CHARACTER WAS > = BASE VALUE
220 BT#=BT#+ INT(Y#*B1#^(X-1)+.5)
    :REM INCREMENT BASE TEN VALUE
230 IF BT#>MV# THEN PRINT@512,"*** NUMBER TOO LARGE RE-ENTER ***"
    :GOTO 200
240 N$=LEFT$(N$,LEN(N$)-1)
    :REM 'DROP' RIGHT-MOST CHARACTER OF STRING
250 NEXT X
    :IF B2#<>10 THEN 270
260 PRINT@640, NM$;" (BASE";B1#;") = ";BT#;"(BASE 10)"
    :GOTO 320
270 NV$=" "
    :REM NEW VALUE IN THE BASE B2# AS A STRING VARIABLE
280 R#=BT#-INT(BT#/B2#)*B2#
    :BT#=INT(BT#/B2#)
290 NV$=X$(R#+1)+NV$
    :REM REMAINDER R# CONVERTED TO APPROPRIATE CHAR X$() FROM
    DATA LIST
300 IF BT#>0 THEN 280
310 PRINT@640, NM$;" (BASE";B1#;") = ";NV$;" (BASE";B2#;")"
320 X$=" "
    :PRINT@768, CHR$(30)"AGAIN (Y/N)"
330 PRINT@783,;
    :INPUT X$
340 IF X$="Y" THEN 50
350 IF X$<>"N" THEN 320 ELSE END
```



SORTED COMPUTER PROGRAM DIRECTORY

Dave McGlumphy Red Bank, TN

To use this program, you need at least a 16K Level II Model I TRS-80. It'll be most helpful to have a printer available. This program can help both tape and disk users. I recently made the move up to a disk system because I wanted the additional speed. I found out very quickly that I was wasting too much time looking for programs that I had on disks. I felt like it was taking me as long to locate the programs and load them from disk as it had previously taken me to load them from tape because I'd used only one program per tape, and I had the tapes in alphabetical order making them very easy to find. Obviously, I needed an alphabetical listing of my programs so I could find them quicker.

I decided to number my disks sequentially using a two digit number and to print the alpha list showing the program name and the number of the disk holding the program. Tape users who put more than one program on a tape can do the same thing, adding the tape counter number to the program name.

There are a few tricks in this program that should be explained since they may not be obvious to the most casual observer on the first reading. In line 130, I read all the data from the DATA statements until I encounter @99, and I keep a count of how many items I've read in the variable, N. In line 150, I clear twice as much room as the number of items that I read, and I multiply that by fourteen since I'm allowing fourteen characters for each element of the string array that I'll sort, twelve for the program name and two for the disk number. If I clear less than twice as much space, BASIC has to rearrange his strings occassionally while building the array, and that looks funny when you watch it happening. If you run out of memory for your application or if you want to see the pauses for BASIC's string rearrangement, change line 150 to CLEAR N*14.

Notice that I DIMension the array for four more than the number of elements that I actually read. Look at line 290 or 300 to see why. If I don't allow the space for the extra elements in the array, I'll sometimes get a SUBSCRIPT OUT OF RANGE error message. Alternatively, I could use the ON ERROR GOTO statement to solve the problem, but the approach I'm using is simple, if not elegant.

Notice also that I actually read the data twice. The first time, I just figure out how much string space to CLEAR. When BASIC executes the CLEAR command, it resets a pointer so that the next time I read some data, I'm actually reading from the beginning of the data, just as though I'd said RESTORE. I could get around this problem by CLEARing an arbitrary amount of string space at the beginning of the program, but I wanted to try dynamically allocating the string space, and that means reading all the data, CLEARing what I need, and then reading the data again to load the array.

In lines 160, 170, 240, and 250, I issue messages to tell you what I'm doing so that you don't have to sit in front of a blank screen not knowing if the machine is working or not.

Line 190 tries to find the disk (or tape) number for all the programs in the current DATA statement. I set up each DATA statement with @nn as the first element, and nn is the sequential number I assigned to each disk. Line 200 limits each data element to 12 characters. If the resulting data element is less than 12 characters, line 210 pads it on the right with periods. Line 220 then puts the disk number as the two rightmost characters, so the result is a string element fourteen bytes long.

Here's the phenomenal part of this program! John C. Hallgren wrote a very fast machine language SORT, and line 250 is where I set up the parameters to tell his sort what I want to do. Refer to the REMarks at the beginning of the program to see what the sort parameters are. You can't believe how fast this sort works until you run it! The data statements from 520 to 620 are his sort program. I've coded this program for disk, so you'll have to make two changes to use it with tape. Change line 390 to read POKE 16526, U AND 255:POKE 16527,U/256. Be sure to include the word AND. It's important. Change line 400 to read U=USR(UN). (The zero is deleted.)

I might be biased, but I've found this program to be very helpful. It was worth the time to make it up, and if you're having trouble keeping up with your tapes and disks, it's probably worth your time to key it in.

```
10 REM DIRECTORY SORT BY DAVE MCGLUMPHY, 4429 PAULA LANE, RED
    BANK, TN, 37415
20 REM A$=STRING ELEMENT TO SORT
30 REM N=# OF ELEMENTS IN A$
40 REM UA=ADDRESS OF 1ST ELEMENT TO SORT
50 REM UH=INIT HIGH ORDER BYTE,
60 REM UL=INIT LOW ORDER BYTE, ALSO LENGTH OF KEY FOR $ SORT
70 REM UN=# OF ELEMENTS TO SORT
80 REM U0=SORT ORDER SWITCH, 0=ASCENDING, 1=DESCENDING
90 REM UP=START POSITION OF THE KEY IN THE SORT STRING
100 CLS
    :PRINT"SORTED DIRECTORY PROGRAM"
    :PRINT
    :PRINT"WHICH DO YOU WANT TO DO:
    ADD NEW PROGRAMS TO THE DIRECTORY
    PRINT A SORTED DIRECTORY"
    :I$=INKEY$
    :I$=""
110 I$=INKEY$
    :IF I$="A" THEN LIST 450-490 ELSE IF I$="P" THEN 120 ELSE 110
120 DEFINT N
    :CLS
    :GOSUB 130
    :GOTO 150
130 PRINT@0, "I'M READING THE DATA TO SEE HOW MANY PROGRAMS THERE
    ARE TO SORT ...";
140 READ D$
    :IF LEN(D$)=3 AND D$="@99" THEN RETURN ELSE N=N+1
    :GOTO 140
150 CLEAR 2*N*14
    :GOSUB 130
    :PRINT"THERE ARE";N;"PROGRAMS,"
    :DIM A$(N+4)
160 PRINT
    :PRINT"NOW I'M BUILDING THE SORT ROUTINE, OF 86 STEPS, I'M
    ON ..."
    :GOSUB 410
    :PRINT
170 PRINT
    :PRINT"NOW I'M BUILDING THE STRING TO BE SORTED..."
    :RESTORE
    :DNBR$=""
180 FOR J=1 TO N
    :READ D$
    :PRINT@362,J;
190 IF LEFT$(D$,1)="@" THEN DNBR$=RIGHT$(D$,2)
    :IF DNBR$="99" THEN 240 ELSE 230
200 IF LEN(D$)>12 THEN D$=LEFT$(D$,12)
210 IF LEN(D$)<12 THEN D$=D$+
    STRING$(12-LEN(D$),".")
220 A$(J)=D$+DNBR$
230 NEXT J
240 PRINT
    :PRINT
    :PRINT"THE ARRAY IS BUILT, NOW TO SORT IT, ";
250 UP=1
    :UL=14
    :U0=0
    :UN=N
    :UA=VARPTR(A$(1))
    :GOSUB 350
    :PRINT"DONE,"
    :PRINT
260 PRINT"DO YOU WANT THE OUTPUT ON THE SCREEN OR ON THE
    PRINTER?";
    :I$=INKEY$
    :I$=""
270 I$=INKEY$
    :IF I$<>"S" AND I$<>"P" THEN 270 ELSE PRINT I$
    :PRINT
    :IF I$="P" AND PEEK(14312)>127 THEN PRINT"THE PRINTER ISN'T
    READY YET,"
    :GOTO 260
```

(Continued on Page 12)

Disk Inventory

A Program by Bud Baker Hutchinson, Kansas

Disk inventory is a program for Model III disk-based TRS-80s. The purpose of the program is to create a master list of where your programs are located. Mr. Baker's program assumes that all programs are on TRSDOS diskettes which can be used in drive 0. You assign a two digit number to each diskette. Once you have assigned that number, you use the disk inventory program to create and expand a disk file which contains a program inventory.

The program allows you to read in a previously created program inventory, dump a new inventory to disk, sort the inventory by both alphabetically by disk and alphabetically by program name. You can read a new directory (and add it to the file), and output the inventory to either video or a printer by either program name or disk number.

Let's look at how the program works by creating a new program inventory. After you have entered the program and verified that there are no errors, save a copy of the program onto disk. I used the filename "DIR/BAS."

Now that you have a copy of the program saved, type RUN. The menu will appear, with eight options. Since we are just starting, the most reasonable choice is <4> READ ANOTHER DISK DIRECTORY. Press ④. The program will instruct you to insert a disk into drive 0 (be sure it is a TRSDOS diskette), and enter a disk number. The disk number is the two digit number you assigned to the diskette whose directory is about to be read.

As soon as you press (ENTER) and the computer determines that your two digit number is valid, the computer will read the disk directory and display it at the top of the video display. Next, you will see the computer "count" the number of entries, and the program will return to the menu.

Now that we have the first directory, the next step is to read in any additional directories that we want included in our program directory. After you finished reading the directories, the next step is to sort them. Press ③ to sort. This program uses the Model III sort utility to sort the directories. What the program does is append the disk number and a blank to the beginning of the filename for the sort by disk, and appends a blank and the disk number to the end of the file name for the sort by filenames.

After you have sorted the directories, write a copy of the list onto disk by using option ②. Now that you have a copy of your disk inventory stored safely on disk, you can use the remaining options 5-8 to look at the inventory and make a copy with your printer if you want one.

Mr. Baker set up the printer routines for a Line Printer IV, so you may want to modify them slightly if you have a different printer.

Program:

```
10 'DISK INVENTORY BY BUD BAKER RT #3 HUTCHINSON,
   KS 67501
20 'NO COPYRIGHT, FREE TO ALL, FOR MODEL III
   TRSDOS ONLY.
30 'READ DIRECTORIES THEN SORT THEN USE THE REST.
40 'WRITTEN FOR LINE PRINTER 4
50 CLEAR 15000
   :DIM A$(50), B$(500), C$(500)
   :CLS
   :GOTO 330
60 POKE 16419, 32
   :CLS
   :PRINT@398, "INSERT DISK TO BE READ IN DR:0
   AND"
   :PRINT@460, "ENTER DISK NUMBER 01, 03, 18, 23,
   43, etc."
70 POKE 15958, 244
    :POKE 15959,245
    :POKE 15960,246
80 PRINT@602,;
    :INPUT N$
90 IF MID$(N$,2,1)="" OR LEN(N$)>2 THEN PRINT@596,
   "TWO DIGITS PLEASE"
    :FOR W=1 TO 500
    :NEXT
    :GOTO 60
100 CMD"D:0"
110 J=2
    :Y=0
    :Z=0
    :C=0
    :P=0
    :FOR Q=0 TO 50
    :A$(Q)=""
    :NEXT Q
120 FOR Z=0 TO J
    :C=0
    :L=15375+(64*Y)+(15*Z)
    :P=P+1
    :IF Y=>1 THEN L=L-15
130 A$(P)=A$(P)+CHR$(PEEK(L+C))
    :C=C+1
    :IF PEEK(L+C)<>32 GOTO 130
140 IF A$(P)=" " GOTO 180
150 B$(DD)=A$(P)+" "+N$
    :C$(DD)=N$+" "+A$(P)
    :DD=DD+1
160 IF LEN(A$(P))=>12 THEN A$(P)="*I GOT IT*"
170 PRINT@(L-15361), STR$(P);"-"A$(P);
    :NEXT Z
    :J=3
    :Y=Y+1
    :Z=0
    :GOTO 120
180 POKE 16419,176
    :GOTO 330
190 CLS
    :PRINT@400,CHR$(23)"SORTING BY NAME"
    :D%=DD
200 CMD"O",D%,B$(0)
210 CLS
    :PRINT@392,CHR$(23)"SORTING BY DISK NUMBER"
220 CMD"O",D%,C$(0)
    :GOTO 330
230 IF MN=7 LPRINT "ALPHABETICAL LIST OF PROGRAMS
   WITH DISK NUMBERS"
240 K=1
    :CLS
    :FOR SS=0 TO DD
    :PRINT B$(SS),
    :IF (SS=50*K) AND (MN=5) THEN K=K+1
    :PRINT
    :INPUT"ENTER TO CONTINUE ";KK;
    :CLS
250 IF MN=7 THEN LPRINT B$(SS),;
260 NEXT SS
    :IF MN=7 LPRINT " "
270 PRINT
    :INPUT"PRESS ENTER FOR MENU  ";KK;
    :GOTO 330
280 CLS
    :PRINT@396, CHR$(23)"PUTTING IT ON DISK"
    :OPEN"O", 1, "DISK/INV"
290 FOR DD=0 TO D%
    :PRINT #1, B$(DD); ","; C$(DD)
    :NEXT DD
    :GOTO 330
300 CLS
    :PRINT@396, CHR$(23)"GETTING IT FROM DISK"
    :OPEN"I", 1, "DISK/INV"
310 FOR DD=0 TO 500
    :INPUT #1, B$(DD), C$(DD)
    :IF B$(DD)="" GOTO 330
320 NEXT DD
330 CLOSE
    :CLS
340 PRINT
    :PRINT
    :PRINT"                M E N U"
350 PRINT
    :PRINT"<1> READ STORED DATA FROM DISK"
360 PRINT "<2> WRITE (STORE) DATA TO DISK"
370 PRINT "<3> SORT BY DISK NUMBER AND NAME"
380 PRINT "<4> READ ANOTHER DISK DIRECTORY"
390 PRINT "<5> DISPLAY BY PROGRAM NAME"
```

(Continued on Page 20)

LISTER/BAS

The continued interest in programs to list BASIC programs prompts us to reprint this article (with minor changes) from our Nov. 1979 Newsletter:

The following Model I/III Disk program was written to allow us to list programs on any printer. Your program must have been SAVEd using the ",A" option (ASCII format). Simply RUN"LISTER/BAS", answer the questions, and you will be able to list programs using any line length, with neat headings and skips at perforations.

```
10 'LISTER/BAS 2.0 - BASIC PAGE LISTER
20 'COPYRIGHT (C) 1979 TANDY CORP.
30 CLEAR 1000
   :DEFINT A-Z
32 '
34 ' ATTEMPT TO OPEN REQUESTED FILE; ERROR TRAP IF
   NOT FOUND
36 '
40 CLS
  :PRINT TAB(20) "BASIC PAGE LISTER 2.0"
  :PRINT
50 LINEINPUT" ENTER FILESPEC: ";F$
60 ON ERROR GOTO 4000
70 ON ERROR GOTO 4100
    :PG=0 'PAGE COUNT SET TO ZERO
71 '
72 '1ST CHARACTER OF LINE IS 0 TO 9 OR 48 TO 57
   DECIMAL
73 'OR FILE IS NOT ASCII!
74 '
80 LINEINPUT#1, L$
    : D= ASC(LEFT$(L$,1))
90 IF D<48 OR D>57 THEN PRINT "* NOT ASCII BASIC
   FILE *"
    : GOTO 50
100 LINEINPUT "     ENTER TITLE: "; TL$
110 LINEINPUT "   TIME AND DATE: "; DT$
120 LINEINPUT "ENTER PAGE WIDTH: "; WD$
130 IF WD$="" THEN WD=64
    : PRINT TAB(17); CHR$(27) WD ELSE WD=VAL(WD$)
140 IF WD<64 OR WD>132 THEN PRINT "* BAD WIDTH (64
    - 132 ONLY) *"
    : GOTO 120
150 INPUT "DO YOU WANT EACH 'STATEMENT' ON A
    SEPARATE LINE";SL$
160 IF LEFT$(SL$,1)= "Y" THEN F1=1 ELSE F1=0
170 PRINT
    : INPUT"TYPE 1 FOR SINGLE SPACE, 2 FOR
    DOUBLE"; SP
172 '
174 '*** MAKE SURE THE PRINTER IS READY ***
176 '
180 PRINT
    : LINEINPUT "READY PRINTER - THEN PRESS ENTER
    >"; A$
190 IF PEEK(14312)>127 THEN PRINT"* PRINTER NOT
    READY *"
    : GOTO 180
200 POKE 16424,67
    : POKE 16425,1 'SET LINE/PAGE AND LINE COUNT
210 GOSUB 500
    : GOTO 260
220 GOSUB 500
230 IF EOF(1) THEN 390
240 LINEINPUT#1, L$
250 IF F1=1 THEN GOSUB 600
    : GOTO 230
260 W1=WD
    : T=0
270 W=W1
    : IF LEN(L$)<W THEN W=LEN(L$)
280 J=INSTR(L$,CHR$(10)) 'CHECK FOR LINEFEED
290 IF J>0 THEN W=J-1 ELSE IF LEN(L$)>W1 THEN
    J=-1
300 LPRINT TAB(T); LEFT$(L$,W)
310 IF J<>0 AND T=0 THEN T=5
    : W1=W1-T
320 IF SP=2 THEN LPRINT" " 'DOUBLE SPACE
330 IF J>0 THEN W=W+1
340 L$=RIGHT$(L$,LEN(L$)-W)
    : IF L$="" THEN 370
350 IF PEEK(16425)>62 THEN LPRINT CHR$(12)
    : GOSUB 500
360 GOTO 270
370 IF PEEK(16425)<63 THEN 230 'PAGE FINISHED?
380 LPRINT CHR$(12)
    : IF EOF(1) THEN 400 ELSE 220
390 LPRINT CHR$(12) 'ALL DONE
400 GOSUB 1000
    : GOSUB 3000
    : PRINT
    : PRINT "* END-OF-LISTING *"
    : CLOSE
    : END
497 '
498 '*** PRINT DASH AND HEADING ***
499 '
500 GOSUB 1000
    : GOSUB 2000
    : RETURN
597 '
598 '*** SINGLE STATEMENT LINES ***
599 '
600 NN=0
    : FOR J=1 TO LEN(L$)
    : V$=MID$(L$,J,1)
    : L=L+1
610 LPRINT V$;
    : IF L>=WD THEN LPRINT" "
    : GOSUB 700
    : LPRINT TAB(10);
    : L=10
620 IF V$=CHR$(34) AND NN=1 THEN NN=0
    : GOTO 640
630 IF V$=CHR$(34) AND NN=0 THEN NN=1
640 IF V$=":" AND NN<>1 THEN LPRINT" "
    : GOSUB 700
    : LPRINT TAB(5);
    : L=5
650 NEXT J
660 LPRINT " "
    : L=0
670 GOSUB 700
680 RETURN
697 '
698 '*** END OF PAGE? ***
699 '
700 IF PEEK(16425)>62 THEN LPRINT CHR$(12)
    : GOSUB 500
710 RETURN
997 '
998 '*** PRINT DASHED LINES FOR PERFORATIONS ***
999 '
1000 LPRINT STRING$(WD,"-")
     : GOSUB 3000
     : RETURN
1997 '
1998 '*** PRINT HEADING AND PAGE NUMBER ***
1999 '
2000 LPRINT TAB(10); LEFT$(TL$+STRING$(30,32),30);
     " "; DT$; " ";
2010 PG=PG+1
     : LPRINT USING "PAGE ###";PG
     : LPRINT TAB(10); STRING$(WD-10,"=")
2997 '
2998 '*** PRINT TWO BLANK LINES ***
2999 '
3000 LPRINT" "
     : LPRINT" "
     : RETURN
3997 '
3998 '*** ERROR TRAPPING ***
3999 '
4000 IF ERR/2+1=54 THEN PRINT"* FILE NOT FOUND *"
     : RESUME 50
4010 IF ERR/2+1=65 THEN PRINT"* BAD FILE NAME *"
     : RESUME 50
4020 PRINT
     : PRINT"ERROR #"; ERR/2+1; "IN LINE"; ERL
     : STOP
4100 RESUME
```

Dirpick

Program by Bud Baker Hutchinson, Kansas

Line 10 of Mr. Baker's Model III Disk Dirpick program indicates that this program is an adaptation from an earlier Model I version written by Barry Kornfeld of New York. We are unfamiliar with Mr. Kornfeld's version.

Dirpick is a straight forward program which reads a disk directory and then allows you to execute any of the programs by pressing the appropriate number (supplied by the computer).

Once you have entered the program, change line 40 as needed. That is, leave it as a REM if you have more than one disk drive on your Model III, or remove REM from the beginning of the line if you have only one drive.

This program will allow you to execute BASIC programs, or it will automatically return to TRSDOS and execute any machine language file with a /CMD extension.

Program:

```
10 'DIRPICK PROGRAM 'MOD I' BY BARRY KORNFELD 190
   WAVERLY PL NY,NY.
20 'MODIFIED FOR 'MOD III' BY BUD BAKER RT3
   HUTCHINSON, KS.
30 CLEAR 1000
    :POKE 16419,32
    :DIM A$(50)
    :J=2
40 REM GOTO 90 'REMOVE REM FOR ONE DRIVE.
50 CLS
    :PRINT@397, CHR$(23)"PRESS DRIVE NUMBER"
    :PRINT@461, CHR$(23)"FOR AUTO DIRECTORY"
60 A$=INKEY$
    :IF A$="" GOTO 60
70 IF VAL(A$)>3 PRINT"   YOU GOTTA BE KIDDING"
    :FOR A=1 TO 500
    :NEXT
    :GOTO 50
80 ON VAL(A$)+1 GOTO 90, 110, 130, 150
90 CMD"D:0"
100 GOTO 160
110 CMD"D:1"
120 GOTO 160
130 CMD"D:2"
140 GOTO 160
150 CMD"D:3"
160 FOR Z=0 TO J
    :C=0
    :L=15375+(64*Y)+(15*Z)
    :P=P+1
    :IF Y=>1 THEN L=L-15
170 A$(P)=A$(P)+CHR$(PEEK(L+C))
    :C=C+1
    :IF PEEK(L+C)<>32 GOTO 170
180 IF A$(P)=" " GOTO 210
190 IF LEN(A$(P))=>12 THEN A$(P)="*TOO-LONG*"
200 PRINT@(L-15361),STR$(P);"-";A$(P);
    :NEXT Z
    :J=3
    :Y=Y+1
    :Z=0
    :GOTO 160
210 PRINT
    :PRINT
    :PRINT"ENTER PROGRAM NUMBER (0 TO RESTART)";
220 POKE 16419,176
    :IF P=>10 GOTO 250
230 X$=INKEY$
    :IF X$="" GOTO 230
240 X=VAL(X$)
    :IF X=0 THEN RUN ELSE GOTO 260
250 INPUT X
    :IF X=0 RUN
260 IF RIGHT$(A$(X),3)="CMD" THEN CMD"I", A$(X)
    ELSE RUN A$(X)
```

Sorting Two Dimensional String Arrays—A Different Approach

Rev. Harry Jansing Louisville, Kentucky

Like Mr. Terrell (March, 1981), I noted with interest the fast sort provided by Allan Emert in the July, 1980 issue.

I used a different approach to allow the sorting by zip code or by mailing addresses. This is perhaps a much simpler procedure than Mr. Terrell's which may appeal to some of you.

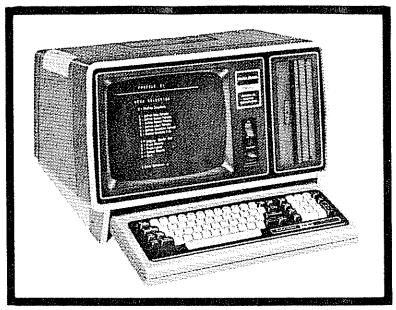
My solution was simply to combine all the various strings—name, address, etc. into a single string with the zip first. The fast sort now works and for printout I use the instring search to return the various items to individual strings.

A simple program can be written to take already existing files and combine them so that the fast sort will work.

Here is a copy of my program:

```
10 'MULTIPLE STRING FAST SORT
20 '
30 CLEAR 5000
40 '
50 ' COMPOSE
60 '
70 FOR X=0 TO 9
80 PRINT
    :PRINT
    :INPUT"NAME";N$
90 INPUT "CODE"; CD$
100 IF LEN(CD$)<>1 PRINT "ONE LETTER ONLY"
    :GOTO 90
110 INPUT"ADDRESS";AD$
120 INPUT"CITY";C$
130 INPUT"STATE";S$
140 INPUT"ZIP CODE";Z$
150 IF LEN(Z$)<>5 PRINT"FIVE DIGITS PLEASE"
    :GOTO 140
160 N$(X)=Z$+CD$+N$+"/"+AD$+"#"+C$+S$
170 NEXT X
180 '
190 '-----------------------------
200 ' INSERT 'FAST SORT' HERE
210 '-----------------------------
220 '
230 ' DECODE PROGRAM
240 '
250 FOR X=0 TO 9
260 FOR I=1 TO LEN(N$(X))
    :IF MID$(N$(X),I,1)="/" THEN 270 ELSE NEXT I
    :GOTO 340
270 FOR II= I TO LEN(N$(X))
    :IF MID$(N$(X),II,1)="#" THEN 280 ELSE NEXT II
    :GOTO 340
280 Z$=LEFT$(N$(X),5)
290 CD$=MID$(N$(X),6,1)
300 N$=MID$(N$(X),7,II-I)
310 AD$=MID$(N$(X),I+1,II-I-1)
320 C$=MID$(N$(X),II+1,LEN(N$(X))-II-2)
330 S$=RIGHT$(N$(X),2)
340 CLS
    :PRINT N$
    :PRINT CD$
    :PRINT AD$
    :PRINT C$
    :PRINT S$
    :PRINT Z$
350 PRINT
    :INPUT "PRESS ENTER FOR NEXT";
    :NEXT X
```

Note: Since Rev. Jansing uses "#" to separate address and city, you should avoid using "#." Instead of APT #5 use APT 5, etc.



Model II

Product Line Manager's News

Three new software packages for Model II have recently been released that I consider to be all parts of one family. One is available at the time I am writing this and the other two should be available by the time you are reading it. ReformaTTer™ is an easy to use program specifically designed to handle transfer of files between Model II TRSDOS and IBM® diskettes.

ReformaTTer automatically organizes data to conform to IBM format (3741 single-density) from TRSDOS format as well as performing conversion between EBCDIC and ASCII character sets. It will automatically initialize your diskettes to the IBM format with all necessary fields required, eliminating the need for pre-initialized diskettes.

You have full control over all aspects of an IBM diskette with ReformaTTer. Creation/expiration date, record length, beginning of extent, end of extent, write protect, accessibility, bypass and verify fields can all be modified. Also, any 'data set' label can have its contents altered, the 'data set' can be renamed and deleted 'data sets' can even be reactivated. ReformaTTer allows you to examine and alter existing data on an IBM diskette directly without transferring the data to a TRSDOS diskette so you save time and money.

The nine major functions of the ReformaTTer program are:

1. IDENTIFY — Display/Alter volume ID of IBM diskette.
2. REDEFINE — Modifies parameters of an existing data set on an IBM diskette.
3. DELETE — Rename and reset an active data set to inactive.
4. DIRECTORY — Displays contents of an IBM or TRSDOS diskette's directory.
5. EDIT — Modifies the contents of existing records of an active data set.
6. DUMP — Display the contents of specific sectors of an IBM diskette.
7. DISPLAY — Display the contents of existing records of an active IBM data set.
8. TRANSFER — Transfer data from a TRSDOS file to an active data set or transfer data from an active data set to a TRSDOS file. Also, TRSDOS to TRSDOS and IBM to IBM data transfers are permitted.
9. INITIALIZE — Structure a diskette in IBM 3740 format.

ReformaTTer requires a TRS-80 Model II computer with 64K of memory and two or more disk drives. The catalog number is 26-4714 and the suggested retail price is $249.00.

The next two programs both deal with BiSync (Binary Synchronous) Communications. The first is an emulator for the IBM 3270. Just look at what you can do with our Binary Synchronous Communications — 3270 program and your TRS-80 Model II. Communicate with IBM Systems 360/370 and 30 series central processing units or any non-IBM devices which are equipped with BSC 3270 communications capability.

This powerful package provides all functions of an IBM 3270/ 3271/ 3275/ 3277 display station including: Screen Formatting, Polling Responses, Data-Link Control, Time-Out Control and Cyclic Redundancy Checksum. It is fully interactive with IBM remote programs such as TSO, CICS, VM/CMS, IMS or others which support IBM 3271 or 3275 terminals. The catalog number is 26-4715 and it sells for a suggested retail price of $995.00.

The last of the family is a BiSync package to emulate the IBM 3780.

It is a comprehensive program designed to allow your TRS-80 Model II computer to function as a Remote Job Entry (RJE) terminal. You may select the use of IBM 2770/ 2780/ 3780/ 3741 protocols and communicate with IBM System 360/370, 30-Series, IBM 2780/3780 terminals, DEC PDP-11, VAX-11 or other devices equipped with binary synchronous communications capabilities.

Commands and options are provided which allow you to: send one or a series of files to a remote, receive data from the remote onto a printer or disk files, automatically convert between EBCDIC and ASCII codes, transmit or receive character coded data in transparent or non-transparent modes, automatically pad or truncate records to match transmitted data to required protocol and printer forms control decoding.

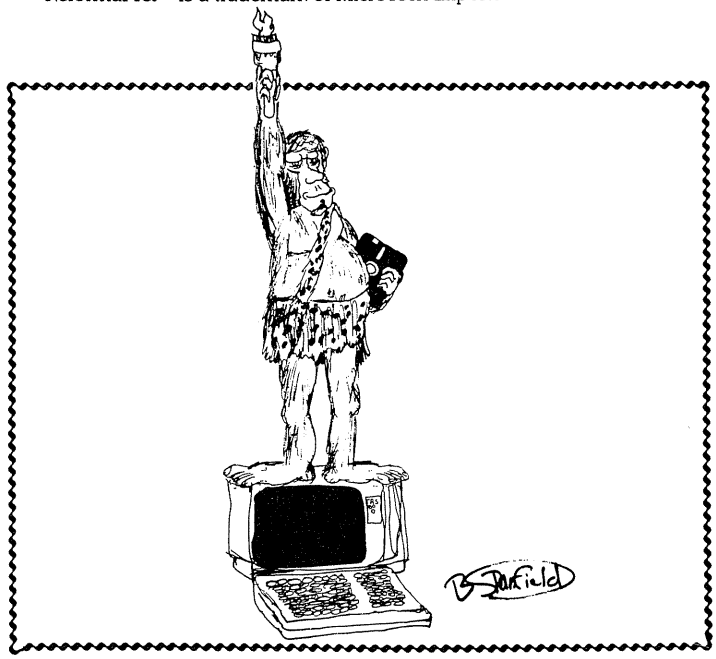
The catalog number for this package is 26-4716 and sells for a suggested retail price of $995.00.

The ReformaTTer and BiSync Communications programs require a TRS-80 Model II computer with 64K of Memory. Synchronous communications are thru the 'A' serial port of the TRS-80 and may operate at data rates up to 19,200 baud (this may vary with length and type of communications connection used). Only half-duplex communications facilities are required. Installation is required for the BiSync packages but not for the ReformaTTer.

The Reformatter and Binary Synchronous Communications programs are sold on a non-exclusive, paid up license for use on multiple CPU's owned by a single purchaser. Title to the media on which the software is recorded is transferred to the purchaser, but not title to the software. The purchaser may reproduce the number of copies of the Binary Synchronous Communications programs as required for their own personal, business or other use. Each copy shall include Radio Shack's copyright notice.

Prices are suggested list only and may vary at individual stores.

IBM® is the registration of International Business Machines, Inc.
ReformaTTer™ is a trademark of MicroTech Exports.



Model II Bugs, Errors, and Fixes

Model II BASIC Interpreter (26-4910)

Model II BASIC always sends a carriage return/line feed after 132 characters (TRSDOS 1.2 and 1.2a) or 255 characters (TRSDOS 2.0 and 2.0a). This is undesirable to some users. In order to negate this feature of BASIC, the following patches should be applied:

```
PATCH BASIC A=3584 F=D4BA57 C=000000
PATCH BASIC A=3588 F=D4BA57 C=000000
PATCH BASIC A=56F8 F=CC0D57 C=C30157
```

This patch should be applied only if a user is writing his/her own software and is experiencing this automatic action as a problem. Under NO circumstances should this patch be applied for use by any Radio Shack software packages.

Model II Capacity Corrections

Model II Three Disk Accounts Receivable (26-4604) will store up to 800 accounts and 2500 transactions, not 1500 accounts and 2500 transactions as stated in our current computer catalog (RSC-5).

Model II Three Disk General Ledger (26-4601) will hold up to 400 GL accounts and 4300 year-to-date transactions, not 500 accounts and 5000 year-to-date detail entries as stated in RSC-5.

Notes on Previous Newsletters

November, 1980

Mr. Robert Jacobs, Associate Professor of Political Science at Central Washington University, sent this suggestion:

The Data Base Manager Program recently published (Nov., 1980) in the *News* can be enhanced as follows:

Edit 860: `860 LINEINPUT"?"; A$(I)`

Add 865: `865 IF A$(I)="" THEN A$(I)=B$(I)`

These changes allow the use of commas and colons in the data entered; otherwise they act as delimiters. Line 865 is added to restore the default record if A$(I) is blank, since otherwise the LINE INPUT function wipes it out.

November, 1980

Mr. Lawrence Basener of Belmond, Iowa sent us some additional corrections for Budget Management (26-1603):

Mr. Basener suggests changing lines 3530 and 3540 of the REPORTS program to read:

```
3530 GOSUB3900
     :LPRINT TAB(20);"BUDGET MANAGEMENT REPORT" + STRING$(16," ");
     :LPRINT USING"PAGE ##";PG
3540 LPRINT TAB(28)"** " H$ " REPORT **"
     :LPRINT TAB(33)"DATE: ";
     :D=DT
     :GOSUB 500
     :LPRINT USING"##/##/##"; D1; D2; D3
```

Mr. Basener points out that if these lines are not edited as shown, the printer will take 2 lines instead of 1. Since the second line results from wrap-around, it is not interpreted by the computer as an extra line feed, and this can cause the printer to skip lines between pages at the wrong places.

Attention CP/M Users

If you are using CP/M in a Model II, you may be experiencing unexplained BDOS Errors. These errors have been traced to a problem within the disk controller chip (a microcode fault, see View from the 7th Floor, March 1981). We have programmed around this fault in our current Model II TRSDOS versions 1.2a and 2.0a. The BDOS Error is an Operating System problem; there is no hardware modification which will cure it.

We have offered full information on how to fix this problem to Digital Research who directed us to their various dealers. At this time we've given the information to Pickles and Trout and Lifeboat Associates. Our information is that both of these companies have released new versions of their CP/M implementations for the Model II which correct the BDOS error problem. We have also contacted other vendors of CP/M and have offered to provide full details to them to fix their implementation of CP/M.

We cannot accept responsibility for problems in Operating Systems other than TRSDOS. The "bottom line" is that if you are using CP/M on the Model II, and have encountered unexplained BDOS errors — contact the vendor that you obtained CP/M from and request a new release.

SO YOU WANT TO PRINIT?

Linda Miller

Have you ever wanted to simplify computer operation by eliminating the necessity of answering the prompts when using the FORMS command? On boot up would you like for your printer to be automatically initialized to your own predetermined printing format? If you answered yes to either of these questions then the Supervisor Call PRINIT may be the answer. Disregarding previous reservations about machine language we reach for the *Model II Owner's Manual*, 2.0 version and turn to the section on Supervisor Calls or SVCs (page 216). Please note two items under "How to Use the Supervisor Calls."

1. Each SVC has a function code.
2. RST 8 is used with every SVC.

Under the subheading "Calling Procedure" we learn that the A register is loaded with the SVC function code and other registers are loaded as indicated by the specific SVC. What, you might ask, is a register? Simplistically speaking it is place in the CPU that holds a value or values to be manipulated.

On page 243 we find PRINIT, an acronym for Printer Initialization, designated as function code 17. Note the letters A-D under the heading "Entry Conditions." These represent registers A,B,C, and D. By using the assembly language load (LD) instruction, the registers can be loaded with values set by the user. For example, register B will be loaded with a number which represents the overall length of the page. The following lines indicate what the values loaded into each of the four registers (A-D) represent.

```
LD A, 17   ;17 is the function code
LD B, n1   ;n1 is the page length
LD C, n2   ;n2 is the number of printed lines per page
LD D, n3   ;n3 is the number of characters per line
```

To complete the program add:

```
RST 8      ;jump to the SVC (all SVCs use this)
RET        ;returns to TRSDOS
```

The next step is to translate all of these instructions to Hex. For the Hex values of this program refer to Appendix E of *How to Program the Z-80* (Cat. No. 62-2066). Here we find "LD A, n" which means "Load the A register with a number (n)." The Hex value for this instruction is 3EH. Therefore we arrive at the following:

(Continued on Page 11)

PRINIT (From Page 10)

Instruction	Description	Hex Value
LD A,n	Load register A with a number (n)	3E ØØH*
LD B,n	Load register B with a number (n)	Ø6 ØØH
LD C,n	Load register C with a number (n)	ØE ØØH
LD D,n	Load register D with a number (n)	16 ØØH
RST 8	Jump to the SVC	CFH
RET	Returns to TRSDOS	C9H

H* The H indicates that these are Hex numbers.

Next determine the numbers which will be loaded into the registers. For this example the page length is 40 lines, the number of printed lines per page is 30, and the line width is 80 characters. Decimal numbers 17, 40, 30, and 80 are converted to Hex.

17 decimal (function code)	equals 11 Hex
4Ø decimal (page length)	equals 28 Hex
3Ø decimal (printed lines per page)	equals 1E Hex
8Ø decimal (characters per line)	equals 5Ø Hex

Combine the Hex values for the program instructions and the above Hex values.

Instruction	Value	Computer Interpretation
LD A, 17	3E 11	Load the A register with 11 hex
LD B, 4Ø	Ø6 28	Load the B register with 28 hex
LD C, 3Ø	ØE 1E	Load the C register with 1E hex
LD D, 8Ø	16 5Ø	Load the D register with 5Ø hex
RST 8	CF	Jump to the SVC
RET	C9	Return to TRSDOS

The resulting code is:

```
3E 11 Ø6 28 ØE 1E 16 5Ø CF C9
```

The program can now be loaded into memory using DEBUG. At "TRSDOS Ready" type DEBUG ON. When the prompt "DEBUG is now on" returns, type DEBUG. Answer the "?" by pressing the (M) key, and "M= " appears. If you accidentally hit a key other than (M) and need to clear it press the (ESC) key and the "?" reappears. With "M= " showing type "FØØØ" (an arbitrary address in high memory where the program will start). Press the (F1) key, and the cursor moves into the memory display area where the above ten byte (two hex digits represent one byte) machine code for PRINIT is keyed in. The code is grouped one byte at a time (two hex digits, remember?) and a space is left between each byte. If you make an error use the left, right, up, or down arrow to move to the position where the code needs to be changed and reenter the code. Note the beginning and ending addresses (FØØØ and FØØ9 respectively). Now press (F2) (0) and "TRSDOS Ready" reappears. (Refer to pages 86-92 in the TRSDOS section of the *Model II Owner's Manual* for additional information on DEBUG.) The program name is FORMS/CIM. FORMS because it seems appropriate and the extension CIM (*C*ore *I*mage *M*odule) which refers to a program stored in memory and on disk with the same image. Although the CIM extension is not mandatory, it is appropriate for this program. Type:

```
DUMP FORMS/CIM START=FØØØ END=FØØ9 (ENTER)
```

The program has now been dumped to disk and stored in the directory under the file name FORMS/CIM. If you want to print a document using these parameters, type FORMS/CIM at "TRSDOS Ready". To get the same printing format each time you boot up with this disk, type AUTO FORMS/CIM at TRSDOS Ready.

The applications of SVC PRINIT are more extensive than can be explored in a short article. Different printing formats could be dumped under different file names on the same disk. These individual files could in turn be loaded into memory depending on the desired printed output.

Multiple Programs in FORTRAN

Robert G. Minty Birmingham, Michigan

I am the owner of a Radio Shack 64K Model II micro with the Model II FORTRAN package version 3.36. (This is the latest version of FORTRAN on TRSDOS 2.0a. If you own an earlier version, it is available at no charge as part number 700-2017, available through your local Radio Shack store.) I purchased this computer and software to do scientific and engineering computing tasks, and have been very happy with the performance and reliability of this equipment.

The most important application that I bought this computer for is a trial-and-error procedure for designing a complex mechanism used in the automobile industry. I currently have a program to do this job running on a large-scale Honeywell time-sharing computer. Knowing that even at 64K I was going to be forced to divide the program up into functional blocks to run, I proceeded to enter the program coding into my Model II, and was able to obtain excellent results.

The problem was that in order to carry information from program to program, it was necessary to write large volumes of numbers out to the disk before leaving one program, and subsequently read the information back in after loading the next sequential program. The problem was that reading and writing several thousand floating-point variables is a time-consuming task indeed, involving some five minutes of "dead" time between the execution of my programs. After initializing the data set and title data, my procedure is to follow a closed loop consisting of data set manipulation, major calculations associated with the design, and design analysis, and return to data set manipulation.

Since the actual calculations and output require only about five minutes of run time, it was understandably very aggravating to have fifteen minutes of disk read/write time superimposed on the loop, so that it took twenty minutes to complete one cycle. The complete job would normally require from twenty to one hundred cycles.

What I really needed was some way to "chain" the programs together so that data and information generated by one program would not be destroyed when the next program was loaded.

The procedure itself is quite simple, and requires virtually no knowledge of machine-level programming or computer architecture.

Long FORTRAN programs that will not fit into core directly are first broken down into individual task-oriented programs. Typically, these tasks might consist of data input, display and/or print the data set, alter the data set, execute one or more calculation procedures, output calculated results to the screen and/or printer, analyze the results and print or display results of the analysis, and final documentation.

After breaking the program down into functional "modules," the programmer should determine what variables must be carried forward from program to program. The variables are listed in a FORTRAN "COMMON" statement, which must appear as the first COMMON statement in every program module. While the variable names need not be duplicates in all the common statements, each variable in one common statement must correspond exactly with its counterpart in every other common statement in "byte" length.

Each program "module" is then compiled individually, using F80. It is best to produce a listing file at this point, and check the length, in bytes, of the common statement as reported at the end of the listing file. Using labeled common will assist in the identification of the common area in the listing map.

The common area length in every program must correspond

(Continued on Page 12)

FORTRAN (From Page 11)

exactly, and the reported length (hexadecimal) should be recorded for future use.

The relocatable version of each program (filename/REL generated by F80) is then linked with any user-supplied subroutines, functions, etc. and loaded into core. This is done using L80 (Linking Loader) and the −E switch (link and load but do not run — Note that the −N switch is not used). At this time, the two hexadecimal numbers (contained within square brackets) that are displayed on the video screen should be recorded for future use.

At this point you should be in TRSDOS, with `TRSDOS Ready` on the video. We now DUMP the program to a file. The filename should be selected (this will be a runtime machine-language program), and the other DUMP parameters must be supplied as shown below:

`START=` the sum of X'3000' and the hexadecimal COMMON length recorded earlier. The result should be a hexadecimal value.

`END=` the rightmost of the two hexadecimal addresses that were displayed between the square brackets when L80 was run.

`TRA=` the leftmost of the two bracketed addresses.

`RELO=` the same as the START address.

`RORT=T`

For additional information on the DUMP command, see your Model II Owner's Manual.

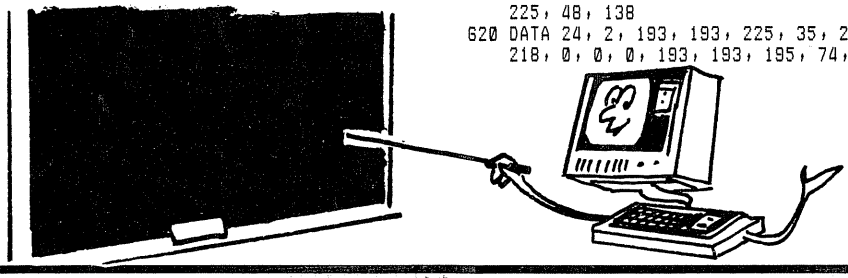
When this DUMP instruction is entered, the program will have been "Saved" on the disk under the name selected in the DUMP command.

Every program module should be linked, loaded, and DUMPed following this procedure with the single exception of programs having a "BLOCK DATA" subprogram. BLOCK DATA subprograms may be incorporated into a program to load tables, data set titles, etc. into the COMMON area, but only one such program is permitted, it must be linked and loaded normally (i.e. using the standard L80 procedures, not the procedure described above) and it must be run before any other of these special programs are run, since running the BLOCK DATA subprogram will overlay the COMMON area variables (other than those that it is defining) with "junk."

When all of the program modules have been dumped, the programs may be run. If the logic of the total program requires a fixed sequence of running, the programs may be set up in a command "DO" file that will run each program in turn or, alternatively, each program may set and/or clear control flags stored in the COMMON area that may be tested for compliance with any logical restrictions.

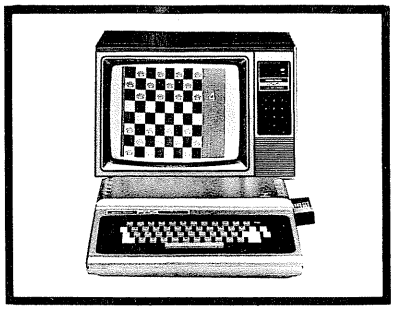
By using the above procedure, I was able to completely eliminate the time-consuming disk read/writes that I found to be so aggravating. Despite a three minute major-block execution time on the TRS-80 vs. six seconds on a large-scale Honeywell 6000 time-sharing computer, the total turn-around time to change the data, run, and evaluate comes out to be about five minutes on both machines, because of the sometimes long "waits" when a lot of users are on the Honeywell System.

For those of us who want to run very large programs, but can't afford the cost of a large computer, this method seems quite appropriate and convenient.



Program Directory (From Page 5)

```
280 FOR J=1 TO N STEP 4
290 IF I$="S" THEN PRINT A$(J), A$(J+1), A$(J+2), A$(J+3)
    :GOTO 310
300 LPRINT A$(J);
    :LPRINT TAB(20) A$(J+1);
    :LPRINT TAB(40) A$(J+2);
    :LPRINT TAB(60) A$(J+3)
310 NEXT J
320 IF I$="P" THE LPRINT CHR$(12)
330 PRINT"TH-TH-TH-THAT'S ALL FOLKS."
    :I$=INKEY$
    :I$=""
340 I$=INKEY$
    :IF I$="" THEN 340 ELSE END
350 REM SETUP AND CALL STRING SORT RTN WHICH WAS WRITTEN BY JOHN
    C. HALLGREN AND PUBLISHED IN THE JANUARY '81 ISSUE OF THE
    TBUG 80 NEWSLETTER, P.O. BOX 787, RUSKIN, FL. 33570
360 U0(9)=UN
    U0(29)=(U0=235)*-235
370 U0(37)=UP AND 255
    :U0(42)=UL AND 255
    :U0(83)=VARPTR(U0(15))
    :U0(86)=VARPTR(U0(3))
380 U0(18)=UA
    :U=VARPTR(U0(0))
390 DEFUSR0=U
    :REM FOR LII BASIC, POKE 16526,U AND 255: POKE 16527, U/256
400 U=USR0(UN)
    :RETURN
410 REM LOAD SUBRTN INTO NUMERIC ARRAY U0
420 DEFINT U
    :DIM U0(86)
    :FOR U=0 TO 86
    :PRINT@247, U;
    :READ UL, UH
    :U!=256*UH+UL
430 UN=UN+ UH+ UL
    :U0(U)=(U!>32767)* 65536+U!
    :NEXT
440 IF UN=20631 RETURN ELSE STOP
450 DATA @00, TRSDOS2.3
460 DATA @02, ACEYDUCE, ASTEROID, BSKTBALL, BIBLE, BINARYL,
    BINARYQ, BIORYTHM, BLACKBOX, CAMEL, CHASE, CLOCKCTR,
    CONECTDT, CHKBK, FIXEDT, TBUGHI, TEL
470 DATA @03, ACCTSREC TARANTO
480 DATA @05, MAIL-V (MAILING LIST)
490 DATA @07, CORRAL, CRAPS, DARTS, DAYOFWK, MAN, DIMRTRNS,
    DODGEM, ELIZA, GUNNER, HEXPAWN, XREF1, XREF2, XREF3, GOLF
500 DATA @08, KING, LANDER, LIFE, LIFEXPCT, MADLIB, MAGAZINE,
    MANEATER, MSTRMIND, MNSTRMAZ, OBSTACLE, RNDSTSND, ROADRACE,
    SAMEDIFF, SIMON, SORTINT, SORTZKEY, SNDSKTCH
510 DATA @99
520 DATA 205, 127, 10, 229, 221, 225, 221, 229, 209, 203, 58,
    203, 27, 213
530 DATA 221, 225, 175, 33, 160, 15, 237, 82, 122, 179, 200, 229,
    33, 0
540 DATA 0, 229, 225, 229, 229, 253, 225, 1, 173, 58, 93, 84, 41,
    25, 9
550 DATA 229, 253, 229, 225, 221, 229, 209, 25, 93, 84, 41, 25,
    9, 235, 225
560 DATA 0, 0, 229, 213, 14, 0, 235, 26, 190, 56, 2, 12, 126,
    197, 0, 14, 1, 0
570 DATA 145, 48, 3, 193, 24, 39, 60, 6, 255, 0, 184, 48, 1, 71,
    35, 19, 126
580 DATA 35, 102, 111, 235, 126, 35, 102, 111, 13, 40, 4, 19, 35,
    24, 249
590 DATA 120, 193, 71, 26, 190, 56, 10, 32, 38, 19, 35, 16, 246,
    203, 9, 48
600 DATA 30, 209, 225, 6, 3, 235, 78, 26, 235, 113, 18, 19, 35,
    16, 246
610 DATA 253, 229, 225, 221, 229, 209, 175, 237, 82, 229, 253,
    225, 48, 138
620 DATA 24, 2, 193, 193, 225, 35, 209, 213, 229, 175, 237, 82,
    218, 0, 0, 0, 193, 193, 195, 74, 72
```



Color Computer

Product Line Manager's News

As I told you in the last issue of the Microcomputer Newsletter, we will go into a more in-depth explanation of the CIRCLE command.

First, the syntax (command structure) for the instruction CIRCLE:

```
CIRCLE (x,y),r,c,hw,start,end
```

x and y denote the x,y coordinates for the center of the circle to be drawn. It should be somewhere on the screen (i.e. x=0–255; y=0--191).

r is the radius of the circle (distance from the center to the edge).

c determines the color to be used. You need to select one of the colors available from the color set you are using. If no color is specified, then the foreground color is used.

hw is the height to width ratio determined by a number between 0 and 255. If no number is specified, 1 is used.

start is just what it says, the starting point of the circle. It is a number between 0 and 1. (Yes, it can be a decimal value; i.e., .2, .5, .8, etc.). If no point is stated, 0 is used.

end is the ending point of the circle. It is also a numeric value between 0 and 1. (It can also be a decimal value.) If no point is stated, 1 is used.

Right up front, let's say that we aren't going to get into the argument that circles don't have starting and ending points because they are round. I'll even admit that circles don't have sides either . . . well, wait till you start using this command.

Let's start off nice and simple . . .

Remember to set up the graphics screen for display by entering the following commands:

```
10 PCLEAR 4      'CLEAR 4 pages of memory
20 PMODE 3,1     'Select res. mode 3
30 PCLS          'Clear the first page
40 SCREEN 1,1    'Select graphics screen
999 GOTO 999     'end to save display
```

Now enter this line and RUN the program we've written so far:

```
50 CIRCLE (125,95),50
```

What we have done is picked a center point near the center of the screen (125,95) and instructed the computer to draw a circle 100 units in diameter. (If the radius is 50, then the diameter has to be 100, right?)

In this resolution mode, we have a buff background (mine looks kind of off-white) with an orange circle in the center section of the screen. Just press (BREAK) to get out of the program, 'cause we're going to do some more fancy things.

Try re-typing line 50 and add: ,7 to the end. The new line 50 should look like this:

```
50 CIRCLE (125,95),50,7
```

Now run the program. What has happened is you have specified the color of the circle instead of letting the computer use the foreground color for the color of the circle. The color 7 is magenta on my screen, a kind of reddish color. This same statement using the number 6 will give you a cyan circle instead. The number 5 will work also with one minor exception. The circle drawn when the number five is used is the same color as the background and therefore is invisible to the naked eye. The number 8 will bring back the orange circle that we originally had. (Right, Watson, the foreground color in this color set is orange!)

If this series of colors is not to your liking, change line 40 to:

```
40 SCREEN 1,0
```

Run the program again. This time you will get a light green background with either a red, blue, or yellow circle. Since this color set was selected in the SCREEN statement, the color you specified in the CIRCLE statement uses the corresponding color in this alternate color set. If you select color #8 in the CIRCLE command and color set 1, then the circle will be orange. If you select color #8 in the circle command but use the color set 0, then the circle is the complementary color red. Colors 1, 2, 3, 4 (green, yellow, blue, red) and 5, 6, 7, 8 (buff, cyan, magenta, orange) respectively, are complementary colors.

Now, back to the CIRCLE command. I have elected to stay with the SCREEN 1,1 command. You are welcome to use whichever color set pleases you most. (Your colors will be different from those in the text following but all the other features will remain the same.)

Continue with the program by changing line 50 to read:

```
50 CIRCLE (125,95),10,8,,5
```

The circle is still located at the approximate center of the screen, but now has a diameter of 20 points, a color of orange (or red), with a height to width ratio of .5. The height to width ratio means the circle will be wider on the horizontal plane than it is tall on the vertical plane. Run the program. We changed the diameter of the circle to make it smaller since the width of the circle is the radius we specified in the CIRCLE command. If we changed the value of .5 to 1.5 (try it), the circle would be taller than it is wide. It would barely have fit on the screen with a radius of 50. If the height/width ratio were 2.5 (with the radius of 50), the top and bottom sections of the circle would have gone off of the screen!

Note: when you don't specify the color to be used, the comma which separates the position where the color selection goes must still be included.

The last little choice we can make with the CIRCLE command is whether or not to make the circle completely round or to just have a partial circle drawn. Enter these lines in place of the old lines:

```
20 PMODE 4,1    'change to higher res.
50 CIRCLE (125,95),50,7,1,0,,85
```

Run the program. Here is what we did. In line 20, we switched to the highest resolution mode (4). In line 50, we put the center point at the same location as before, gave it a radius of 50 again, changed the color of the circle to one of the colors available (res. 4 has only 2 colors from each of the 2 color sets), specified a round circle (1), and told the computer to start drawing the circle at the 3 o'clock position and continue to draw it until approximately 1 o'clock. If we had left in the previous height/width ratio, the same thing would have happened to the circle, except the incomplete circle would have been oblong instead of round.

Note: If you are going to use the start/end portion of the CIRCLE command, you must also use the height/width portion of the command. For a normal circle, again, this is 1.

Trivia to remember: the circle draws starting at the 3 o'clock position (both AM and PM on one of those old circular analog clocks). It draws the circle, whether round or oblong, in a clockwise direction. It can be drawn anywhere on the screen and can have an edge falling off of the screen. It is very fast, simple to use but very powerful. Chapter 5 in the Going Ahead with Extended BASIC manual will give you more information concerning the CIRCLE command as well as some sample programs to try out. All of the numbers for x, y, r, c, hw, start, and end can be numbers or variables. You can use variables to show changing shapes or at different locations on the screen with different sizes or whatever your imagination can dream up.

Binary Search

Dan Belemecich Jamestown, California

Once a fast sort has organized your data use this simple binary search to locate records. Listing 1 is the actual search algorithm, listing 2 is a test program.

The sorted array contains 500 records.

The variable 'N' needs to be slightly larger than the array and 'N1' should be about half of 'N.'

In this example, we are searching a numeric array, S(n). To use this technique with a string array, change the search variable to W$ and the array to S$(n).

LISTING 1

```
1' THE SORTED ARRAY FOR THIS SAMPLE
2' LISTING WOULD BE DIM S(550)
3'
10 INPUT"SEARCH KEY";W
20 N=540
    :N1=250
    :N2=0
30 FOR I=0 TO 9
    :IF W>S(N1+1) THEN 40 ELSE 50
40 N2=N1
    :N1=INT((N1+N)/2)
    :NEXT
    :GOTO 90
50 IF W=S(N1+1)THEN 80
60 N=N1
    :N1=INT((N1+N2)/2)
    :NEXT
    :GOTO 90
70 '
80 ' THE SEARCH IS SUCCESSFUL, S(N1)
81 ' IDENTIFIES THE DESIRED RECORD.
82 '
90 ' THE SEARCH IS NOT SUCCESSFUL
95 '
```

LISTING 2

```
10 DIM S(550)
    :FOR I=0 TO 500
20 S(I+1)=I+1
    :NEXT
30 INPUT"VALUE TO SEARCH FOR";W
40 N=540
    :N1=250
    :N2=0
50 FOR I=0 TO 9
    :IF W>S(N1+1) THEN 60 ELSE 70
60 N2=N1
    :N1=INT((N1+N)/2)
    :NEXT
    :GOTO 150
70 IF W=S(N1+1) THEN 100
80 N=N1
    :N1=INT((N1+N2)/2)
    :NEXT
    :GOTO 150
100 PRINT S(N1+1)
    :GOTO 30
150 PRINT"NOT FOUND"
    :STOP
    :GOTO 30
160 '
```

Screen Input/Correction Routine

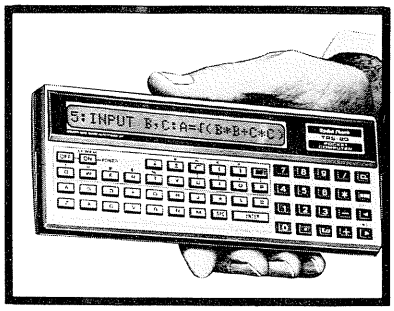
W. Robert Davis Santa Rosa, California

For some programs it is convenient to have input statements and the input values remain on the screen. Also, it is desirable to be able to correct an input value without re-running the program.

Here is a routine I have found to be very useful. A call for error correction is placed following the input section. If not called, pressing (ENTER) causes the message to be erased and the next program line will appear in its place. If called, the error routine erases the input section of the display and re-displays the first input statement along with the previous input value.

Note for Model II readers: If you would like to try this routine on your Model II, change CHR$(27);CHR$(30) to CHR$(11);CHR$(23) in lines 50 and 1015.

```
10 CLS
    : PRINT TAB(20) "DATA INPUT IS NEXT"
15 'PRINT"PREVIOUS"
    : PRINT"VALUE"
20 PRINT"P= $";P; TAB(15);
    : INPUT"PRINCIPAL $";P
30 PRINT"I= %";I; TAB(15);
    : INPUT"INTEREST %";I
40 GOSUB 1000
    : IF FLG=1 THEN 50 ELSE 100
50 FOR ER=1 TO 2
    : PRINT CHR$(27); CHR$(30);
    : NEXT
    : FLG=0
    : GOTO 20
100 'NEXT PROGRAM LINE
999 END
1000 PRINT
    : E$=""
    : PRINT"* ERROR?, TYPE 'E' ELSE PRESS 'ENTER'"
1010 E$=INKEY$
    : IF E$="" THEN 1010 ELSE IF E$="E" THEN 1020 ELSE EE=2
    : GOTO 1015
1015 FOR ER=1 TO EE
    : PRINT CHR$(27); CHR$(30);
    : NEXT
    : RETURN
1020 PRINT
    : PRINT"PRESS 'ENTER' WHERE NO CHANGE REQUIRED"
1025 FLG=1
    : EE=4
    : FOR DL=1 TO 1500
    : NEXT
    : GOTO 1015
```



Pocket Computer

Product Line Manager's News

This month we would like to begin a series of articles which will bring you useful programs for your pocket computer, along with discussions of certain technical aspects of the pocket computer which we think you may find interesting.

Last month we introduced you to the new Pocket Printer. There are a few additional facts about the printer which we would like to give you this month. First, none of the Radio Shack software which is currently available was written to run with the printer.

When you slide your Pocket Computer into the printer, and set the printer switch to ON, press (BREAK) twice, the computer "recognizes" that the printer is present. As long as both the POWER and PRINT switches are in the ON position, and the printer is attached to the Pocket Computer, ALL PRINT commands will be routed to the printer instead of the LCD display. Further, the computer will not stop after each PRINT command as it does when the printer is not present or the "PRINT" switch is in the OFF position. PAUSE commands will still be sent to the LCD display, and will work as usual. If you instruct the Pocket Computer to LIST a program while the printer is connected and ON, the entire program will be listed on the printer. LIST with a line number (LIST ln) will still be listed on the LCD display.

What this means if you are running a Radio Shack program which uses (SHFT) (SPC) to access a menu, and the printer is attached and ON, is that the computer will enter a continuous loop which consists primarily of PRINT statements followed by a GOTO the beginning of the menu. The result will be a continuously repeated printout of the program menu until you stop the printout by using the PRINT ON/OFF switch on the printer or the (CA/BREAK) key on the pocket computer (also known as the (ON) key). If you don't need yards of the same menu printed, print one or two copies and then stop the printer.

The menus in our programs will give you the most problems when you try to use them with the printer. The remaining parts of the programs should work with the printer without disastrous results. The printouts may not be "pretty" or well formated, but they should be usable. Refer to your printer manual for information on how the printer will handle large numbers. Our current software was written before we knew all the details about the printer, so we were unable to format the information for use both with and without the printer. You can expect that much of our future Pocket Computer software will be written to use the printer where appropriate.

Enough technical information, here is a program which will calculate the inverse of a square matrix with a size up to 10 x 10 (so we are told, I only checked it with matrices up to 4 x 4):

Inverse of a Matrix

```
10 "A":INPUT "ORDER ? ";A
20 I=10: FOR E=1 TO A
30 FOR F=1 TO A
40 PAUSE USING "####";E;F: INPUT G
50 A(I)=G: I=I+1
60 NEXT F: NEXT E
70 END
80 "S": FOR B=1 TO A
90 GOSUB 500
100 A(H)=A(H)+1
110 NEXT B
120 GOSUB 500
130 D=A(H)-1
140 IF D=0 PRINT"NO SOLUTION": END
150 GOSUB 600
160 B=B-1
170 IF B>0 GOTO 120
180 FOR B=1 TO A
190 GOSUB 500
200 A(H)=A(H)-1: NEXT B: BEEP 3
210 "B": I=10: FOR E=1 TO A
220 FOR F=1 TO A
230 PRINT USING "####";E;F; USING "######.####"; A(I)
240 I=I+1
250 NEXT F: NEXT E
260 END
270 "F": PAUSE "CORRECT": INPUT "I= ";E: INPUT "J= ";F: INPUT G
280 GOSUB 510
290 A(I)=G: GOTO 270
500 C=B: GOTO 520
510 C=F
520 C=C-A+9
530 H=A*B+C: I=A*E+C
540 RETURN
600 FOR F=1 TO A
610 GOSUB 510
620 A(H)=A(H)/D: NEXT F
630 FOR E=1 TO A
640 IF B=E GOTO 690
650 GOSUB 500
660 D=A(I): FOR F=1 TO A
670 GOSUB 510
680 A(I)=A(I)-D*A(H): NEXT F
690 NEXT E: RETURN
```

While you are entering this program, don't forget to take full advantage of your pocket computer's capabilities. For instance, when entering a program, use abbreviations whenever possible. If you use abbreviations, line 10 is entered as:

```
10 "A":I."ORDER ? ";A
```

Line 230 becomes:

```
230 P.U."####";E;F;U."######.####";A(I)
```

Abbreviations not only make program entry faster, they save space in the input buffer. Note: P. is the abbreviation for PRINT, the abbreviation for PAUSE is PA.

You can also take advantage of your pocket computer's ability to duplicate lines. GOSUB XXX appears as a separate line eight times. Rather than entering GOS. XXX each time, you only have to enter it once. For the first occurrence, the entry is:

```
90 GOS.500
```

For line 120 (the next occurrence) use:

Use L.90 or the UP arrow to return to line 90. You will see:

```
90:GOSUB 500
```

Press the (◀) or (▶) arrow. The cursor will move over the G in GOSUB. Press the (◀) arrow twice. This will move the cursor over the 9 in 90. Since the new line number is 120, we need to insert a space for the 1. Press (SHFT), then (INS). This creates a space. Press (1) (2). The line number now reads 120. No other changes are needed, so press (ENTER). This procedure does not save you a large number of keystrokes on a line like 120 GOSUB 500, but if you were duplicating a line with 40 or 50 characters, this procedure can save you a considerable amount of time.

How does the Program work?

If you look at the program listing, you will notice that we have DEFined four keys. Press the (MODE) key until your Pocket Computer is in the DEF mode.

"A" — is the data entry routine.
"S" — is the inverse calculation routine
"F" — is used to fix incorrect data
"S" — is used to repeat or re-display the elements of the inverted matrix.

(Continued on Page 16)

Pocket (From Page 15)

"A"—Data Entry

Press (SHFT)(A) to begin data entry. The first question will be "ORDER ?". The order of the square matrix is the number of elements in a row or column. The manual which provided this program indicates that you can find the inverse of a 10 x 10 matrix. (If n memories are available, the number of possible elements in the matrix is the square root of (n – 9)).

Once you have entered the order of the matrix, the program will display the element number and then a question mark. When the question mark appears, enter the proper value. The information is entered by rows and columns within the row.

For example, to enter the information for this 2nd order matrix:

$$\begin{bmatrix} 6 & 7 \\ 5 & 6 \end{bmatrix}$$

Press (SHFT)(A). For the ORDER ? question enter (2).
The display will flash 1 1 at the ? enter (6)
1 2 (7)
2 1 (5)
2 2 (6)

When the > prompt returns, data entry is complete.

"F"—Data Correction

If you have made an error during data entry, use (SHFT)(F) to correct the item. After you press (SHFT)(F), the display will flash CORRECT, then prompt you to enter a value for I =. This value is the ROW the incorrect element appears in. J= requests the column of the incorrect element. When the ? appears, enter the correct value for element (I,J). The display again flashes CORRECT, and returns to I = for further corrections. When you have made all needed corrections, press (SHFT)(S) to begin the inverse calculation, or press (BREAK) to end the data correction program.

"S"—Inverse Calculation

After you have completed your data entry, and have made any needed corrections, pressing (SHFT)(S) will begin the calculation of the matrix inverse. This procedure can take a while. When the calculation is completed, the computer will BEEP 3 times and display element 1, 1 of the inverse matrix. For our example 2nd order matrix, what will be displayed is:

```
1     1     5.9999
```

Press (ENTER) for each of the remaining elements:

```
(ENTER)    1     2     -7.0000
(ENTER)    2     1     -4.9999
(ENTER)    2     2      6.0000
```

These values give us an inverse matrix of:

$$\begin{bmatrix} 6 & -7 \\ -5 & 6 \end{bmatrix}$$

Notice that we rounded the results for elements (1,1) and (2,1). When to round, and when to not round can be determined by inspection (reasonableness) and by multiplying the two matrices together. The result should be the identity matrix for the order of matrix being used.

If you should try to calculate the inverse of a singular or non-invertible matrix, the computer will display "NO SOLUTION."

"B"—Re-display results

Once the computer has calculated the elements of the inverse matrix, you can re-display the results, in order, by pressing (SHFT)(B).

Summary: I tried this program on a number of matrices of 2nd, 3rd and 4th order, and received reliable results in every case but one. The matrix:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

is invertable, but the computer indicated NO SOLUTION.

Perhaps one of our readers has a better program for inverting matrices?

Fancy Print

William F. Myers Fraser, Michigan

Here is a program I would like to submit for your newsletter.

This program for Model I/III prints an array of strings in the middle of the screen dynamically by the use of one line of video RAM. The POS(0) begins at 32 and ends at 48.

```
990 DEFSTR B, W, X
1000 X=BB(F)
     : TP=32
     : LN=464
     : TR=0
     : FOR P=1 TO LEN(X)
     : W= MID$(X,P,TP)
     : PRINT@LN, W;
     : TS=LEN(X)-P
     : IF TS>30 THEN NEXT P ELSE TR=TR+1
     : X1=BB(F+1)
     : W1=LEFT$(X1,TR)
     : PRINT@496-TR, W1;
     : NEXT P
     : F=F+1
     : X=""
     : X1=""
     : IF W1="END" THEN END ELSE 1000
```

Simple Word Processing

Bruce Webb Mountain View, California

I have a Model I with expansion interface, two drives, line printer IV and needed a way to use the system to type letters on letterhead. I did not wish to go to the extreme of upper/lower case mod and Scripsit.

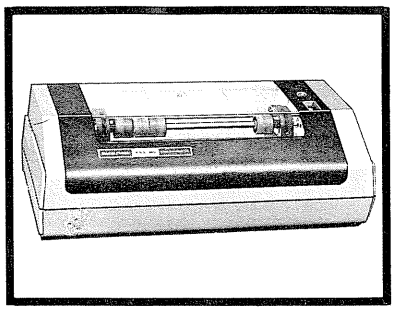
This program is my simple solution to the problem of using both letterhead and straightening out the upper/lower case situation.

This method is slow but works well enough for my purposes:

```
10 CLS
15 CLEAR 1000
30 LPRINT CHR$(27); CHR$(17)
35 A$=STRING$(21,CHR$(32))
40 LINE INPUT X$
50 GOSUB 10110
60 LPRINT A$; 0$
70 GOTO 40
10110 A=LEN(X$)
    :0$=""
    :IF X$="" RETURN
10120 FOR B=1 TO A
    :C=ASC(MID$(X$,B,1))
10130 IF C>64 AND C<91 THEN C=C+32
    :GOTO 10150
10140 IF C>95 THEN C=C-32
10150 0$=0$+CHR$(C)
10160 NEXT B
    :RETURN
```

More Computer Clubs

Microcomputer Users International
c/o Bernie Arbic, Secretary
118 East 6th Ave.
Sault Ste. Marie, Michigan 49783



Peripherals

Product Line Manager's News

This is clean house month. Behind every P.L.M. is a good secretary. Cindy says if I don't get rid of all these notes that have piled up on my desk they will wind up in next week's recycled newsprint.

RIBBON LIFE EXPECTATIONS

We are adding a host of new printer users every month so the following comments bear repeating.

Our printers can lay down from 30 to 160 characters per second depending on the model. Even Cindy can't type that fast! Your printer manual will quote a ribbon life of anywhere from 250,000 to 2 million characters. Don't be mislead by that seemingly big number. Line Printer V for instance, printing at 100% duty (constant, non stop 132 character lines) can deplete the ink in less than five hours. At that rate a 3100 sheet box of paper could use up to ten ribbons! Here is a chart outlining ribbon life of various printers expressed in real world terms of worst case lines and pages.

TRS-80 Ribbon Life Expectancies:

Printer	Cat. Num.	LPM	Ribbon	CPR	Pages	Hrs.
LP I	26-1150/52	—	26-1413	1 mil.	250	—
LP II	26-1154	31	26-1413	1 mil.	250	4.1
LP III	26-1156	48	26-1414	2 mil.	500	5.3
LP IV	26-1159	22	26-1413	1 mil.	250	5.7
LP V	26-1165	60	26-1414	2 mil.	500	4.2
LP VI	26-1166	33	26-1418	1 mil.	250	3.8
LP VII	26-1167	11	26-1424	1 mil.	250	11.5
DW II	26-1158	20	26-1419	250,000	67	1.7
Pocket*	26-3505	60	26-3507	85,000	—	1.5

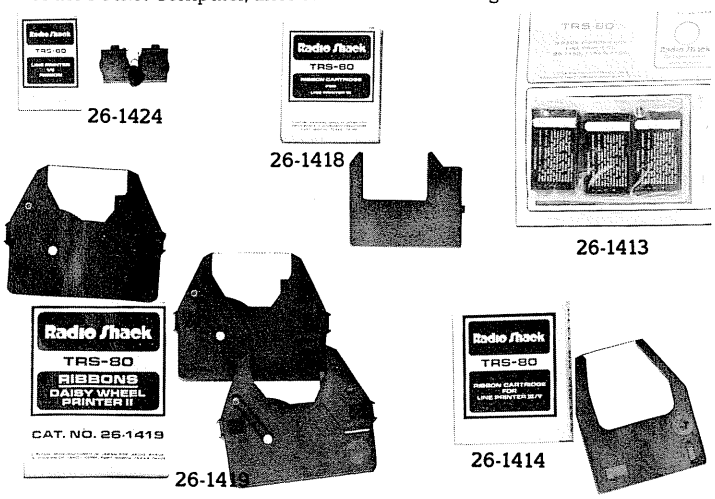
LPM is 80 or 132 character Lines Per Minute depending on the printer.

CPR is the number of Characters Per Ribbon.

Pages are 50 lines per page, 80 characters per line. Note that most applications will average fewer than 80 characters per line so this should be a conservative value.

Hours are the number of hours of printing you could get printing non-stop 132 characters per line, 66 lines per page. We use 66 lines per page to eliminate pauses for form feeds. There are virtually no applications which would subject a printer to this work load, and most of our printers are not designed to work at maximum output for extended periods. We give you this figure only for comparison purposes, and to help you conceptualize the expected life of a ribbon.

*For the Pocket Computer, lines are 16 characters long.



Another important related note: it is imperative that you maintain proper impression control position in dot matrix printers. All of the dot matrix printers have a feature which allows you to move the printhead farther away from the platen when using multi-part forms to prevent smudging or to allow insertion of paper. If you move the head back, make sure you replace it to the closest position when returning to single part paper. If the paper-printhead gap is too large you can damage the head. A damaged head can fray a ribbon . . . (see June page 24) and here we go again!

I hate to be in a chiding mood but I must address another misunderstanding. Some users have complained that their Tractor for the D.W. II mishandles paper. (I know . . . most of you are still waiting to see your tractor! Hang on, our increased factory production should be making its way through to your end of the pipeline real soon! PROMISE). Anyway, investigation has revealed that most mishaps occur when the user forgets to release the platen pressure lever when using the tractor. Also please note that the mechanism on the current tractor moves only in a forward motion.

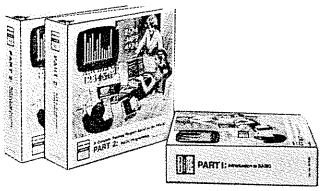
Enough bad news. Here's some good news. Deliveries of the Plotter/Printer should have resumed by now. There have been some changes made in the product during its 5 month absence. This new version has had a few design shortcomings corrected. The character mode now prints 80 columns instead of 75 and auto-wraparound is provided. In addition lower case characters are now converted to upper case and printed instead of being ignored! Older models can be upgraded by Radio Shack Service Centers for normal repairs and parts charges. In early May, the total parts and labor for the upgrade was estimated to be $70.

RANDOM NOTES FLOATING TO THE TOP OF MY DESK

If you have a color computer don't forget the 8-bit printer driver (700-2013—FREE) which is necessary for Color BASIC 1.0 CPUs. The graphic screenprint utility (26-3021 — $4.95) allows dumps of any resolution graphic screen to the L.P. VII. It also includes its own 8-bit driver. ----- The linefeed mechanism of LP VII is not designed to pull the heavy selfstick mailing labels like our stock number 26-1404 and 72-205 (2 up labels available from Computer Centers only). In order to accommodate those of you who want to print labels with LP VII we are introducing a pack of dry adhesive labels. They come two-up on a 9½ fan fold sheet. They can easily be used on the LP II/IV as well. (26-1456—$9.95) ----- HERE IS SOME EXCITING NEWS: Suggested retail prices have been reduced on all floppy media. Single 5¼" diskettes have been reduced to $4.95 (26-0305). The three-pack (26-0405) is now $13.95 and the Ten-pack (26-0406) now goes for $39.95. The Ten-pack of 8" diskettes (26-4906) has been reduced to $59.95. There is also a new generous volume discount structure for the Ten-packs. You read it here first, folks! ----- Direct Connect Modem (26-1172) is for single line phones only. We do not recommend using a multi line phone. ----- Cassette Comm Software (26-1139) is for 16K (and up) Level II or disk based Model I TRS-80s only. This program will not work in 4K of RAM, Level I machines, or any Model III.

COMING ATTRACTIONS

The RADIO SHACK annual catalog should be arriving in about a month. Is it full of new goodies! Next month I can start to tell you about the new peripheral products in the line . . . Stay tuned right here . . . Happy Printing!



Education

Educational Products News

Using Scripsit with a Network 2 Controller to Teach Word Processing

By Dennis Tanner
With Thanks to Sandra Williams and Bill Thames

Word processors are rapidly replacing typewriters as text-handling machines in offices throughout the world. Students in business schools and high school business classes need to learn word-processing skills if they are to survive in the modern office.

Scripsit™ is Radio Shack's word-processing program. It allows the user to enter and edit text on the video screen of the TRS-80, then to print the corrected copy on a printer. The typist can save the text in its edited form on a disk or tape (depending on the system) for later use.

Scripsit and Radio Shack's Network 2 Controller make a great team for teaching word processing. Up to 16 tape systems can be connected to a Model III disk system using the controller. This forms a versatile classroom setup that lets the teacher send the Scripsit program and sample text to the student stations. It also lets the students send their work to the teacher's disk system for evaluation, printing, and fast storage on a diskette.

The equipment and software required for this are as follows: a 32K or 48K Model III disk system, from one to sixteen 16K Model I or Model III tape systems, a Network 2 Controller, a cassette tape recorder, Model III Disk Scripsit, and Tape Scripsit. (Use Model I Tape Scripsit if the tape systems are Model I's and Model III Tape Scripsit if the tape systems are Model III's. The programs are different.) A line printer may also be used.

If you are using both Model I and Model III tape systems, you should make a BACKUP copy of the Disk SCRIPSIT diskette. Then do the following twelve steps twice: once with Model III Disk Scripsit and Model I Tape Scripsit, and once (on the second Disk Scripsit diskette) with Model III Tape Scripsit.

The first step is to load Tape Scripsit into the disk system so it can be saved on the diskette. You only need to do this once, since you can use the program from the disk after that. To load the program, follow this procedure:

1) Connect the tape recorder to the Model III disk system.
2) Place the Scripsit tape in the tape recorder. Rewind the tape and press the PLAY button on the recorder. Adjust the volume to between 5 and 7.
3) Turn on the Computer.
4) When the red light on the drive goes off, place the Scripsit diskette into the bottom drive of the Model III.
5) Press the orange RESET button.
6) Enter the Date and Time (if desired).
7) When `TRSDOS Ready` appears, type:
   `TAPE (S=T, D=D)`
   and press (ENTER).
8) When `Cass?` appears, type:
   L
9) When `Press ANY Key to begin cassette` appears, press any key.
10) The computer will begin loading the program from the tape. This takes about five minutes. When the tape has been loaded, the disk drive will come on and Tape Scripsit will be stored on the diskette under the filename SCRIPS/CMD.
11) When TRSDOS Ready appears, type:
    `BUILD GETSET` (ENTER)
12) When the computer

Displays:	You should type:
`Hit BREAK to exit`	
`Type in up to 63 characters`	`BASIC` (ENTER)
`Type in up to 63 characters`	(ENTER)
`Type in up to 63 characters`	(ENTER)
`Type in up to 63 characters`	`POKE 16913,0` (ENTER)
`Type in up to 63 characters`	(ENTER)
`Type in up to 63 characters`	`CMD"S"` (ENTER)
`Type in up to 63 characters`	`SCRIPSIT` (ENTER)
`Type in up to 63 characters`	`Press` (BREAK)

At this point, `TRSDOS Ready` appears again, and you are ready to proceed. The next step is to load Tape Scripsit from the disk system through the controller to the tape systems. To prepare for this, you should plug the controller's power supply into the jack marked "PWR" on the back of the controller. The cable from the cassette port of the disk system should be plugged into the jack marked "CPU." The cable from the cassette ports of each of the tape systems should be plugged into one of the jacks marked "REMOTE UNITS I/O." The power switch on the controller should be on, the selector set at "MPLX" and the baud rate set at 500.

Turn on each of the tape systems. At each tape system, follow these steps:

1) (This step applies to Model III tape systems only.)
   When `Cass?` appears, type:
   L
2) When `Memory Size?` appears, press (ENTER).
3) When `READY` appears, type:
   `SYSTEM`
   and press (ENTER).
4) When `*?` appears, type:
   `SCRIPS`
   and press (ENTER).

At the disk system, after you have completed the above steps for all the tape systems, follow these steps:

1) Be sure the diskette containing "SCRIPS/CMD" is in the bottom disk drive.
2) If `TRSDOS Ready` is not displayed on the screen, press the orange RESET button.

(Continued on Page 19)

Word Processing (From Page 18)

3) When `TRSDOS Ready` appears, type:

`TAPE (S=D, D=T)`

and press (ENTER).

4) When `Cass?` appears, type:

`L`

5) When `Device = Disk to Tape - File-spec?` appears, type:

`SCRIPS/CMD`

and press (ENTER).

6) When `Press ANY Key when Cassette ready` appears, press any key.

Two stars (asterisks) should appear in the upper right corner of the screen, and the right star should blink. When the stars stop blinking, *? will appear on the video for each tape system. At each system, press the slash key (/) and press (ENTER). At this point, each tape system is ready for word processing with Tape Scripsit, and the disk system returns to `TRSDOS Ready`.

As a teacher of a word-processing class, you may want to "send" each student at the tape machines a sample letter or some other text that you have prepared beforehand. To prepare a document beforehand, type in the text using Disk Scripsit and save it on the diskette according to the instructions in the Disk Scripsit package. When you have saved your text on the disk, the system can be turned off or used for any other purpose.

When you are ready to present the text to your students, use the following procedure:

1) Be sure each tape station has had Tape Scripsit loaded in according to the steps above.

2) When Scripsit's blinking cursor appears on the tape machine's video display, press the (BREAK) key, type:

`L`

and press (ENTER) at each system that is to receive the text. The red light on the controller corresponding to each of the receiving tape systems will come on.

The remaining steps should be performed at the disk system:

3) Be sure the diskette containing Disk Scripsit is in the bottom disk drive.

4) When `TRSDOS Ready` appears, type:

`DO GETSET`

and press (ENTER).

(In this example, the filename "SAMPLE" is used. Be sure to use your own filename when loading in your text.)

5) When Scripsit's blinking cursor appears, press the (BREAK) key, type:

`L SAMPLE`

and press (ENTER). This will load the text from disk.

6) When the blinking cursor returns, press the (BREAK) key, type:

`S,T`

and press (ENTER).

7) The text from the disk system will now be transferred to each properly set tape system.

One more thing you can do with the Network 2 Controller is to send text from the students' tape systems to the teacher's disk system to be evaluated, printed, and saved on the disk. Here's how to do this:

1) At the disk system, load in Scripsit as instructed in steps 3 and 4 immediately above.

2) Turn the "INPUT SELECT" knob on the controller to the number that corresponds with the tape system that will be sending the text.

3) When Scripsit's blinking cursor appears on the disk system, press the (BREAK) key, type:

`L,T`

and press (ENTER).

4) At the tape machine that will be sending the text, press (BREAK), type:

`S`

and press (ENTER).

5) When the blinking cursor reappears on the disk machine, you may save the text on the diskette. (The filename used in the following example is "STUDENT1"; you should use your own filenames.) Press the (BREAK) key, type:

`S STUDENT1`

and press (ENTER).

The text will be saved on the diskette under the filename you chose.

This system has enough flexibility so the teacher can individualize instruction by sending different projects to different students, or present the entire class with the same document. Student work is saved quickly on diskettes.

Business people might note that this very same system could be used in an office. What other system has sixteen word processors which can all be used for different jobs to be saved on a diskette, for about a thousand dollars per word processor?

QSP TO OFFER RADIO SHACK COMPUTERS AS PREMIUMS TO SCHOOLS

The Radio Shack division of Tandy Corporation has announced an agreement with QSP, Inc., a subsidiary of The Readers Digest Association, Inc., that will enable elementary schools to earn TRS-80 computers as premiums from magazine subscription sales.

QSP is taking their established fund-raising program a step further by offering the opportunity to bring computer literacy into the classroom — a new and growing concept in the educational field.

Charles A. Phillips, a senior vice president for Radio Shack, said that "QSP's innovative program should prove a boon to educators who find themselves budget constrained to bring the computer age to their classrooms and satisfy parents' demands for helping their children become literate with computers."

Radio Shack's long-term commitment to support educational uses of the TRS-80 microcomputer in schools is evidenced in the fact that a complete department has been set up for that purpose.

"We are supporting a massive development program that is producing high quality, tested and validated courseware," commented William Gattis, director of Radio Shack's Education Division. "We are supporting field test sites in numerous schools to test and refine our courseware products before placing them on the market."

Schools interested in the QSP program should contact William E. Drake, QSP, Inc., A Subsidiary of the Readers Digest Association, Inc., Box 2003, Ridgefield, CT 06877.

Computers in Education

Dania Stager-Snow Graduate School of Education
Rutgers the State University
New Brunswick, New Jersey 08903

Computers are pervasive in our culture. You find computers in one form or another in a diversity of situations. For example, the local mechanic uses a computer to diagnose mechanical problems, while down the street a clerk registers a sale on a computer terminal that not only notes the sale and the change, if cash is given, but takes inventory, or charges the customer for monthly billing from the central office. Thus a knowledge of computers is becoming increasingly necessary to function in society.

A student graduating from high school today is entering a work market that is increasingly computerized. Without the basic knowledge of computers that has become broadly known as computer literacy, the student is at a distinct disadvantage. The question is raised — How can the student become computer literate if teachers and those who train teachers are themselves not computer literate? True, there are rare cases where individual schools or teachers have adopted some sort of program, but this serves only to accentuate the existing gap between affluent and not-so-affluent school districts. The Graduate School of Education (GSE) at Rutgers University, recognizing the need for teachers and administrators to understand the role of computers in the society, is cooperating with practitioners in the field to bring the microcomputer into central focus in the instructional process. This effort entails expanded teacher training accompanied by massive professional development programs to bring the educational community up to functional levels in computer literacy.

Heightened public awareness of microcomputers has created a receptive climate among the faculty and students of GSE. Similarly, teachers in the nation's classrooms are becoming aware of the need to explore the capabilities of these machines. With this in mind, Dania Stager-Snow, a "micro-buff" and doctoral candidate in the GSE, approached Tandy Corporation with the request for a loan of two microcomputers for demonstration purposes. Tandy Corporation Senior Vice-President Charles A. Phillips agreed that the GSE's unique position, as the only public institution in New Jersey awarding the doctoral degree, made it a prime location for demonstration and dissemination of computer information. Accordingly, he approved the loan of two TRS-80 Level II Model I microcomputers to the GSE.

The two microcomputers were placed in a highly visible area in the GSE Learning Resource Center. The Learning Resource Center seemed like an ideal location since it is used extensively by both students and faculty. Under the direction of Ms. Stager-Snow and with active support from Dean Irene Athey, an intensive campaign for "computer literacy" and computer awareness was initiated. First, a series of workshops for faculty and students was conducted dealing with the operation and capabilities of the machines, and including some training in BASIC programming. Lectures and demonstrations were given on the educational uses of computers and their implications for the different disciplines. Several school districts subsequently requested that the workshops be repeated around the state to help acquaint practicing teachers with the ideas being presented. A policy of loaning the computers to faculty and students on weekends proved to be a very popular innovation which allowed individuals to increase their skills on the machines.

With the assistance from the Radio Shack Division of Tandy Corporation, Rutgers University's Graduate School of Education is enthusiastically integrating the microcomputer into its educational programs. The GSE's primary mission is the advancement of interdisciplinary research and inquiry, and microcomputers clearly serve a useful purpose in data storage and analysis. It is in the instructional uses of microcomputers, however, that the greatest interest has been shown by faculty and students. Thus the GSE community has endorsed the movement to microcomputers and is incorporating them into its course offerings on a broad scale.

To this end, a core curriculum dealing with computers, education, and the societal issues raised by computers is being created to extend the existing program. For example: Dr. George Pallrand with a group of doctoral and post-doctoral students is exploring the development of conditional reasoning in students using the microcomputer with programs they are developing in Logic; Dr. May Huang uses the microcomputer to demonstrate a program in nutrition education; Drs. Edward Fry and Martin Kling require students in certain courses to design programs for applications in reading; and doctoral candidate Stager-Snow, Drs. Hillson (from Elementary Education), Pallrand (from Science Education), Scrupski (from Sociology), and Simcoe (from the Bureau of Educational Research and Development) are examining the effects of the microcomputer on education in non-school settings, particularly in the home environment.

The above examples help illustrate the breadth and diversity of interests and applications that the faculty of Rutgers' GSE have developed using the popular TRS-80s. Because of this interest and support, the GSE is establishing a Microcomputer Instructional Center. The loan of the two TRS-80s from Tandy Corporation has been instrumental in moving the School forward in its plans to develop a center serving the various New Jersey and neighboring educational communities. The concepts, applications, and implications of microcomputers are now part of the intellectual climate at the GSE.

Disk Inventory (From Page 6)

```
400 PRINT "<6> DISPLAY BY DISK NUMBER"
410 PRINT "<7> PRINT TO PRINTER BY PROGRAM NAME"
420 PRINT "<8> PRINT TO PRINTER BY DISK NUMBER"
430 Z$=INKEY$
    :IF Z$="" GOTO 430
440 MN=VAL(Z$)
    :ON MN GOTO 300, 280, 190, 60, 240, 470, 570,
    570
450 IF MN<1 OR MN>8 GOTO 330
460 IF MN=8 LPRINT"PROGRAMS BY DISK NUMBER IN
    ALPHABETICAL ORDER"
470 KK=0
    :CLS
    :PRINT
    :INPUT"ENTER DISK NUMBER DESIRED OR PRESS
    ENTER FOR ALL  ";KK
480 K=1
    :CLS
    :FOR SS=0 TO DD
    :LL=VAL(LEFT$(C$(SS),2))
490 IF KK>00 AND KK<>LL THEN NEXT SS ELSE 500
500 IF SS=0 GOTO 520
510 IF (KK=0) AND (LL>PP) AND (MN=6) THEN PRINT
    :PRINT
    :INPUT"  PRESS ENTER FOR NEXT DISK ";KK;
    :CLS
520 IF (MN=8) AND (KK=0) AND (LL>PP) THEN LPRINT" "
    :LPRINT" "
530 PP=VAL(LEFT$(C$(SS),2))
    :PRINT C$(SS),;
    :IF MN=8 THEN LPRINT C$(SS),;
540 NEXT SS
550 IF MN=8 LPRINT " "
    :LPRINT" "
560 PRINT
    :INPUT"PRESS ENTER FOR MENU ";KK;
    :GOTO 330
570 IF PEEK(14312)<>60 THEN CLS
    :PRINT@398, CHR$(23)"PRINTER NOT READY "
    :FOR W=1 TO 1000
    :NEXT W
    :GOTO 330
580 CLS
    :LPRINT CHR$(27); CHR$(20);
    :LPRINT" "
    :IF MN=7 GOTO 230 ELSE 460
590 GOTO 330
```

Microcomputing for DUCKs?

During the summer of 1981 Duke University, Durham, North Carolina, in cooperation with Radio Shack, is offering a series of computer camps for youths. The four Duke University Computer Kamps (DUCK) are designed to develop a basic computer literacy through knowledge acquired about the various types and programming of computer hardware and software. The students live in supervised non-air conditioned dormitories, eat meals in the university dining halls and enjoy other opportunities associated with the Duke University campus.

The DUCK experience seeks to identify and develop the talented and motivated pioneers of the future. Many hours of careful planning and scheduling by professionals in the field led to the development of the DUCK outline.

Early in the first week three extensive classroom sessions will be used to provide instruction in the fundamentals of "BASIC" programming. These are among the first activities in twelve eight hour days of scheduled activities.

As part of the scheduled activities, kampers will be asked to conceptualize and develop two special projects; one personal project and one team project. Included in the DUCK experience are fourteen computer laboratory sessions so kampers will have a chance to gain experience on the various types of computers available. Among the computer facilities is a computing laboratory, provided by Radio Shack, which includes a configuration of 40 personal computers and an extensive software library. Other computer facilities include the Duke University Computation Center, and the Triangle Universities Computation Center. All of these computer facilities are available to kampers who wish to gain extra practice and experience over and above the sessions provided in the regular eight hour day.

In addition to their computer projects, kampers will participate in discussions with guest speakers having experience in computer applications. Some of the topics to be explored in these discussions will be computer applications in medicine, law enforcement, chemistry, sports, engineering, banking, the arts, government, space exploration, communications, prognostication, and personal/home use.

Two field trips are planned to the Research Triangle Park to visit companies or institutions using computing in engineering, research, and production. Kampers will experience through these visits how professionals, production employees, and administrators use various computers in different business and industrial applications. They will gain exposure to the Triangle Universities Computing Center (TUCC) which is one of the largest and best known computing facilities in the United States.

An "Open-House" event is scheduled to allow kampers to demonstrate the skills acquired and the projects developed. Kampers will display and explain the projects they have designed and developed.

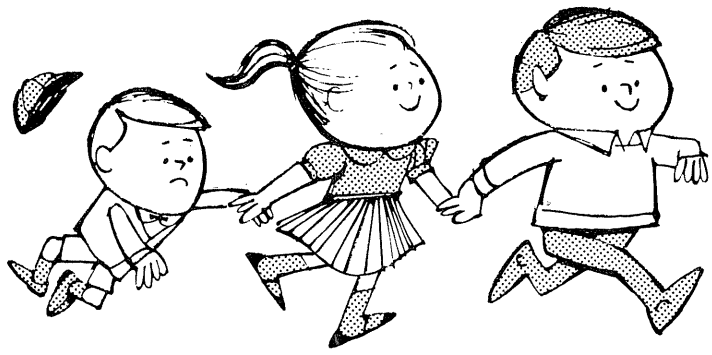
The main social event of the Kamp is a special banquet where kampers will receive awards and certificates in the presence of an audience of invited scholars, educators, business people, and community officials.

In addition to the scheduled daytime activities a full range of evening recreational activities has been carefully developed by the staff recreational advisors to assure kampers of ample time for rest, relaxation, and fun. For example, twelve recreational sessions have been scheduled throughout the kamp calendar in which kampers have full use of all campus recreational facilities, such as the gymnasium, swimming pool, tennis courts, and many other facilities. Special supervised events are scheduled so kampers may leave the campus and enjoy a sample of the community activities like a Durham Bulls baseball game, or local community theatre production, and possibly, performances of the American Dance Festival.

The instructional staff and consultants will be Ph.D. level computer scientists with interest in teaching youths regarding the use of computers.

For further information contact:

Duke University Computer Kamp
121 Allen Building
Duke University
Durham, North Carolina 27706
(919) 684-2621



Paging Routine

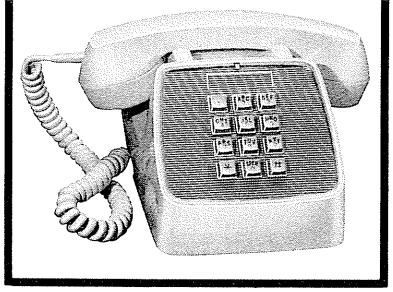
Charles Talbot Dearborn, Michigan

I was interested in Richard Halloran's program on paging for program lists (Dec. 1980)—a "want" that I've had for some time.

You may be interested in some enhancements that I worked out to better meet my needs:

1. A title line including date (time can be added using TIME$ function) and page number.
2. Eliminating the need to set 'N'—the number of lines by using an error trapping routine (statement 10) to end the program when the program list is complete.
3. Number of lines per page is set by the value of L in statement 60 (I use 57).
4. Width of printer line is set by N in statement 60 (I use 132).
5. Second lines of long program lines are indented to avoid confusion with the line numbers.

```
10 CLEAR 1500
    :ON ERROR GOTO 160
20 DEFSTR A-C
    :DEFINT I-N
30 INPUT "ENTER NAME OF PROGRAM TO BE LISTED (MUST BE SAVED
   ASCII)"; A
40 INPUT "ENTER TODAYS DATE"; A1
50 OPEN"I", 1, A
60 J=0
    : L=57
    : N=132
70 J=J+1
80 LPRINT CHR$(14); STRING$(28," "); A; STRING$(4," ");
    "DATE "; A1;   " PAGE"  J
90 LPRINT " "
    :POKE 16425,1
100 LINEINPUT #1, B
110 K=LEN(B)
120 IF K>N THEN B1=B
    :B=LEFT$(B,N)
130 LPRINT B
140 IF K>N THEN B="   "+ RIGHT$(B1,K-N)
    :GOTO 110
150 IF PEEK(16425)>L THEN LPRINT CHR$(12)
    :GOTO 70 ELSE 100
160 CLOSE
    :LPRINT CHR$(12)
    :END
```



Customer Service

Hints, Help and Tips

Since this is our first column in the newsletter, I wanted to take the time to tell you what we hope to accomplish. First, let me say that these articles are being written by the Customer Service Representatives. (We are the people who answer the toll-free WATS lines.) We would like any feedback you have to offer on our effectiveness. In general these articles will consist of a feature story, usually pertaining to one particular package or procedure, which we have gotten a significant number of inquiries about. The remaining part of the article will consist of answers to the questions we get asked most frequently. We hope that this will help you and us. If we can get you answers to the fairly common questions, for which there are simple answers, this will keep you from having to call us, wait for a representative and then be done in a matter of minutes. It will also help us, by freeing the WATS lines for individuals with more complex questions. Our purpose is to get you answers in the best and fastest way possible. If you have questions that are not answered in the newsletter, call. But if we can provide answers here, it may save you time and frustration. (We know it can be very difficult to get in on the WATS lines.)

How many times have you called Customer Services with a problem and their response is that you must go to a backup? You reply that you have no backup or that the only backup you have has the same problem. At that point all we can do is tell you to start over. We have to do this daily. We know that it is frustrating for you — it is also frustrating for us. We care, and we want your software to work for you, not you having to always work to support the software.

How many backups should you have? How often should you make a backup? This month we are going to answer these questions and explain the importance of having a working backup procedure.

The question has been asked many times: "What is considered to be an adequate number or frequency of backups?" It is our opinion that backups need to be made every day that a program is run. We suggest that you make backups more than once a day if the amount of information being added to the files is large. We cannot give you any firm guidelines, but you might ask yourself this question: "How long will it take to re-enter all of the information that has been added since the last backup?" If it will only take twenty minutes to re-enter the entire morning's information, you may not want to make another backup now. If the answer is five hours, and it is only one o'clock, you might consider another backup. Backups can often be made in a way that will not overly disrupt the working day. When the computer operator takes a coffee break, or goes down the hall to make a stack of photo-copies, have the computer doing a backup at the same time. If you leave to go to lunch, have the computer do a backup while you are away. Of course this assumes that you have at least two disk drives on your system so you do not have to be present while the backup is being made.

Backups which are made during the day, or on a daily basis, should have a three disk "family." (We are going to talk about one backup a day, made at the end of each day. If you make more than one backup a day, use this procedure, and each "generation" will simply be for a shorter time period than a day.) On the morning of day one (say 01/01/81) you create a disk with program and data files (working copy) for your newly acquired Accounts Receivable program. (We will call this set A.) At the end of the day (01/01/81) you make your first generation of backups. You should now have two sets of identical disks. The first set (set A — the father generation) is the working copy you created that morning. The second set (set B — the son generation) is the backup of set A that you made at the end of the day.

When you begin work on 01/02/81, you should use set B as your working disks, with set A put aside. At the end of the day (01/02/81) you will make the second generation of backups. Your working disks (set B) will be backed up onto blank diskettes to create a set C. You now have two identical sets of the most current information (sets B — the new father, and C — the new son) and one copy of information which is a day old (set A — which now becomes the grandfather generation).

On 01/03/81, you start the day by using disk set C. The purpose of using three sets of disks (A, B, and C) is to give you insurance. If you discover that set C is glitched (because you try to use C and the program crashes), you go to the original (set B). Hopefully, set B is alright, you make a new backup set C, and continue by using the new set C. (DON'T use set B as your working disk for 01/03! If set B is good, it is your insurance policy.) If you find that set B is also glitched, and you have followed our procedure so far, disk set A contains good information from 01/01/81 and all you have to redo is 01/02/81. A lot of work maybe, but not as much as if you had to reconstruct both 01/01 and 01/02.

At the end of 01/03/81, backup set C onto set A. Again, you have two sets with current information (C — the new father generation, and A — which becomes the new son generation) and one set that is one day out of date (set B — the new grandfather generation).

At the beginning of each day (or backup period if you backup more frequently) you use the son as the working disks. At the end of the day, the son becomes the father of a new son, and the old father becomes a grandfather. Grandfathers pass away and become new sons.

On 01/04 you would begin the day using set A. Set A is the son disk you created from father set C on 01/03. Set B is the grandfather set to A. At the end of the 01/04 father A creates son B, and C is the new grandfather. You would then follow this procedure for a three generation daily backup system.

The daily three generation backup system gives you good day to day protection, but what happens if a ten ton elephant breaks into your office and sits on all three sets?

On a weekly basis, backups need to be made every Friday. On the first Friday, a backup will need to be made of the work completed on that Friday. If the end of Friday daily father disk is B, then B would be used to make the regular daily C son copy, and A would still become the daily grandfather. You would also use B to make a weekly backup set D. The D backup set should be stored in a location which is physically separate from your normal operations. That way, if disaster strikes, once you get the elephant out of the office you will have all work through the previous Friday on backup set D and the only work to be reconstructed will be what has been done in the current week.

We recommend a two generation weekly backup. At the end of our first week we created backup set D. At the end of week two (the second Friday), a backup will need to be made on a clean set of diskettes. Leave set D undisturbed and create a new weekly backup set E. On every Friday thereafter, backups will need to be made rotating between the two weekly sets D and E.

(Continued on Page 23)

Customer Services (From Page 22)

For extra protection, set up a monthly backup procedure. The monthly procedure, just as the weekly, would be a once a month 2-generation procedure (sets F and G). You may wish to keep the monthly backup diskettes in a safe deposit box, or some other location where they will be protected from any possible mishap.

The chart at the end of this article should help you see what the whole procedure looks like. (What is shown is an example only. The actual disk "names" backed up will depend on your situation.) You will notice that the chart includes one additional back-up which we haven't mentioned. That is a final backup at the end of the year. In general this annual backup would be used more for archival purposes than for data reconstruction.

The point, of course, is to make frequent backups, and to keep "insurance" copies. If the monthly backup sets F and G are only removed from the safe-deposit box on the last Friday of each month (only one set, either F or G), then the safe-deposit box will always contain a disk set which is no more than one month old. The whole purpose of backups is to minimize the amount of data that can be lost, and to make reconstruction as easy as possible. We hope that you never have to call on backup sets D through G to actually reconstruct data. But, if the worst happens, those disks will make the reconstruction as easy as possible.

Frequently Asked Questions

Model I/III

Question — Will Haunted House load on Model III?

Answer — Yes, but when you first load it in, you will get a *checksum error.* Follow this procedure to load the first part of the tape — rewind tape — type in SYSTEM (ENTER) — type in, HAUNT (ENTER) — press play on recorder — when checksum error appears on screen — rewind tape — press play — press (ENTER) on computer.

Question — How do you get the maximum number of points in PYRAMID?

Answer — Hint: If you wander around too much, the batteries in your flashlight will burn out and you'll have to get batteries from the vending machine, which will cost you several coins.

Question — The federal tax being withheld from my employees' paycheck is not being calculated properly.

Answer — Make sure that when you are setting up the payroll system that you use the annual table for both married and single employees (i.e., table 7 in the circular E)

CUSTOMER SERVICES ADDRESS AND PHONE NUMBERS

8AM to 7PM Central Time

Computer Services
900 Two Tandy Center
Fort Worth, Texas 76102

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

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

Switchboard — 1-817-390-3583

Back Up Chart

	Daily Backup		Weekly Backup		Monthly Backup		Yearly Backup		Insurance copies		
	Disk to Disk		Disk to Disk		Disk to Disk		Disk to Disk		Daily	Weekly	Monthly
Day 1	A	B									
Day 2	B	C							A		
Day 3	C	A							B		
Day 4	A	B							C		
Day 5 / Week 1	B	C	B	D					A		
Week 2	C	A	C	E					B	D	
Week 3	A	B	A	D					C	E	
Week 4 / Month 1	B	C	B	E	B	F			A	D	
Month 2	C	A	C	D	C	G			B	E	F
Month 3	A	B	A	E	A	F			C	D	G
•											
•											
•											
Month 11	B	C	B	D	B	G			A	E	F
Month 12	C	A	C	E	C	F	C	H	B	D	G

Radio Shack®

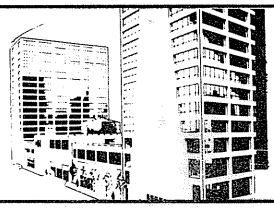
MICROCOMPUTER NEWS
P.O. BOX 2910
FORT WORTH, TX 76101

BULK RATE
U.S. POSTAGE
PAID
Radio Shack
A Div. of Tandy Corp.

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

IF UNDELIVERABLE DO NOT RETURN

Fort Worth Scene



Well, it has been a few months since I have had time to look around and write this column.

Where are the Level I users? I know that there are Level I users for both Model I and Model III who read the newsletter, but I never hear from you. Send me your favorite program that you have written for Level I. I would like to look at it, and publish it if I can. Don't let all those Level II users overshadow Level I! Level I users unite!

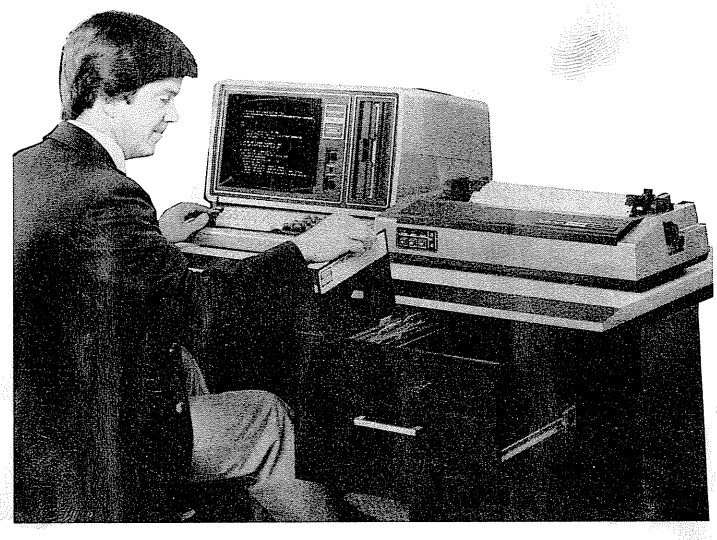
Here in Fort Worth, I just changed offices, lost my view of the Trinity River, but gained a little elbow room and an assistant! Yes, we now have a writer on the Newsletter staff to help put this thing together and get it out "on time."

Linda Miller is coming from Radio Shack's Customer Services department so she should be aware of the type of information you want and the types of questions you are asking. It will take Linda a little while to get used to our procedures, but we have her first article ever in this issue, and look forward to many more in the future. We are also looking forward to the new ideas that Linda is bringing with her. A column about Women and Computers? If you are interested, write to Linda at the Microcomputer News, P.O. Box 2910, Fort Worth, Texas 76101.

What interesting uses are YOU finding for your TRS-80? If you would like to share your use with other TRS-80 owners, drop us the information to P.O. Box 2910. We will use all of the information we can.

Both the Model I/III and the Model II Product Line Managers have recently asked me to remind users of Radio Shack Applications programs to NOT move or transfer these applications to new versions of TRSDOS when these new versions become available. Frequently these new versions will require minor changes to the applications programs before they will function correctly. If you feel you need to move up to a new TRSDOS version, check with your Computer Center or call Customer Services first. They may have specific changes that may be needed, or they may be able to provide you with information about a current or forthcoming update of your Applications program to the new TRSDOS.

That's all the notes I have this month. With Linda around to help get the Newsletter out each month (among her and my other responsibilities) Fort Worth Scene should start appearing regularly again. Until next Month,
Bruce W. Elliott, Editor
(Nameless no longer!)



TRS-80 Microcomputer News is published monthly by Radio Shack, a division of Tandy Corporation. A single one year subscription is available free to purchasers of new TRS-80 Microcomputer systems with addresses in the United States, Puerto Rico, and APO or FPO addresses. Subscriptions to other addresses are not available. The subscription rate for renewals and other interested persons is twelve dollars ($12.00) 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 76101.

Back issues of Microcomputer News 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.