Volume 4, Issue 2 **FEBRUARY, 1982**

Price $1.50

TRS-80® Microcomputer News

Information Published for TRS-80 users.

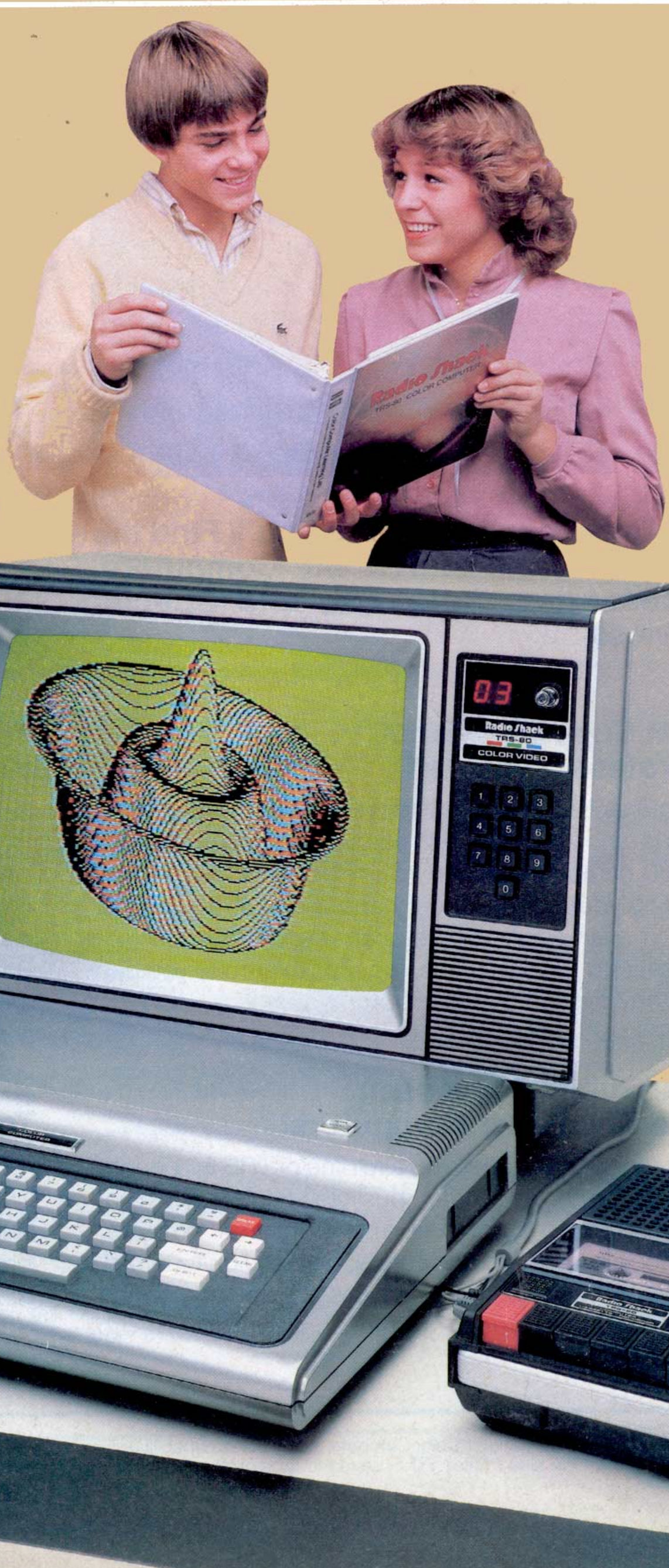
Color Computer Assembly Language

by William Barden

Random Files

Hard Disk Information

High Resolution Graphics



Radio Shack Computer Centers Mean Total Customer Support

The Only Location Where You'll Find the Complete Line of TRS-80® Computers, Accessories, Software, and Special Services

Total Support Means . . . A Professional Staff

Radio Shack Computer Centers are staffed by trained experts who can help you reach a decision on just the TRS-80 computer system you need. How? By understanding your needs, and by answering all your questions in plain English. They can explain our leasing, service, and training plans, too. For example, you can lease any system over $1500!

Total Support Means . . . More Than Just Computers!

Computer Centers offer you a full line of TRS-80 peripherals, software, and other business-related accessories, including business and personal copiers, telephone dialers and answerers, workstation furniture, printed forms, and more. All this, in addition to the world's best-supported microcomputer—Radio Shack's TRS-80!

Total Support Means . . . Quality Software At Low Prices

A Computer Center is the place to find the quality TRS-80 software you need. We offer more ready-to-run programs than any other microcomputer manufacturer. We have programs for business, industry, education, the professions, personal applications, even games! And, you can choose our software packages "right off the shelf" because they're in stock at Computer Centers nationwide.

Total Support Means . . . Complete Training Facilities

Our professional seminars will acquaint you and your personnel with our TRS-80 computer systems. You'll get "hands-on" experience, because you'll be using the TRS-80 while you learn. Develop BASIC programming skills, learn specific business applications and more. We even have free scholarships for educators.

Total Support Means . . . Keeping Your System Up and Running

Service is available through all of our Computer Centers. In fact, most Centers have complete service facilities on the premises. You'll receive the kind of prompt, courteous attention you expect. We want to get you back in service as soon as possible. That's why Radio Shack offers affordable "In-Shop" and, in most areas, convenient "On-Site" and "Limited On-Site" service agreements.

Total Support Means . . . Prompt, Personal Assistance

Our Customer Service Representatives are devoted to helping you with any after-the-sale questions or problems. You receive complete technical assistance and support, whether you've purchased one system or one hundred. Radio Shack Computer Centers simply give you the best support you'll find *anywhere.*

On the Cover. The versatile Color Computer has graphic capabilities which can challenge your imagination indefinitely. The image on the screen could be printed on the Line Printer VII or saved to tape on the CTR-80A so that it could be reloaded for future enjoyment.

TRS-80 Microcomputer News is published monthly by Radio Shack, a division of Tandy Corporation. A single six-month subscription is available free to purchasers of new TRS-80 Microcomputer systems with addresses in the United States, Canada and APO or FPO addresses. Subscriptions to other addresses are not available. The subscription rate for renewals and other interested persons with U.S., APO or FPO addresses is twelve dollars ($12.00) per year, check or money order. Single copies of the Microcomputer News may be purchased from Radio Shack Computer Centers or Computer Departments for $1.50 suggested retail each. The subscription rate for renewals and other interested persons with Canadian addresses is Eighteen dollars ($18.00) per year, check or money order in U.S. funds. All correspondence related to subscriptions should be sent to: Microcomputer News, P.O. Box 2910, Fort Worth, Texas 76113-2910.

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

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

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

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

Notes to Program Users:

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

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

Comments on our program listing style:

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

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

Trademark Credits

CompuServe(TM) CompuServe, Inc.
DIF(TM) Personal Software, Inc.
Dow Jones(TM) Dow Jones Co.
Personnel Manager(TM) Image Producers
Project Manager(TM) Image Producers
ReformaTTer(TM) Microtech Exports
Time Manager(TM) Image Producers
VisiCalc(TM) Personal Software, Inc.

Program Pak(TM) Tandy Corporation
SCRIPSIT(TM) Tandy Corporation
TRSDOS(TM) Tandy Corporation
TRS-80® Tandy Corporation

Volume 4, Issue 2 **FEBRUARY, 1982**

Price $1.50

TRS-80® Microcomputer News

Information Published for TRS-80 users.

Contents:

View From The Seventh Floor

by Jon Shirley,
Vice President,
Radio Shack Computer Merchandising

This column is being done so late that we almost had a blank page this month. All right, I can hear the snickers out there. The reason is that I have been doing a lot of traveling. I went to Las Vegas to see the COMDEX show which is a very good trade show aimed at what are called ISO's—Independent Sales Organizations. That means both retail computer dealers and the OEM's or systems houses, who are the people who buy computers to resell with their software. It was a costly experience since the casinos made off with their share of my money, but the show was great. You just have no idea how many people there are in this business these days. At least three new computers were introduced there plus bunches of peripherals and software. At times I really wonder if there are enough customers to buy all that stuff.

After a few hours at the show you get the impression that all the computers start to look alike and most seem to work alike. The difference comes from the software and the support the different manufacturers do or do not offer. A second generation of software is just around the corner which will be much easier to use and far more interactive from program to program. After all it does not really matter what is inside of the CPU as long as the software will do what you, the customer, needs. We ran our entire company on a mainframe with a core memory of 8K only 17 years ago and at that time that was an expanded memory.

After COMDEX I went to Florida to attend a word processing conference. This was a real insider show with representatives from all the big name word processing companies there. There were 11 people from IBM alone although they did not offer a speaker. I was very pleased to learn that in the low-end market word processors, which includes the IBM Display Writer, the Wangwriter and some other famous names, we are ranked in the top four with the Model II and SCRIPSIT. During the conference yet another personal computer was announced. This one will be from Olivetti and will be officially announced about the time you read this. I guess that will be the first one from Europe to make these shores except for Sinclair.

I would like to apologize to all of you who have been waiting for some of our software which is late. Especially those Model III owners who waited so long for VisiCalc, FORTRAN and the Editor Assembler. I have heard from a lot of you and all I can offer as a word of explanation is that we really do try not to announce a new product until we are fairly sure of its release date. That means it is in test and looking good. Of course this leaves us wide open to Murphy's law which says that if anything can go wrong, it will. The software law should be that even if nothing can go wrong it will anyway.

Sometimes the most solid looking package will develop a strange bug that defies all efforts to be found. Or even worse, the fix of that bug generates another bug and the test cycle is repeated again, and again, and—well you get the point. Our policy for the monthly flyers is that if an item is not in stock when the advertising material goes to the printer it will not be included in the flyer. This has been working correctly, it's the catalogs that are supposed to last for several months that get us in trouble. We like those catalogs to include new products that may not be available for up to four months after the book comes out, yet we are doing the book way in advance of its print date. We try to be conservative in those dates but we do miss. All I can promise is that we will try to do better and be more conservative. Remember we are excited about our new products and want you to be also. So please accept our apologies and try to understand that we know how you feel and we do not do it on purpose.

Color Computer owners might be interested in a monthly cassette by mail publication called Chromasette Magazine. It's for extended BASIC owners only and has about 6 programs per month. I have looked at a few and some are very neat and all can help you learn to program. Write to Chromasette at P.O. Box 1087, Santa Barbara, Ca. 93102 for details. In the last issue, the flyer that came with it had a review on the disk system and said we should have included 8K RAM on the disk controller board so no RAM at all was taken up by the DOS. It's a neat idea even though we take up less RAM than virtually any other disk system, but our engineers said it was not a workable idea for a variety of reasons plus we did not think it would have been worth the extra cost.

I just took home a copy of Art Gallery for the Color Computer and I think it's a worth a look by you C.C. owners, especially if you have young kids. Around my house it is almost as popular as Space Assault. It comes with a keyboard overlay to identify what the keys do and it makes creating exciting graphics really easy. If any of you have a use for a sign that is in color on a TV screen which can have letters of different shapes and sizes plus simple graphics, take a look at this package. I know of one very big company (not us) who uses Color Computers as signs to direct people around their facility. The "signs" can be easily changed each day as the need arises. It is also useful as a store window silent salesman when the store is closed.

I never get bored hearing of the many different applications that people find for our computers and if you have one that is unique drop us a line. I know of one company that built a Model I into the chicken breeding buildings they sold to control temperature and other things. Now that is a really unique use.

Finally I would like to thank the gentleman who wrote me a very nice letter and said he thought I reacted too much in this column to the "bozos" who wrote in with their problems. He said don't forget the great majority of owners who like their TRS-80, read the newsletter, and do not write letters. Considering the very large mass of owners we have and the very few

(Continued on page 6)

Random Access Files

Random (or direct) access files allow you to change data in any record or to directly access any record in a file.

RANDOM FILE STATEMENTS

1. OPEN

The first statement in creating a Random (Direct access on Model II) file is the OPEN statement. The following are example OPEN statements.

OPEN "R", 1, "NAMES/TEL" Model I, III

OPEN "D", 1, "NAMES/TEL" Model II

If the file NAMES/TEL already exists, it is OPENed for access. If the file NAMES/TEL does not already exist, it is created at this time. The "R" (or "D" for Model II) specifies that the file access will be Random which means that you will be able to both GET (read) records from or PUT (write) records to the file.

1 - This number specifies a buffer - in this case buffer #1. If the OPEN statement had been OPEN "R",5, NAMES/TEL then access to the file NAME/TEL would be through buffer 5.

Any access of a random file will be done through the buffer indicated in the OPEN statement. From 1 to 15 buffers can be open at one time. On the Model I and III the number of buffers open at a time is determined by the response to the HOW MANY FILES? prompt in Disk BASIC.

Responses to HOW MANY FILES? prompt - Model I/III

HOW MANY FILES?	⟨**ENTER**⟩	3 buffers reserved
HOW MANY FILES?	n	n buffers will be available for file use where n = 's any number from 0 to 15.

Model II Disk BASIC requires a TRSDOS entry like:

BASIC -F:5

which loads BASIC and sets aside 5 buffers for file access. If the TRSDOS entry is 'BASIC' then 0 buffers are reserved. Model II may have from 0 to 15 buffers available for file use.

"NAMES/TEL" - This TRSDOS file name follows standard TRSDOS format for file names.

2. FIELD

The FIELD statement provides access to the Random file buffer by determining how the data is organized in the buffer and subsequently stored on disk or read into your program.

FIELD 1, NME$ AS 30, NUM$ AS 8

In this file each record will consist of two fields. (Field is not to be confused with the statement FIELD.) The first field is NME$, the name which we will use for name and the second field NUM$ which we will use for the phone number.

Using the FIELD statement, the number of bytes the fields (NME$ and NUM$) are going to use in the record is determined. The total length of the two fields in this example is 38 bytes but when the record is stored on disk it will take up 256 bytes of storage space. With variable length records the disk space can be organized more efficiently but that is a subject for another article.

3. LSET, RSET

These commands are used to move the data items into the buffer fields previously defined by the FIELD statement. Either LSET (for Left Set) or RSET (for Right Set) can be used. If we LSET, the data for the name field (NME$) will be moved into the 30 byte field of the buffer beginning in the leftmost position of the field. With RSET the last character in the string will be the rightmost character in the field.

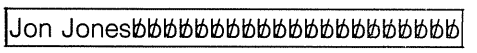
As a demonstration of LSET and RSET consider the following examples.

NME$ = "Jon Jones"

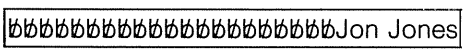
b's represent unused (blank) field space.

Each box represents the 30 byte field for NME$.

LSET - When NME$ is LSET into the buffer field it can be represented like this.



RSET - When NME$ is RSET into the buffer field for NME$ it can be represented like this.



LSET NME$ = "John Smith"	(If N$ = "John Smith" then LSET NME$ = N$ could be used.)
LSET NUM$ = "444-4444"	(If T$ = "444-4444" then LSET NUM$ = T$ could be used.)

Once the information has been LSET or RSET it can be PUT (written) on disk.

4. PUT

PUT n, n1 - The first number in the PUT statement is the buffer number and must match the buffer number which was used in the OPEN statement. "n1" is the record number to be written to disk. If "n1" is not indicated and the file has not been previously written to, the record will be written as record number one. If the file has previously been written to, the record currently in the buffer will be written to the next record position.

To write a record, a statement like the following would be used.

PUT 1,5

PUT 1,5 causes the record currently stored in buffer #1 to be written in the 5th record position of the disk file.

5. GET

GET n, n1 - The GET statement enables you to GET (read) a record which has previously been PUT (written) on disk. "n" is the same buffer number which was used in the OPEN statement. "n1" specifies the record to be read (with GET) from the disk file. If "n1" is omitted the current record will be read. If this is the first time that the file has been accessed in the program then the current record will be

number one. If it is the sixth time that the file has been accessed then the current record number is six.

6. CLOSE

CLOSE n, n1, . . . When file access is completed then the file must be CLOSEd. The CLOSE statement insures that any information still in the buffer is written to disk. In the statement CLOSE 1, 2 only the OPEN files which are using buffers 1 and 2 are CLOSEd. CLOSE with no buffer numbers causes all OPEN files to be closed.

EXAMPLES

The two brief programs that follow are included to illustrate the use of the Random file statements discussed above. Program I will ask for input of names and telephone numbers which will be written to disk. Program II will open NAME/NUM, GET the record number that you specify and print it on the screen.

PROGRAM I

```
100 CLS
110 OPEN"R",1,"NAME/NUM"
120 FIELD 1, 30 AS NME$, 10 AS NUM$
130 FOR X=1 TO 5
140 INPUT "NAME"; N$
150 INPUT "PHONE"; T$
160 CLS
170 LSET NME$=N$
180 LSET NUM$=T$
190 PUT 1,X
200 NEXT X
210 CLOSE
```

Lines 100 & 160 - Clear the screen

Line 110 - OPENs the file NAME/NUM for Random (Direct) access in buffer 1.

Line 120 - FIELDs buffer 1 with 30 bytes for NME$ and 10 bytes as NUM$.

Lines 130 & 200 - Set up a five step loop.

Line 140 - Allows entry of a name (actually any string) each time through the loop.

Line 150 - Allows entry of a telephone number (again any string) each time through the loop.

Line 170 - LSETs NME$ equal to N$ - the name entered in line 140.

Line 180 - LSETs NUM$ equal to T$ - the number entered in line 150.

Line 190 - Writes the record in buffer #1 to disk. The record numbers will be 1 through 5.

Line 210 - CLOSEs NAME/NUM which is the only open file in this program.

PROGRAM II

```
510 OPEN"R",1,"NAME/NUM"
520 FIELD 1, 30 AS NME$,10 AS NUM$
530 INPUT "WHICH RECORD (1-5)"; X
535 IF X<1 OR X>5 GOTO 570
540 GET 1, X
550 PRINT NME$; NUM$
560 GOTO 530
570 CLOSE
```

Line 510 - OPENs the file NAME/NUM for Random (Direct) access using buffer number 1.

Line 520 - FIELDs buffer 1 with 30 bytes for NME$ and 10 bytes for NUM$.

Line 530 - Prompts user to enter the number of the record to be read from disk.

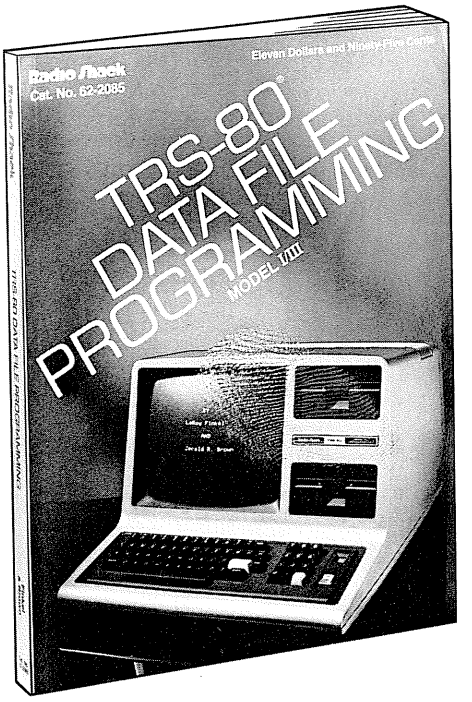
Line 535 - Checks to see if the number input is greater than 1 or less than 5. If either condition is true then execution goes to line 570 where all open files are closed.

Line 540 - The record number specified in line 530 is read from disk.

Line 550 - PRINTs the names (NME$) and phone numbers (NUM$) after they are read in from disk.

Line 560 - Returns to line 530.

Line 570 - CLOSEs the OPEN file NAME/NUM.



Seventh Floor (From page 4)

letters I do get he is right and I thank him for putting things in perspective. 1982 is going to be a good year for you TRS-80 owners, I know what is arriving this year and there will be something for everyone.

Until next month . . .

Last-Minute P.S.: It seems there is a rumor to the effect that non-FCC approved equipment may not be sold as of January 1, 1982. The December '81 issue of 80 Microcomputing states this as "may not be marketed." Sorry Wayne, ABSOLUTELY NOT CORRECT. The law states that computers and peripherals subject to class B requirements (Models I, III and Color Computer) may not be manufactured after January 1, 1981 (not 1982). Anything made before that date can be sold at any time. We obtained permission to make Model I expansion interfaces for one additional year and as of December 30, 1981 we are not making any more E.I.'s, although they should be available from our stores until around the end of June.

Color Computer Assembly Language Programming

by William Barden, Jr.

I was sipping my Diet mineral water and munching on lo-cal cheesecake for lunch (writers tend toward corpulence) when my office phone rang. "Is this William Barden, Jr. author of Radio Shack's 'TRS-80 Assembly Language Programming'?" "Yes," I admitted, eyeing the remainder of the cheesecake. "I have a collect call for you from Radio Shack, Fort Worth. Will you accept the charges?" "Yes, Operator."

Twenty minutes later, I hung up the phone and finished the cheesecake. Incredible! What a coup I had pulled! I had promised to do a series of articles on assembly language programming on the Color Computer. In return I would get complimentary copies of the newsletter. And people say that Radio Shack drives a hard bargain . . .

In fact, this **is** the first of a series of articles on that subject. I can't promise that I'll make a Color Computer assembly language programmer out of you with a short monthly column. At least not without your help. I **can** promise you enthusiastic material on assembly language programming on the CC. I'll start from the ground up, but you'll have to promise me that you'll supplement the columns with independent study. I know that you're going to enjoy some of the things that can be done with assembly language on the CC. Read on, and I'll give you an example.

IS MACHINE LANGUAGE CLICKS AND WHIRRS?

At this point you're probably pretty familiar with Color BASIC or Extended Color BASIC. You might have even noticed the DEFUSR and USR commands in BASIC. These are hooks to "machine language" code that may be called from BASIC. Machine language is the end product of assembly language.

Take the program below. It shows an assembly language listing for a program to output SOUND through the TV speaker. (Unlike the SOUND or PLAY command in BASIC, it can output a variety of sounds, and not just notes.) We'll use it as an example to explain assembly language and machine language and come back to it later on.

```
MACHINE-LANGUAGE CODE   EDIT LINE NUMBERS   ASSEMBLY-LANGUAGE CODE

                        ØØ1ØØ ************************************************
                        ØØ11Ø * SOUND PROGRAM. OUTPUTS SOUND BYTES FROM BUFFER  *
                        ØØ12Ø ************************************************
ØØØØ 1ØBE 3FØØ          ØØ13Ø SOUND   LDY    $3FØØ   GET # OF TIMES
ØØØ4 BE   3FØ3          ØØ14Ø SNDØ1Ø  LDX    $3FØ3     GET START ADDRESS
ØØØ7 A6   8Ø            ØØ15Ø SNDØ2Ø  LDA    ,X+         GET NEXT BYTE
ØØØ9 48                 ØØ16Ø         LSLA                ALIGN
ØØØA 48                 ØØ17Ø         LSLA
ØØØB B7   FF2Ø          ØØ18Ø         STA    $FF2Ø        OUTPUT
ØØØE 8D   ØC            ØØ19Ø         BSR    DELAY        DELAY
ØØ1Ø 1F   1Ø            ØØ2ØØ         TFR    X,D          CURRENT BYTE ADDR TO A,B
ØØ12 B3   3FØ5          ØØ21Ø         SUBD   $3FØ5        SUBTRACT END
ØØ15 26   FØ            ØØ22Ø         BNE    SNDØ2Ø       GO IF NOT END
ØØ17 31   3F            ØØ23Ø         LEAY   -1,Y       DECREMENT # OF TIMES
ØØ19 26   E9            ØØ24Ø         BNE    SNDØ1Ø     GO IF NOT Ø
ØØ1B 39                 ØØ25Ø         RTS              RETURN - DONE
ØØ1C B6   3FØ2          ØØ26Ø DELAY   LDA    $3FØ2    GET DELAY COUNT
ØØ1F 4A                 ØØ27Ø DELØ1Ø  DECA               COUNT-1
ØØ2Ø 26   FD            ØØ28Ø         BNE    DELØ1Ø     GO IF NOT Ø
ØØ22 39                 ØØ29Ø         RTS              RETURN
          ØØØØ          ØØ3ØØ         END
ØØØØØ TOTAL ERRORS
```

Figure 1. SOUND Assembly Language Program

The portion of the program on the right labeled "assembly language" is the actual text that is input. This text is processed by the Color Computer Editor/Assembler. The Editor/Assembly (after some coaxing), spews out the code on the left, labeled "machine language".

The machine language is really the binary data (ones and zeroes) that the 6809E microprocessor in the Color Computer understands. The binary data is in a shorthand form of binary called "hexadecimal" data. Hexadecimal is usually called "hex".

The machine language code in binary is put either onto a cassette tape "object" file so that it can be loaded into the CC by a CLOADM command, or put directly into RAM (random access memory) so that it can be tested (debugged).

THE HEART OF THE COLOR COMPUTER

The 6809E microprocessor is the heart of the CC. It has a built-in instruction set that is represented by machine codes such as the ones on the listing. These instructions are quite a bit different than BASIC instructions. They're very rudimentary, to put it nicely. You might compare BASIC to machine-code instructions the way you'd compare a James Joyce novel to a first-grade primer.

The 6809E can add two numbers, subtract two numbers, alter the sequence of instructions (a GOTO), load a number into a 6809E register, and do other types of processing at about this level. About the most powerful operation the 6809E can do is multiplication of two 8-bit numbers. This is not to say that the 6809E is not a powerful microprocessor! It is an excellent example of semiconductor design, lest I offend Motorola engineers. It just understands things on the "See the computer. Run, computer. Run, run, run!" level.

And that's the crux of the problem! We've got to string the rudimentary commands of the 6809E together to form some meaningful programs. It's certainly harder than BASIC.

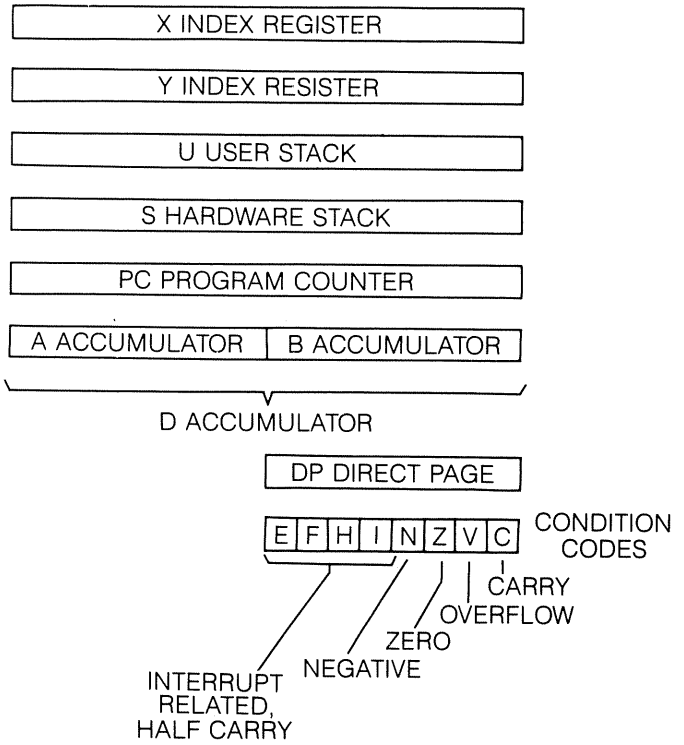


Figure 2. 6809E Programming Model

However, the resulting machine language program will be extremely fast and the rewards will be great.

To start our odyssey, then, we've got to become more familiar with the 6809E microprocessor. Figure 2 shows its component parts, at least the ones we'll have access to as assembly language programmers. In standard industry jargon, this is called the "programming model".

CPU REGISTERS

As you can see from the figure, there are a number of **registers** in the cpu. (I'll use the term cpu, or central processing unit, as an equivalent term for the 6809E.) A register is really just another memory location in the cpu. If registers are just other memory locations, why not put them in RAM? Like a lot of things in computers, the explanation for this is historical. Originally, registers in the cpu operated much faster than memory, so it was convenient to have high-speed cpu registers to store intermediate results. Now, the speed difference is much less, but the 6809E and other microprocessors still retain a set of registers for handling data.

Registers in the 6809E are either 8 or 16 bits long. Every computer user now knows that a bit is a contraction of "binary digit" and that it represents on/off or 1/0. Eight bits can hold 00000000 through 11111111, which is equivalent to 0 through 255 decimal, or $00 through $FF hex. (The "$" stands for "hexadecimal"). Sixteen-bit registers can hold 0 through 65,535 decimal or $0000 through $FFFF hex.

The main registers that hold intermediate results are the A and B registers. The A register (historical basis again) originally stood for "Accumulator." In early computers, the accumulator held intermediate results of adds, subtracts, and other operations. In the 6809E we've got two 8-bit accumulators named A and B, and they serve the same purpose. Joined together, they represent one 16-bit register, called the "D" register. The A and B registers can be handled either as two 8-bit registers or one 16-bit register.

The PC, or Program Counter, is a 16-bit register that points to the next instruction byte. Machine language instructions in the 6809E are one, two, three, or four bytes long. The PC contains the address of the next instruction to be executed at all times. If we have a machine language program at $3F00 (decimal 16128) in RAM, for example, the PC would start off at $3F00 and increment through $3F01, $3F02, . . . as each instruction was executed. Of course, in machine language we've got "branches and branch to subroutines" (the equivalents of BASIC's GOTO, IF . . . THEN GOTO . . . , and GOSUB), so at times the PC gets a new address other than the next in sequence jammed into it to cause the branch.

There are two Stack Pointer registers, U and S. A "Stack" is a reserved area of RAM, usually about 100 bytes or so. Every time a "Branch to Subroutine" occurs (BASIC GOSUB), the address of the next instruction after the Branch to Subroutine (in the PC) is saved in the Stack. Each address from the PC occupies 2 bytes, and the S Stack Pointer is adjusted to point to the next two bytes of the Stack.

The Stack area, "builds down" in RAM. Every time a Branch to Subroutine is done, two more address bytes are saved on the Stack, and the S Stack Pointer is adjusted by subtracting 2. Of course, every subroutine has a Return From Subroutine (equivalent to a BASIC RETURN) which retrieves the address from the Stack, reloads the PC, and allows the program to continue from the instruction directly after the Branch to Subroutine.

In addition to saving subroutine addresses, the Stack can also be used to temporarily hold data from cpu registers or other memory locations. The Stack also holds addresses on **interrupts**, a subject which we'll discuss in a future column.

TWO FOR THE PRICE OF ONE

Actually, I lied. (It probably won't be the last time.) There can be more than one Stack. The S Stack Pointer points to the **Hardware** Stack, which is normally reserved for subroutine addresses, interrupt addresses, and occasional temporary data. Most microprocessors have only one Stack Pointer; the 6809E, however, has two. The U, or User Stack Pointer, can be used by the programmer for whatever purpose he wishes. (Well, almost any purpose). Again, we'll discuss this operation in future columns.

The X and Y index registers are two 16-bit registers which can be used in certain **indexing address modes** in the 6809E. Indexed addressing allows you to set a pointer to the start of a table of data, and then very easily access the data from that point of reference. It sounds like another subject for a future column, and it is.

The Direct Page is also used in 6809E addressing; it can be set to point to a "page" of memory. A page is 256 bytes long. Page 0 starts at RAM location 0, page 1 at location 256, and so forth.

The Condition Codes hold the results of operations performed in the cpu such as adds, subtracts, compares, and many others. Its equivalent functions in BASIC are the IF . . . THEN statements. Instead of saying "IF A=0 THEN GOTO 1000" we say "If the Z(ero) Condition Code is set, then Branch," or "Branch on Equal to Zero." There are condition codes for a zero condition, negative result, overflow (number too large for the A, B, or other registers), carry (like a decimal carry), and system functions such as interrupts.

And that, in a nutshell, is how the 6809E cpu looks, at least to a programmer. All machine language operations, from adding two numbers to implementing a high-speed game with color graphics, are done by manipulating data in the registers. Operations such as simple adds, subtracts, compares, transfers to and from memory, and shifts are used as building blocks to construct all kinds of complex operations.

A SOUND PROGRAM (HOPEFULLY)

Let's see how a typical assembly language program works in the Color Computer, and how it is connected to BASIC. We'll use the example seen earlier, the SOUND program.

GENERATING SOUND IN THE COLOR COMPUTER

One way the Color Computer generates sounds is by doing a "six-bit" digital-to-analog (D to A) conversion. A six-bit binary value from 000000 to 111111 is transformed into a voltage from about 0 volts to about 5 volts. The greater the six-bit value, the greater the voltage. This technique is used to "synthesize" sine waves for cassette recording, to create musical tones, to create sound effects, and even to create voices.

We can use assembly language to output successive 6-bit values to the digital-to-analog converter in the CC and create our own sounds. Assembly language is great for this because it's very fast. BASIC just does not have the speed to output the 6-bit values at a rapid enough rate.

The address $FF20 (65312) in the CC is the input to the analog-to-digital converter. If we store a 6-bit value in that address, the digital-to-analog converter will convert it to a voltage level. If we store values at a rapid rate, we'll get a varying voltage level at the D to A output. Some of the waveforms we can get are shown in Figure 3.

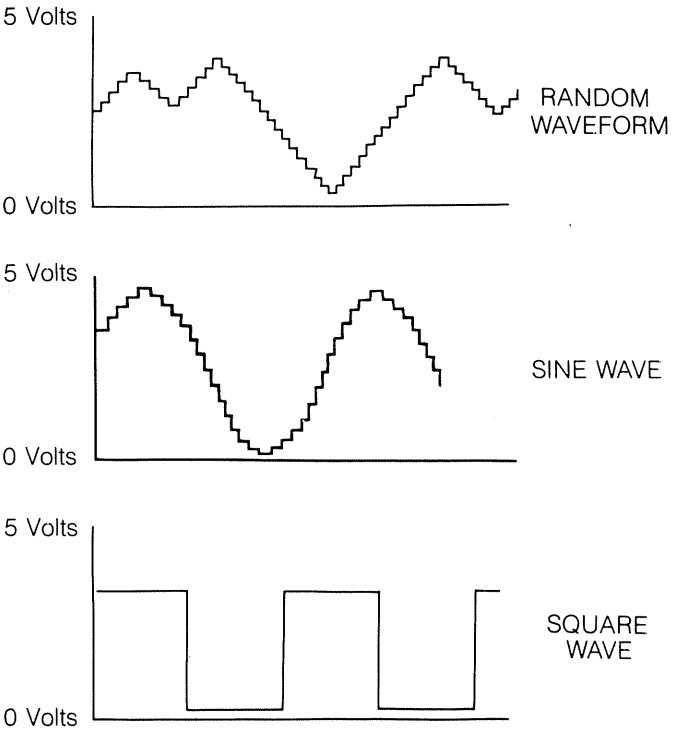


Figure 3. Digital to Analog Output Waveforms

The output of the D to A can be routed to the TV speaker, and we can easily hear the results. The SOUND program lets us experiment with different data rates, different data, and even repeat the same data for thousands of times. An infinite number of sound effects (flying saucers, laser blasts, crashes, buzzes) can be generated just from RAM and ROM data in memory alone, or you can try your own data.

VARIABLES IN THE SOUND PROGRAM

The SOUND program uses four variables from the BASIC program to generate a sound. The four variables must be POKEd into locations $3F00 through $3F06 as shown in Figure 4.

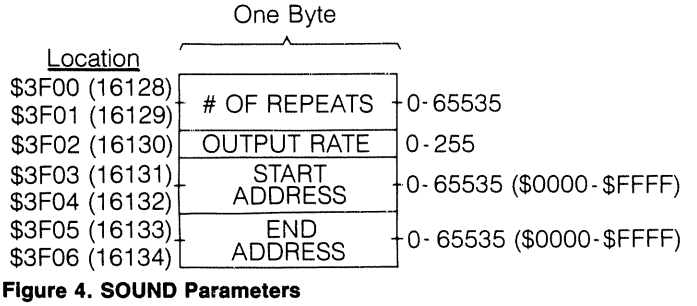


Figure 4. SOUND Parameters

The first variable is the number of times the data is to be repeated. If this is a 1, the data will be output in one burst to the D to A. If this variable is a 10, ten consecutive bursts will be done, but because of the speed of machine language, it will

sound like one long continuous burst. Up to 65,535 bursts can be done, and this parameter therefore requires two bytes of storage at $3F00, $3F01.

The second variable is the output rate. A value of 1 is the fastest and a value of 255 is the slowest. This variable is stored by a POKE to location $3F02.

The third and fourth variables are the starting and ending addresses of the data. These two variables are stored by POKEs to locations $3F03,$3F04 and locations $3F05,$3F06. You can put any start and end locations into the two variables - ROM locations, RAM locations, or your own data area that has been initialized by POKEs.

THE BASIC DRIVER

Figure 5 shows the BASIC driver, the program that calls the SOUND assembly language program. Statements 110 through 130 contain the machine language code for SOUND in decimal values of DATA statements. If you look at the first location of SOUND, you'll see a hex "10BE." Hex 10 is decimal 16 and hex BE is decimal 190, and these are the first two values in the DATA statement in line 110. There are 35 DATA values, each corresponding to a machine language byte in SOUND.

```
1ØØ REM BASIC DRIVER FOR AL SOUND
11Ø DATA 16, 19Ø, 63, Ø, 19Ø, 63, 3, 166, 128,
     72, 72, 183, 255, 32
12Ø DATA 141, 12, 31, 16, 179, 63, 5, 38, 24Ø,
     49, 63, 38, 233, 57
13Ø DATA 182, 63, 2, 74, 38, 253, 57
14Ø FOR I=&H3FØ7 TO &H3F29
15Ø READ A
16Ø POKE I, A
17Ø NEXT I
18Ø DEFUSRØ=&H3FØ7
19Ø POKE &HFFØ1, (PEEK(&HFFØ1) AND &HF7)
2ØØ POKE &HFFØ3, (PEEK(&HFFØ3) AND &HF7)
21Ø POKE &HFF23, (PEEK(&HFF23) OR 8)
22Ø INPUT "# TIMES"; N
23Ø INPUT "DELAY"; D
24Ø INPUT "START"; S
25Ø INPUT "END"; E
26Ø POKE &H3FØØ, INT(N/256)
     : POKE &H3FØ1, N-INT(N/256)*256
27Ø POKE &H3FØ2, D
28Ø POKE &H3FØ3, INT(S/256)
     : POKE &H3FØ4, S-INT(S/256)*256
29Ø POKE &H3FØ5, INT(E/256)
     : POKE &H3FØ6, E-INT(E/256)*256
3ØØ A=USRØ(Ø)
31Ø GOTO 22Ø
```

Figure 5. BASIC Driver for SOUND

The first action taken in the BASIC driver is to move the 35 DATA values to locations $3F07 (16135) through $3F29 (16169). The "&H" here is Extended Color BASIC's prefix that replaces a "$" used in assembly language. This is done in BASIC lines 140 through 170. After this action, the machine language code for SOUND is now in the $3F07 area. (Use 16135 and 16169 for &H3F07 and &H3F29 for Color BASIC.)

An alternative to this action would have been to assemble SOUND with output to cassette tape. A CLOADM would then have loaded the machine code of SOUND into this area. We took the route of including the machine language code as part of a BASIC program, however, to make it one package, a common trick.

The statement at line 180 defines the machine language code as starting at location &H3F07 by a DEFUSR0. (Use "180 POKE 275,63: POKE 276,7" for Color BASIC.)

The next three statements at 190 through 210 route the output of the D to A to the TV audio and turn on the sound. Locations &HFF01, &HFF03, and &HFF23 are special "hardware" addresses that are dedicated to system functions. (Use 65281, 65283, and 65315 for Color BASIC.)

Next, the four parameters are input using INPUT statements. The # of repeats, output rate, start address, and end address are represented by N, D, S, and E, respectively. The four parameters are stored in locations &H3F00 through &H3F06 by POKEs in statements 260 through 290. The POKEs store the repeat count, start, and end as two 8-bit values; INT(X/256) gets the first (high-order) byte, while X-INT(X/256)*256 finds the second (low-order) byte. (Use 16128, 16129, 16130, 16131, 16132, 16133, and 16134 for Color BASIC.)

The USR0 call at line 300 transfers control to the SOUND routine at &H3F07 BASIC finds the proper address from the DEFUSR0 address. The A variable and "(0)" are meaningless; they are "dummy" variables, but necessary to prevent a BASIC syntax error. (Use "USR(0)" for Color BASIC.)

SOUND ASSEMBLY LANGUAGE CODE

Look at Figure 1 again and we'll show you what happens in SOUND.

The LDY $3F00 gets the number of times parameter from location $3F00 and $3F01. The 16-bit value is put into the Y index register. This value will be decreased by 1 each time through the loop from line 140 through 240.

The LDX $3F03 gets the start address parameter from locations $3F03 and $3F04. The 16-bit value is put into the X index register. This "pointer" address value will be increased by 1 for every output to the D to A over the range of data from start to end address. The ",X+" operand automatically increments X by 1 each time "LDA ,X+" is executed.

The loop from 150 through 220 outputs data from start to end address at the output rate specified.

The LDA ,X+ gets a byte of data, and aligns it to the form XXXXXX00, where XXXXXX is the 6 bits of data. The data value is then output to the D-to-A at hardware address $FF20.

The DELAY subroutine is called right after the output. DELAY gets the output rate count from location $3F02, and delays accordingly. It returns to the TFR X,D instruction.

The TFR X,D tests the current address of the data by subtracting the end address. If they are not equal, the loop starting at SND020 is executed again. If the start and end addresses are equal, the RTS returns back to the BASIC driver at BASIC line 310.

HOW TO USE SOUND

Perform a CLEAR 200,16127 before entering the BASIC program of Figure 5. This protects the area from $3F00 on and prevents BASIC functions from destroying the program or parameters. Then enter the BASIC program and CSAVE it on cassette.

Run the BASIC program alone. The SOUND program is in the form of DATA statements and will be moved to the $3F07 area. Try various values for N, D, S, and E. A good area to produce a pure tone is located at 43100 through 43135; this is the "sine wave table" used to produce a cassette sine wave. Other areas will produce sound effects of different types.

WHAT TO DO IF YOU'RE CONFUSED

We've covered an awful lot of ground in this column - everything from the layout of the 6809E through Color Computer hardware addresses through BASIC interfacing. Don't feel bad if you don't understand everything presented. The purpose of this introductory article was to show you a typical assembly language program that works, that you can have fun with, and that you can use as a sample for your own study.

The next time we meet in this column I expect you to have looked over the DEFUSR and USR commands in BASIC, investigated how the parameters were stored, and looked up how most of the instructions in SOUND work in a good book on 6809E programming. (By the way, **my** Radio Shack book on Color Computer Assembly Language Programming will be out shortly.)

Next time we'll be looking at another example of an assembly language programming, which will show more of the power of the Color Computer. Until then, keep assembling!

Editor's Note: William Barden has written several books for Radio Shack. TRS-80 Assembly Language Programming is the book he wrote on Z-80 assembly language for the TRS-80 Model I (also applicable to the Model III).

Fort Worth Scene

I just saw the first printed copies of the December Microcomputer News. It looks good, but we will continue to make some modifications over the next few months to enhance readability.

WE BLEW IT!

In the September, 1981 issue we published a Cassette Merge Program for Model I. It does not work as advertised. It works great in the Model I system I use at home, but does not seem to work in other systems. So . . . if you have been having problems getting the program to work in a Model I, we apologize. If you have been trying to get the program to work in a Model III, and if your tapes were produced on a Model III, replace the 66 at the end of line 50 with a 67. Leonard Scott sent that to us.

Several people have given (or sent) me notes for this issue, so here are some brief bits of information:

APPLICATION SOFTWARE SOURCEBOOK

February 28th, 1982 is the cut-off date for Volume IV of the Application Software Sourcebook. If you had a listing in either volume I or II, you need to renew the listing by Feb. 28 if you want the listing to appear in Volume IV.

MODEL II SCRIPSIT USERS

If you have been using the 1.0 version, and you upgrade to 2.0, the following 2.0 procedure will let SCRIPSIT copy your 1.0 files to a 2.0 disk automatically (if you have at least two drives):

1. Make BACKUPs of all your 1.0 diskettes.
2. Place a 2.0 disk in disk drive zero.
3. Place one of the BACKUP 1.0 diskettes into drive one.
4. Open the first document on drive one, and create User key 1 as follows:

USER KEYSTROKES	DISCUSSION
⟨CTRL⟩Z 1 C	Begin capturing user key 1
⟨ESC⟩ Q	Exit the current document
2 C ⟨ENTER⟩	
⟨ESC⟩	Copy the first document to drive zero
2 ⟨F2⟩ Y (password) ⟨ENTER⟩	Delete the FIRST document on drive one
2 O ⟨ESC⟩ ⟨ESC⟩	Open the NEW first document on drive one
⟨ESC⟩ 1	Restart user key 1
⟨CRTL⟩ Z	Close the user key capture

5. Press ⟨ESC⟩ 1.

This user key will copy all documents from the 1.0 diskette in drive one to the 2.0 diskette in drive zero. The key also

(Continued on page 18)

Confessions of a Computing Family

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

Fifteen years ago, Al Smith became acquainted—albeit on a limited basis—with the world of computers and computer language. Call it planning, foresight, or merely happenstance, but today Al Smith feels right at home on his TRS-80 computer, whether programming data relating to his insurance business, or accessing the CompuServe Information Service for the latest New York Times book reviews.

Fifteen years from now, Al Smith might well look back on this time as the pioneering state of home computer use. Just as another generation recalls the advent of television—endless wrestling matches, Uncle Miltie, everything in living black and white—the computer users of today may look back on the late 1970s and early 1980s as a time of innovation, experimentation and technological discovery.

Although Smith says, "I'm really impressed with the database on CompuServe," he qualifies that statement with the prediction that, "I don't think what we have now is anything compared to what we'll see in 20 years, or even in five years."

Al Smith, an insurance agent in Columbus, Ohio, purchased his personal computer approximately two years ago. At the same time, he began subscribing to the CompuServe Information Service. During the week, he puts the computer to work in his insurance agency. Evenings and weekends, it becomes the center of attraction in the Smiths' suburban split-level home.

The family—Al's wife Ann, 10-year-old Jimmy, 8-year-old Brent and 3-year-old DeAnn—are suitably impressed with the system's capabilities, according to Al. "The boys' school system is now developing computer uses in the schools," he points out. "Now, the boys are getting an early education about computers at home. They're not afraid to sit down and play with it and learn from it." He reports that a favorite CompuServe feature is games. "Adventure," with its caves and dragons and assorted beasties, is played most often by Smith's sons and their friends.

Al Smith admits to "getting on there and playing games once in a while," but he points out that a real benefit of the CompuServe Information Service is "being able to trade information back and forth with other users." He also enjoys reading the various newspapers available on CompuServe as well as The Associated Press wire service.

He estimates that the service is used about 15 hours each month in the Smith household. Its popularity grows in proportion to the outside temperature decline. "In the wintertime, this is an especially good activity," Smith notes.

Smith is enthused about the future of home computer use. "Even now, I'm always pleasantly surprised to run through the What's New column on CompuServe and find out the new features that have been added," he says. "And I'm really looking forward to the day when we'll be able to get libraries, encyclopedias — you name it."

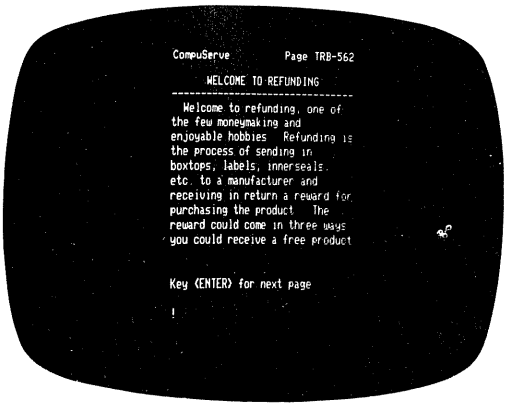
On CompuServe, Al, that day is closer than you think.

"REFUNDLE BUNDLE" NOW AVAILABLE ON COMPUSERVE

Refundle Bundle, a clearinghouse for consumer information about making and saving money through manufacturers refund offers, is now an information provider on the CompuServe Information Service.

For the veteran refunder and novice alike, Refundle Bundle offers valuable hints and tips on such topics as what to save from packages and how to save it, definitions of terms involved in refunding and descriptions of the various kinds of refund offers, such as the "form required" versus "no form required." An interactive section is also available through which readers can receive answers to specific questions.

CompuServe subscribers can access this information for the standard fee of $5 per hour weekday evenings, all day Saturday, Sunday and holidays. Weekday daytime access is also available. To access the service, a subscriber needs a personal computer or terminal, a telephone and a modem. The Refundle Bundle is accessed through main menu item 7, Home Information, or Go TRB-1.



HOW MUCH ARE YOU WORTH?

Determining what your assets and liabilities are is no easy task to undertake on your own. To be sure, certain procedures are rather elementary—adding up the worth of your house, money you have in savings accounts, for in-

stance. In fact, you may think you have a fairly good estimate of your assets and liabilities.

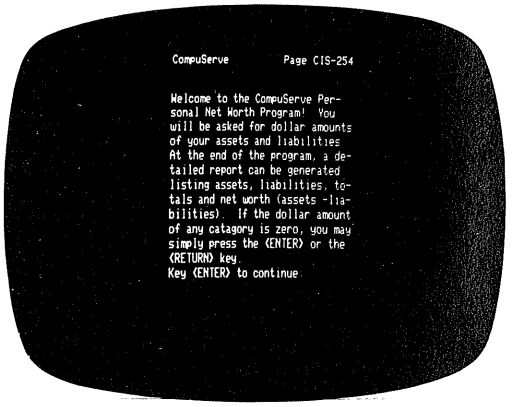
But wait a minute. Didn't you forget to include the cash in your individual retirement account? And how about the diamond pendant you inherited from Aunt Bessie? Did you enter that $750 loan you owe to your brother-in-law under liabilities? Perhaps not. However, such information must be included for any true accounting of your net worth. Hmmmm . . . maybe this procedure isn't so simple, at that.

The CompuServe Information Service has the answer, with a Personal Assets/Liabilities Program. By asking you the proper questions and recording your monetary answers, the program offers a precise system for determining your total assets and liabilities.

The program, of course, operates accurately only with your help. You, not the computer, must have the financial facts at hand. You must know the cash value of your life insurance policy, for instance, and the approximate worth of that diamond pendant. The program acts as a kind of "prompter," by listing categories you may not have considered. The program then totals your net worth by adding up your responses in various categories and subtracting the amount of your liabilities.

Of course, the personal financial information that you enter on the CompuServe Information Service is totally confidential.

To use the assets/liabilities program, access main menu item 9, MicroNET Personal Computing, and type R WORTH after the OK prompt.



RAYLUX REPORTS: FOR THE INVESTER WHO EXPLORES ALL OPTIONS

"Explore all options." You've heard such advice before, and seen the practice in action. A physician, for instance, examines all possible choices before proceeding with a course of medical care. A journalist checks reliable sources before going with a news story. A gourmet chef experiments with various ingredients until he finds the right combination for his next out-of-this-world concoction. So, too, should the savvy investor explore all the options before making financial decisions. Raylux Reports, a financial information service on the CompuServe Information Service, is one such option.

The Raylux philosophy is reflected in a paragraph that appears near the beginning of Raylux Reports on CompuServe: "Raylux is in the business of predicting the future, a vocation prone to error but essential if we're to navigate the waters of the future with any confidence. There are many seers—some more reliable than others—but unfortunately there is no way to know in advance who'll be right and who wrong." The Raylux Report stresses the message that it is vital for investors to educate themselves about facts and opinions at hand, so that they can make the most informed financial decisions possible.

To that end, Raylux provides information in five categories:

Business Outlook is a straightforward accounting of current economic trends. A recent sample topic, "Misleading Monetary and Employment Indicators," discusses the impact of those two variables and incorrect conclusions drawn therefrom.

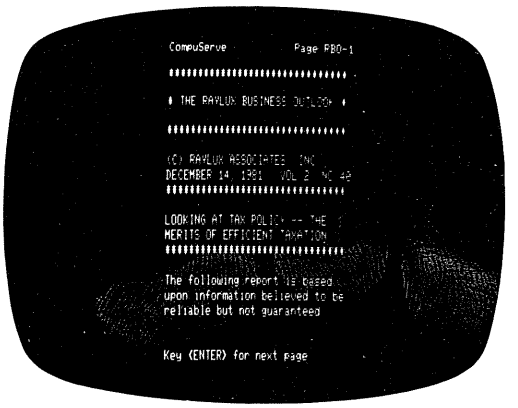
Financial Commentary is a flexible category, perhaps high lighting one particular investment area. A recent report on gemstones, for example, consists of an interview with a noted authority in the gemstone industry, in a question/answer format.

Stock Market Outlook is a combination historic examination and preview look at the stock market, and its "bullish" or "bearish" state.

Securities Glossary is a listing of well-known and lesser-known securities terms ("Acceptance," "Average Debt," "Maintenance," "Book Value," for example, is explored by a simple definition followed by a more thorough Raylux explanation: "Debt can take many forms, from unsecured debentures, secured first mortgages, bank notes, commercial paper," and so on. Raylux tells us.

Industrial Outlook Model is a computer program for making corporate projections on a 32K microcomputer system. The model, Raylux says, "simply represents information for you to consider when making personal financial decisions." Such a device "is important for consistency checks or error analyses of the projections routinely rendered by high-powered securities analysts and investment advisers," Raylux points out. The model allows investors to make five-year forecasts for specific companies, given other assumptions about profitability, dividend payout policy and other factors. Investors, with the Raylux model, can predict what percentage of growth would likely result in the deterioration or improvement in the company's debt-to-equity ratio.

Investors all over the country have enthusiastically adopted Raylux Reports as a valuable investment decision-making guide. Access Raylux through main menu item 2, Finance, or Go CIS-34.



OTHER NEW SERVICES ON COMPUSERVE

Three additions have been made to CompuServe's stable of electronic newspapers.

The St. Louis Post-Dispatch, the Atlanta Journal and Constitution and the (Framingham, Mass.) Middlesex News are now offering electronic editions of their news.

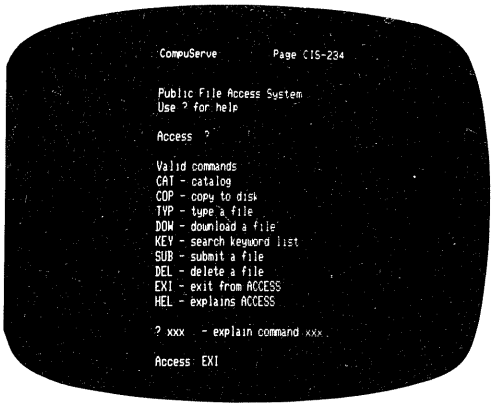
The Post-Dispatch offers a wide selection of classified ads from the St. Louis area, in addition to local and national news items updated hourly.

The Atlanta newspapers offer information on local airline flights and local weather as well as their full menu of news.

You can find the newest newspapers, in addition to other newspapers such as The Washington Post, The New York Times and the San Francisco Examiner, under main menu item 1, Newspapers.

For those of you who would like to share computer programs you have written with other microcomputerists, CompuServe now offers an area of our service called Access. The Access area is designed to be a place where CompuServe Information Service customers can share programs and knowledge. The Access area contains a wide selection of user-contributed programs, games, utilities, patches and some interesting text files. Go CIS-7 to reach Access, or type R ACCESS at the OK prompt in MicroNET Personal Computing, main menu item 9.

Questions and comments about the CompuServe Information Service can be sent to Richard A. Baker, editorial director, CompuServe Information Service, 5000 Arlington Centre Blvd., P.O. Box 20212, Columbus, Ohio 43220 or through Feedback, main menu item 5, CompuServe User Information.



Notes Off The CompuServe Wire

70715,272 - 01-Oct-81 - 23:06

From Gary E. Snyder
37 Sumner Park
Rochester NY 14616

The following two liner will turn a Model II with Daisy Wheel into a conventional typewriter:

```
1Ø CLS
     : PRINT CHR$(14); 'TURN ON DUAL FUNCTION
2Ø A$=INPUT$(1)
     : IF A$=CHR$(1) THEN PRINT CHR$(15)
     : END ELSE PRINT A$;
     : GOTO 2Ø 'IF F1 THEN TURN DUAL OFF AND END
```

Pressing the F1 key will end the simulation. Have fun with this one . . . Gary

71465,213 - 30-Sep-81 - 19:55

After spending hours trying to send many variables to my TRS-80 Model II assembly programs, I discovered that the random disk file buffers are always in the same locations in memory, regardless of the BASIC program.

Variable file buffer 1 starts at &H6C83 and continues for 255 bytes. In BASIC I load this memory area with the data I want to send to the assembly program using the LSET and RSET commands. I can then access the buffer in assembler with simple mnemonic code instructions. Any data I want to return to BASIC is put back into the buffer and BASIC can access the data by using the variables defined in the field statement.

I thought this might help assembly programmers who needed to pass more than one value to a USR call.

L.W. Host 71465,213

Computer Clubs

Things have been a little hectic around the Microcomputer News in the last few months, and we have not printed any information on computer clubs. Here are the names and contact information for clubs that we have heard about in the last few months. The clubs are listed alphabetically by state.

FORT SMITH COMPUTER CLUB
Dale Perrymore, Pres.
5820 South Y Street
Fort Smith, AR 72903
(501) 452-5362

TRS-80 USERS GROUP OF NAPA
4432 Springwood Drive
Napa, CA 94558

FAIRFIELD COUNTY TRS-80 USERS GROUP
c/o Alan Abrahamson
10 Richlee Road
Norwalk, CT 06851
(203) 866-2670

ASSOCIATION OF PERSONAL COMPUTER USERS
P.O. Box 19427
Washington D.C. 20036
(301) 229-2346

TRS-80 USERS GROUP OF ATLANTA, LTD.
1637 Columbia Drive
Decatur, GA 30034

COMPUTER TIME
ROBERT STUART COMPUTER CLUB
Robert Stuart Junior High School
644 Caswell Avenue, West
Twin Falls, ID 83301

NORTHEAST COMPUTER CLUB
P.O. Box 50252
Indianapolis, IN 46250

LOCAL ORGANIZATION OF COMPUTER OPERATORS
Jamesson's / Radio Shack
U.S. Route 1
Waldoboro, ME 04572
(207) 832-4218

MICROCOMPUTER USERS INTERNATIONAL
c/o Jack Decker—newsletter editor
1804 West 18th Street Lot #155
Sault Sainte Marie, MI 49783

LLIST
91 Valley Street
Rochester, NY 14612

The BASICS
17 North High Street
Fredericktown, OH 43019
(614) 494-9002

PORTLAND AREA USER GROUP
P.O. Box 1656
Beaverton, OR 97075

JACKSON AMATEUR COMPUTER SOCIETY
C.B.C.C.
2355 Camp Baker Road
Medford, OR 97501
(503) 535-6883

CAPATUG
Capital Area TRS-80 Users Group
c/o Computerland
4644 Carlisle Pike
Mechanicsburg, PA 17055

WINNIPEG MICRO - 80 USERS GROUP
Mr. Don Wood, President
(204) 452-5978 (evenings)
or
Mr. Don Rigg, Newsletter Editor
(204) 253-9230 (evenings)
188 Dromore Avenue
Winnipeg, Manitoba CANADA R3M 0J3

ASSOCIATION DE MICRO INFOMATIQUE DU SAGUENAY
a/s LABRI bureau C-405 Pav. princ.
University du Quebec a Chicoutimi
930 est, Jacques Cartier
Chicoutimi, Quebec CANADA G7H 2B1

LP VI Underline/Boldface

Don Wood
188 Dromore Avenue
Winnipeg, Manitoba CANADA R3M 0J3

Enclosed are a couple of routines I use to obtain true underlines and bold print on my Line Printer VI.

UNDERLINE

```
1Ø INPUT A$
     : ' INPUT MESSAGE
2Ø LPRINT A$;
     : ' PRINT MESSAGE
3Ø LPRINT CHR$( 27 ) ; CHR$ ( 28 )
     : ' SET TO 12 LINES / INCH AND PROVIDE LINE
     FEED
4Ø LPRINT STRING$ ( LEN ( A$ ) , 241 )
     : ' UNDERLINE MESSAGE
5Ø LPRINT CHR$ ( 27 ) ; CHR$ ( 54 );
     : RESET TO 6 LINES / INCH
```

BOLDFACE

```
1Ø FOR X = 1 TO 2
2Ø LPRINT "FOR BOLDFACE" ;
     : ' PRINT MESSAGE
3Ø LPRINT CHR$ (27) ; CHR$ (14) ;
     : ' SET TO CONDENSED PRINT
4Ø LPRINT CHR$ (27) ; CHR$ (15) ;
     : ' CANCEL CONDENSED PRINT (RETURNS CARRIAGE
     TO BEGINNING OF LINE)
5Ø NEXT X
     : ' LOOP TO OVERSTIKE
6Ø LPRINT
     : ' PROVIDE LINE FEED
```

Microfile Hints

William G.S. Smith
17 Keeler Close
Ridgefield, CT 06877

I have solved a practical problem I was having with MICROFILES: I have files of information on people. I often want to print their names first-name-first for mailing lists and other purposes. But, if I want to alphabetize the list, I must enter people's names in the stilted last-name-first format. One way around this problem is to have one field for the last name and another field for the first name, middle name, etc.

This arrangement allows me to sort on the LAST NAME field, yet still print the last name at the end of the name where it belongs. However, this is not an ideal solution. I must leave enough space in the FIRST NAME field for long first names, middle initials, titles, etc. This can lead to some pretty strange looking mailing labels, with huge gaps between first name and last name because of the unused spaces.

I was confronted with this problem and tried all sorts of complicated solutions. I wanted normal looking labels, but I also wanted to sort alphabetically. Then I realized the simple solution; that I could have my cake and eat it to by having TWO name fields.

One field is strictly for printing; I type in the full name just the way I want it to appear: Then, "Dr. & Mrs. Aloysius T. de B. Bindlestiff, III" and "Mr. Joe Smith" both look normal on the label. The second name field is for the last name only and is used for sorting. I also use it when all I need to print is the last name. I call this field simply "LN" for "last name." It is true that I have to type the last name twice, but the results are worth it.

The manual does not make it clear that the page break function will work only if you also use a heading. If you want a printout that leaves space between the pages you must, therefore, include a heading. However, you need not have unwanted printing at the head of each page. Simply make the heading consist of a few blanks. That way both you and the program will be happy.

Questions We Ask You

Our column often includes some of your most frequently asked or most interesting questions. In today's column we are going to repeat some of the questions we most frequently ask customers and share with you why these questions are really important.

Question 1: Have you got a BACKUP?

Reasons for BACKUPs: BACKUPs are a resource in case of burglary, fire, employee theft or accident. Remember that BACKUPs should be done in pairs (program and data) to help maintain equal wear on your disks. Take a BACKUP copy of your disks home with you at night. This can help you recover your information and can help the insurance company in any claim which might arise.

Accidents that might occur could be caused by hardware or input error. Misalignment of drives, dirt in your drives, mishandled disks, power surges, or excessive vibrations can create a "glitch" on your disk. A disk left on a printer can be wiped out just as easily as if it were bulk erased. Sometimes an employee is so upset by a problem they have created that they increase the problem themselves.

If a crashed index (particularly in Accounts Receivable and Accounts Payable) occurs (usually caused by exiting a program improperly) it is best to go to BACKUPs. Remember to exit the program as prompted, for example, "PRESS @ TO EXIT" means just that. If you get back to the main menu and then pull the disks out or hit RESET, the index-file will not be written. If you have noticed that the computer accesses the disk when you exit, this is when the index-file is being written to disk.

BACKUPs made frequently and kept updated, save hours of work and often save having to do a program over completely.

Question 2: What version of the program do you have?

Reason for asking about version number: Many of our programs are updated to meet different needs. In some situations there are new laws or tax changes that require additional information be kept. This may or may not be part of your problem or question about a program, but it does tell us where we are starting from. In many programs the version number is in the very beginning of the listing, and some programs incorporate a command for asking for version number (VisiCalc /V). It would be a good idea to make a printer copy of the whole program (if it is written in BASIC) by LLISTing it.

Question 3: What version of TRSDOS are you using with this program?

The reason for asking about TRSDOS (Tandy Radio Shack Disk Operating System): different models of TRS-80s use different TRSDOS versions for different programs. For Model I business applications programs that have been converted for use on a Model III, the TRSDOS should be 1.2. On the current Model III programs (that did not require converting), the TRSDOS should be 1.3. Model I programs are on 2.3 TRSDOS.

TRSDOS numbers, although similar to program version numbers, are different in meaning and the numerals do not reflect the computer model the TRSDOS is associated with or the newness of the program.

You may also think that we don't want to help you when you tell us that you're using xDOS. This is not really the case. We have not tried the program on anything other than the TRSDOS on which it was released. We can not support all the operating systems which have been released by other companies. While we are flattered that so many people are supplying both software and hardware for our computers, we are unable to support anything other than what we sell. We hope you understand that we'd like to help you, but just can't.

Question 4: Have you made any changes in the program or are you using it differently then it was intended?

Also, tell us if you have applied a patch to your program, even if it is one that we have sent to you. It might change the program slightly. You should also let us know if any other changes have been made to your program. Tell us also about any variations or changes in use or programming. We might not be able to help you if any large changes have been made, but at least we will be able to tell you right away if a correction could be made with those changes that are already in the program. Often we try to solve a problem and, after many hours of input by many members of our staff, we find out that a large change or deletion has been made in the program.

Speaking of staff, the folks here who answer your questions have spent a great deal of time learning about the software packages and often have specialized areas of expertise. Our main goal is to help you as quickly and as efficiently as possible. As a result, we will not only go to the best possible source in our department for an answer to an unusual question, but we will also spend time replicating your problem if necessary. Sometimes we will ask you to send us BACKUPs of your disks and copies of your paper work to help speed up the process or clarify the issue. We appreciate your help and support in solving your problems.

Now for some final areas where you can help us. If you have heard from a friend that there is a patch for a program, don't assume that the patch is necessarily needed in your copy of the program. Call us and check first. Patches are not always "preventive medicine".

The worst problems always seem to happen over a weekend or in the evening when we are not available to answer your questions. Please take notes on every thing that is happening and that happened before the problem occurred, note all error messages and their numbers. Write down in detail what the program did, what the hardware did and especially what you did.

Check Writer 80

We now have an opportunity to tell you about one of Radio Shack's newest programs, Check Writer 80™, our an-

swer to "pegboard accounting."

Check Writer 80 is one of our most versatile programs, with the capacity for storing information on up to nine bank accounts. The program also has the ability to display that account information automatically, by just entering the account number. Check Writer 80 can be an asset for the small business person and/or the home record keeper.

Check Writer 80 (which is also a check register), maintains a file of seventy-five payees and up to thirty expense codes, and allows the user to determine the "To-Date Balance" in any of the expense areas.

Payee number codes can be alpha or numeric or a combination of both. The expense code most frequently associated with a payee is displayed on the check entry screen. For instance, for paying checks to the xxxx oil company you might have created a code for transportation expense. When you enter the oil company's account number the display will show not only the name and other payee data but also the expense code for transportation expense. You can override the code number, if necessary. You can also write or record checks for non-regular payees stored in the system by using 00 for the payee account number. Normally the payee data will only show if the account number is valid.

In the bank reconciliation section, the bank name will automatically display after the account number is entered. There is an entry location for a service charge and for the ending balance listed on your bank's statement. The reconciliation section allows five fields to be displayed. A "CL" in the first field denotes each entry which has been previously cleared, check numbers or a blank for deposits will be in the second field. The third field will exhibit the payee's name or the word deposit. The check or deposit date fall into the fourth field and the check or deposit amount display in the last area. Checks can be written for amounts as large as nine hundred ninety-nine thousand, nine hundred ninety-nine dollars and ninety-nine cents.

When writing checks, the next check number is autogenerated after the bank account number is entered. You have the option to print checks with either tractor or friction feed printers. A sample check for a proper alignment test is included in your program manual. Your Computer Center or Radio Shack Store with an expanded computer department will be able to help you in ordering checks.

This program generates several end of period reports. After entering your bank(s)' statements, reports can be run. The general check register bank reconciliation lists deposits, checks and ending balances. It lists the name(s) of the bank(s), the account(s), the last check listed, the closing date and the ending balance.

The report options on the payees are by name sequence (you only have to enter the first ten characters) or by ID number sequence. Other options include an ongoing expense list as well as a year-end expense report.

We mentioned that Check Writer 80 can be used for your small business and we are sure that you can think of many ways it can be an aid for your business as well. One of the areas where it can be used is for keeping petty cash accounts separated. Check Writer 80 is a computerized bookkeeper, giving you truly simplified accounting.

MOST FREQUENTLY ASKED QUESTIONS

Question 1: How do I tell if the special character set is toggled in, or if the space compression codes are in use on my Model III?

Answer: Use the following program to be sure that the special characters are in use:

```
1Ø PRINT @ Ø, CHR$(192)
2Ø IF PEEK(1536Ø)=32 THEN PRINT CHR$(21)
3Ø CLS
```

This will print a space if the space compression codes are active, and a spade if the special character set is active. It then tests for a space character, and toggles the special character set on if a space is found.

Question 2: ROUTE was removed from the 1.3 Model III operating system. Why?

Answer: The ROUTE command was removed because of problems encountered with the ROM routines involved. On occasion, routing will cause loss of data, loss of program, etc. Some of these errors may even appear to be serious hardware problems. There were other problems related to routes which were not logical, but were possible, such as keyboard to printer or printer to keyboard. This would cause the computer to be looking for input from the printer or to try to print with the keyboard.

If you are trying to use a serial printer, the only permanent "fix" for the missing ROUTE is to write a serial driver program to reside in high memory, similar to the way LPC works with parallel printers. A temporary fix is to dump your printer output into a disk file, then use Model III Scripsit to read the file and output the information using Scripsit's serial printer capabilities.

Question 3: I am using several Model I programs on my Model III which ask me to "SHIFT D" to delete or "SHIFT I" to insert, etc. I can never make these functions work. What is wrong with the program, and what can I do?

Answer: There is nothing wrong with the program. You just need to remember that on a Model I, the shifted character is a lower case character (this is without the lower-case mode and driver, of course.) In order to output the lower case character on your Model III, you will need to "SHIFT Ø" (to go into lower-case mode), then press the key indicated. Be sure to "SHIFT Ø" again to return to all caps. Failure to do this will usually cause the program to "lock up" because most programs are written to reject the lower-case characters anywhere else.

Question 4: Is there some way to get into low baud in Disk BASIC on the Model III without having to make the patch given on Page 12 of the TRSDOS Manual?

Answer: Yes, all you need to do is type in (at the BASIC Ready prompt, or as a statement in your program) "POKE 16913,Ø." To return to high baud, type "POKE 16913,1." You might want to remember that all Model III cassette data files (created using PRINT#-1) are 500 baud files. Reading and writing tape files does not change the baud rate you have set for reading and writing tape program files.

Question 5: How do you print through the serial port with SCRIPSIT on the Model III?

Answer: If you are using Disk SCRIPSIT, do a "SETCOM" at TRSDOS, and set the protocol to match your printer. Then, when you want to print, type **〈BREAK〉** P,S **〈ENTER〉**.

If you have tape SCRIPSIT, you must set your protocol using the POKES at 16888, 16889 and 16890 (See Chapter 8 of your owner's manual). You must then initial-

ize the RS-232-C by executing the following program:

```
1Ø POKE 16526,9
2Ø POKE 16527,Ø
3Ø X=USR(Ø)
```

You may then load SCRIPSIT in the usual manner, and print using the ⟨**BREAK**⟩ P,S ⟨**ENTER**⟩ sequence. One word of caution is required here, though. This uses the ROM drivers, which do not allow for "handshaking," so you must use a baud rate low enough to allow the printer to keep up.

Question 6: I use Model III and a serial printer. I use SETCOM ⟨**ENTER**⟩ to initialize the RS-232, but my printer doesn't work. What happened?

Answer: Even though it appears to, Model III TRSDOS does NOT initialize SETCOM to a set of default values. Each time TRSDOS is rebooted, you must re-execute a complete SETCOM command, specifying BAUD, WORD, PARITY, and STOP.

Question 7: Why does "In-Memory Information" not load on my Model I? The Model III side loads at the store without a problem.

Answer: The Model I version loads a short program which must be executed by typing "/ ⟨**ENTER**⟩" at the *? prompt. The rest of the program then loads as a data file and when the loading is complete, the first screen appears.

Computer Customer Services Address and Phone Numbers

8AM to 5PM Central Time

Computer Customer Services
400 Atrium, One 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

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

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

Switchboard—1-817-390-3583

Fort Worth Scene (From page 11)

DELETES each document from the disk in drive one, so BACKUPs are very important. Please note that each document on the 1.0 disk must have the same password as every other document on the disk for this function to work properly.

GENEALOGY:

We have had at least one request for an article on Genealogy Research with Micro's. If you are using your TRS-80 for Genealogy, you might put together an article for us to publish.

TINY PASCAL:

One of our readers is looking for "short and efficient" Tiny Pascal subroutines for Log n and Exp. If you have created useful subroutines for Tiny Pascal, we would like to publish them for other users.

VISICALC FOR THE COLOR COMPUTER:

VisiCalc will not be available for the Color Computer. What we have, however, is a similar program on ROM Pak called Spectaculator (26-3104). If you are looking for this type of software, drop by your local Radio Shack store, Spectaculator should be in stock now.

AGRI-BUSINESS PEOPLE:

Have you written an agricultural program for the TRS-80? If you have, please send your name, address, phone number, the name of and a brief description of the program. (Please do not send the program itself).

Radio Shack
Department AX10
One Tandy Center
Fort Worth, Texas 76102

IN CASE YOU HAVEN'T HEARD:

October, 1981 - Tandy Corporation and Matra S.A. of France have announced that they have signed a definitive agreement to set up a manufacturing operation in France to initially manufacture TRS-80 Model III microcomputers. The agreement is subject to approval of the boards of directors of both companies and formal approval of the French government.

October, 1981 - John Roach, President and CEO of Tandy Corporation, announced that the company will install and operate its own electronic information database (videotex) for the Fort Worth/Tarrant County Texas area. The objectives of the Tandy videotex service will be to provide subscribers with continuously updated information, on demand, 24 hours a day. A TRS-80 Model II computer and the newly developed Communications Multiplexer will comprise the database equipment.

Bruce Elliott—Editor
Linda Miller—Writer

Direct-Connect Modem II

This month I have another new product for you. We should be shipping the first of the new modem 26-1173, Direct-Connect Modem II by the end of January. This modem has many advanced features in a compact unit at a price that is a real bargain! It is a stand-alone modem which connects directly to the phone line and supports auto-dial and auto-answer operation.

The modem section is a standard Bell 103J compatible device. Its baud rate is 0 to 300. Transmit level is −10dBm. The receive sensitivity is −40dBm. We had originally specified a sensitivity of −45 dBm as many other units on the market boasted of this level. We found in "real world" testing that this level of sensitivity is too great! At least one major network system had a ring signal with strong harmonics in the frequencies used by the modem signaling, producing lots of confusion! By reducing the sensitivity a little, no degradation of performance has been observed and the ring response problem has been eliminated.

D.C. Modem II has four operating modes:

1. Manual answer
2. Manual originate
3. Auto-Answer
4. Auto-originate

It also includes two test modes. There is a local test which tests the operation of the unit itself and a remote loopback which performs test of the total communication link.

In the manual modes the unit operates in the same manner as the 26-1172 D.C. Modem I (except that it does not work with the Model I cassette port). For the automatic mode an on-board microprocessor takes over! There are several modems on the market which can perform some of the same functions with a fairly large overhead of software in the host computer. The microprocessor in the D.C. Modem II is programmed by simple ASCII commands sent from the host. The Modem does all the work.

There are 12 commands briefly outlined below:

D Dial command - this precedes the number to be dialed.

P Pause - this inserts a two second pause to allow local systems to access outside lines.They can be cascaded to allow pauses of any length. i.e. PPPP

C Clear - Clears all registers to default state

L Look - causes the Modem to dump the contents of the user programmable registers to the host terminal at 300 Baud.

R Rotary Dial - pulse dialing is used

F Fast Dial - selects 20 PPS pulse dial rate

S Slow Dial - selects 10 PPS pulse dialing

T Tone Dial - Dials the number with Bell DTMF tones at 10 digits per second. (By the way, Tone and Pulse dialing can be intermixed for those of you that live in an area not supporting tone dialing but who need to access systems which require tone signalling.)

Q Local Loopback test - same as Local test switch.

O Originate - puts Modem in Auto-Originate mode

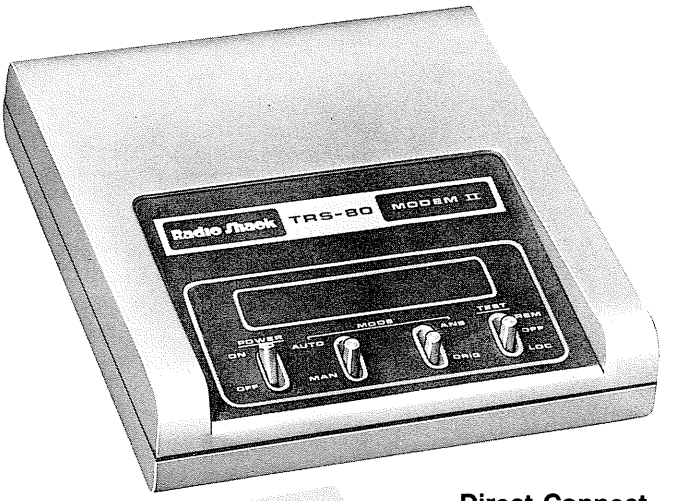
X Execute - Follows a command string to cause execution of the command.

* - An asterisk opens the modem memory for programming.

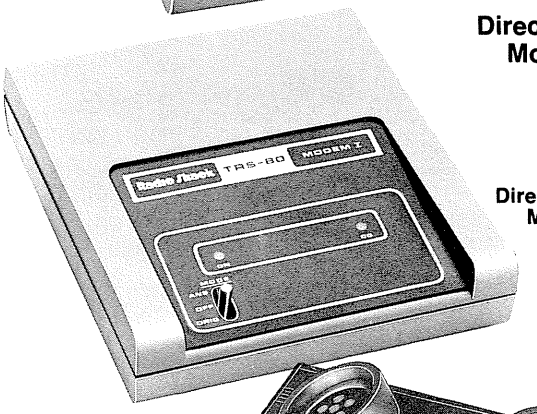
Here is a typical command string:

*DR4358214PT645218576352X

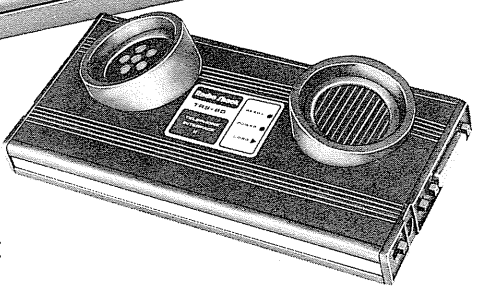
Any BASIC or machine language program can be used to send programming information to the modem. In most cases, the programming code (similar to a driving routine) can be appended to the front of the communications program. The simple commands control a powerful device opening a new world of possibilities. We are in the processes of installing a Model III based communications system in every Radio Shack store in the country for inventory and accounting uses. The D.C. Modem II is an integral part of the system, allowing the big Tandem Computer system in Fort Worth to access the stores in the "dead of the night"!



Direct-Connect Modem II



Direct-Connect Modem I



Telephone Interface II

More PRINT USING

We published an article entitled "Using PRINT USING" in the August '81 issue of Microcomputer News that has generated quite a bit of response from readers. I would like to share with you their comments.

PRINT USING is a valuable Level II or Model III BASIC statement that comes in handy for screen formatting or report preparation. With PRINT USING and the addition of PRINT @ as described in the August article it is easy to create professional looking reports by aligning dollar signs, decimal points and even controlling the spacing between columns of numbers. Even if you don't have that August article handy, PRINT USING is described in your Model III Operation and BASIC Reference manual or your Model I Level II Users Manual.

Here are some areas that weren't addressed in that article but are covered by our readers in the following letters:

1) How can I easily produce multiple column reports?
2) How about using PRINT USING for printers?
3) Finally since a format string can be up to 255 characters long and can contain any number of line feeds and literals some advanced PRINT USING techniques are available.

J. Nelson Phillips
Route 5, Box 69
Shelbyville, TN 37160

The technique shown in the August article is limited in its usefulness. It does not allow the use of more than one 'PRINT USING' format on any one print line, for example.

A more elegant solution which allows total flexibility is shown below:

```
100 PRINT TAB(M) USING A$; W;
110 PRINT TAB(N) USING B$; X;
120 PRINT TAB(P) USING C$; Y;
130 PRINT TAB(Q) USING D$; Z
```

Note the semi-colon appended to all but the last PRINT statement. The semi-colons inhibit the CARRIAGE RETURN at the end of the line. Thus, when the computer executes Line 100, no CARRIAGE RETURN occurs. Then Line 110 is executed, with the variable 'X' being printed on the same line and to the right of 'W'. This continues until a line without the appended semi-colon is encountered. Then at the end of that line, a CARRIAGE RETURN occurs, terminating the printing on that line.

This technique works nicely within a FOR-NEXT loop, particularly with engineering analysis programs where succeeding columns of data may require varying PRINT USING formats.

This technique may require the insertion of semi-colons preceding 'USING' when a Model I, Level II machine is employed.

* * *

Quincy G. Leslie
SNELLING & SNELLING
401 W Front Street, Suite L-3
Traverse City, MI 49684

Quite a nice article on "Using PRINT USING" appeared in your August issue, Volume 3, Issue 8. However, the thrust of its information seemed oriented to those restricting "PRINT USING" to CRT screen printing only.

The response to the question of combining the PRINT TAB(n) statement with PRINT USING was "Do not try. . . ." Still the difficulty remains for those who are programming for the automatic printing of various forms, for which the LPRINT statement combining TAB and USING is essential.

LPRINT TAB(n) and LPRINT USING can be combined! At least on Model III's driving Daisy Wheel Printer II's, on which the following has worked:

```
140 LPRINT TAB (n) USING E$; B
```

which will print B, with line feed and carriage return.

Multiple columns print with this format:

```
140 LPRINT TAB(n) USING E$; A;: LPRINT TAB(n)
    USING B$;B;: LPRINT TAB(n) USING F$; C
```

Perceptive readers will note that multiple column printouts are really several complete LPRINT statements in the same program line, separated by ":" 's and incorporating trailing ";" at the end of each statement to suppress line feed/carriage return. This feature in effect causes each variable to be printed in the USING format specified, without any line feed or carriage return. Therefore, the next variable to be printed falls on the very same line with the previous variable(s).

The trailing ";" is deleted in the final LPRINT statement to force the LF/CR.

* * *

Earl R. Kooi
2196 Albright Avenue
Upland, CA 91786

Relative to the article on "USING PRINT USING" on page 6 of the August, 1981 issue of Microcomputer News, the solution is fine as long as the built-in tab stops fit the spacing desired. However, if more control over the spacing is desired, I believe the attached solution is preferable.

This solution works fine on my Model I.

```
10 CLS
20 FOR K=1000 TO 2000 STEP 100
30 A=K
40 B=A/3
50 C=A/4
60 A$="$$#,###.##"
70 LPRINTTAB(10)USINGA$;A;
    : LPRINTTAB(25)USINGA$;B;
    : LPRINTTAB(40)USINGA$;C
80 NEXT
```

```
$1,000.00       $333.33         $250.00
$1,100.00       $366.67         $275.00
$1,200.00       $400.00         $300.00
$1,300.00       $433.33         $325.00
$1,400.00       $466.67         $350.00
$1,500.00       $500.00         $375.00
$1,600.00       $533.33         $400.00
$1,700.00       $566.67         $425.00
$1,800.00       $600.00         $450.00
$1,900.00       $633.33         $475.00
$2,000.00       $666.67         $500.00
```

* * *

Johnny Bond
4000 Crestview Drive
Huntsville, AL 35805

ADVANCED PRINT USING

The article on PRINT USING in the August issue of MICROCOMPUTER NEWS prompted me to share a little more about the power of PRINT USING. These two short programs should give the reader some insight as to how far you can carry a good thing!

In PROGRAM #1, Line 20, we define S$ with the literal, "NAME: ", followed by "% %", a field to accommodate a 10 character string, then another literal, "BIRTHDATE:". The first "##" after "BIRTHDATE: " will accommodate a two digit month, the "/" will be printed, then a "##" for a two digit day, another "/" and a two digit year. Lines 30-60 input the necessary information and Line 70 prints it using S$. Now, hold your breath, RUN it and what you see should vaguely resemble SCREEN #1. Sure makes line formatting a lot easier, does it not!

So much for kid stuff, let us go for PAGE formatting!

PROGRAM #2 does basically the same thing except that Line 20 we add some LINE FEEDs. A LINE FEED is the down arrow at the left of your keyboard. It will not print either on your screen or your printer as a down arrow, because it is recognized as a LINE FEED. I have inserted a (↓) in the positions where the LINE FEED should be inserted. There is also some additional information this time: a literal "TELEPHONE:", with space for an area code and the local number.

Lines 30 - 90 again input the necessary information. Line 90 accepts the phone number as one single precision number, and Lines 100 and 110 divide it into the customary format for printing, just for the sake of a demonstration. Line 130 prints it all, using the new S$, and the results should look like SCREEN #2.

Format string (S$ in these programs) can be up to 255 characters long and can contain any number of LINE FEEDS and literals. The literals may consist of almost all TRS-80 generated characters. If a . (period) is placed between TWO numeric ("#") fields it will be seen as a decimal point in the middle of ONE numeric field. A " (double quote) can not be used for obvious reasons, and for some reason "-" (minus sign or hyphen) will not print if used between two numeric fields.

I hope this will help some readers in their programming as much as it has me.

PROGRAM #1

```
10 CLS
20 S$ = " NAME: %         % BIRTHDATE: ##/##/##"
30 INPUT "NAME"; N$
40 INPUT "YEAR OF BIRTH (2 DIGITS)"; YB
50 INPUT "MONTH OF BIRTH (NUMBER)"; MB
60 INPUT "DAY OF BIRTH"; DB
70 PRINT USING S$; N$, MB, DB YB
80 PRINT
90 END
```

SCREEN #1

PROGRAM #2

```
10 CLS
20 S$ = "NAME: %         % ↓
BIRTHDATE : ##/##/## ↓
TELEPHONE : (###) ###/####"
30 INPUT "NAME"; N$
40 INPUT "YEAR OF BIRTH (2 DIGITS)"; YB
50 INPUT "MONTH OF BIRTH (NUMBER)"; MB
60 INPUT "DAY OF BIRTH"; DB
70 INPUT "AREA CODE"; AC
80 INPUT "PHONE NUMBER (FORMAT #######)"; PN
90 T1 = INT (PN / 10000)
100 T2 = PN - (T1 * 10000)
110 PRINT
120 PRINT USING S$; N$, MB, DB, YB, AC, T1, T2
130 PRINT
140 END
```

SCREEN #2

* * *

Model I/III Bugs, Errors and Fixes

MODEL III TRSDOS (26-312)

The following patches will allow Model III Disk DEBUG to inspect and/or change addresses in low memory RAM and ROM (read only):

```
PATCH *5:0 (ADD=4EDF,FIND=38E6,CHG=0000)
PATCH *F:0 (ADD=4F04,FIND=D0,CHG=C9)
PATCH *5:0 (ADD=506E,FIND=38E3,CHG=0000)
```

DEBUG has not been fully tested in low memory addresses and caution should be exercised while using DEBUG in low memory.

ACCOUNTS RECEIVABLE 26-1555

Model I, Version 3.0 Accounts Receivable prints zero balance statements.

The problem is corrected by following the steps listed below:

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

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

Old Line:

```
91Ø IF(CVD(DB$)=ØANDCVI(DH$)=Ø)OR
     ASC(DF$)=7THENRETURNELSEGOSUB129Ø:AB#=PB:
     GOSUB148Ø:SV$=SV$(ST)
```

New Line:

```
91Ø IF CVD(DB$)=ØORASC(DF$)=7THENRETURN
     ELSEGOSUB129Ø:AB#=PB#:GOSUB148Ø:SV$=SV$(ST)
```

NOTE: Prior to 3.0:

Old Line:

```
1Ø35 IF(CVD(DB$)=ØANDASC(DH$)=Ø)ORASC(DF$)=7
     THEN112Ø
```

NEW Line:

```
1Ø35 IFCVD(DB$)=ØORASC(DF$)=7THEN112Ø
```

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

DISK PAYROLL (26-1556)

In Model I Version 2.0 Disk Payroll, after correction is made to line 1026 errors are found when adding state tax tables.

If adding line 1026 causes "PR4RGST" to go over 10 sectors, delete lines 6 through 22 and add this:

```
6 IN$="":W$=INKEY$:WL=Ø:PRINTSTRING$(FL,136)
     STRING$(FL,24);
15 FORW=ØTO1:PRINTCHR$(W+14);:FORWS=1TO25:
     W$=INKEY$:IFW$=""THENNEXTWS,W:GOTO15
25 PRINTCHR$(14);:IFW$="@"THENIN$=W$:W$=CHR$(13)
     ELSEIFW$>CHR$(31)THEN75
35 IFW$=CHR$(13)THENPRINTSTRING$(FL-WL,32)
     CHR$(15);:W=1:WS=25:NEXTWS,W:RETURN
45 IFW$=CHR$(24)THENPRINTSTRING$(WL,24);:GOTO5
55 IFW$<>CHR$(8)ORWL=ØTHEN15ELSEPRINTCHR$(24);
65 IN$=LEFT$(IN$,LEN(IN$)-1):WL=WL-1:
     POKE16418,136:GOTO15
75 IFFL=WLTHEN15ELSEPRINTW$;:IN$=IN$+W$:WL=WL+1:
     IFABS(FL)<>1THEN15ELSEW$=CHR$(13):GOTO35
```

PROFILE (26-1562)

In both Model I and III Version 3.0/3.1, "Illegal Access Attempt To Protected File" errors have been reported to occur when SORTING.

The following patches should be applied to 3.0 and 3.1 PROFILE if you have the Model III computer:

1. At TRSDOS READY type the patches in with ⟨**ENTER**⟩ after each line.

```
PATCH SORT (ADD=7854,FIND=5816ØØØ6,CHG=B728Ø5AF)
PATCH SORT (ADD=7858,FIND=Ø5AFCB13,CHG=C6321ØFC)
PATCH SORT (ADD=785C,FIND=CB121ØF9,CHG=5F16ØØØØ)
```

If you have a Model 1, you will need to run the following program with a backup copy of PROFILE 3.0 program disk in drive 1.

1. Load BASIC and type in the following program:

```
2Ø F$="":OPEN"R",1,"SORT"
3Ø FIELD1,152AS D$,12 AS C$
4Ø GET 1,2
5Ø FOR I=1TO12
6Ø READ J:F$=F$+CHR$(J)
7Ø NEXT
8Ø LSET C$=F$
9Ø PUT 1,2
1ØØ CLOSE
11Ø END
12Ø DATA 183,4Ø,5,175,198,5Ø,16,252,95,22,Ø,Ø
```

2. Type RUN
3. Use the corrected program disk to continue running PROFILE.

The RANDOM FILE sample in the manual has several lines that need to be changed so that it will work properly.

When typing in the program the following line changes should be made.

Old Line:

```
32Ø FORI=1TOMD+1:OPEN"I",1
     "PRODAT:"+CHR$(47+1):NEXTI
```

New Line:

```
32Ø OPEN"R",1,"PRODAT:"+CHR$(47+1),RL
```

Old Line:

```
34Ø D=D$LN(J)
```

New Line:

```
34Ø D=D+LN(J)
```

Old Line:

```
39Ø DN=1:FORI=1TOMD+1:IFN>D(I)THENDN=I+1
```

New Line:

```
39Ø DR=1:FORI=1TOMD+1:IFN>D(I)THENDR=I+1
```

Old Line:

```
4ØØ NEXT:NN=N-D(DN-1):GETDN,NN
```

New Line:

```
4ØØ NEXT:NN=N-D(DR-1):GETDR,NN
```

Old Line:

```
41Ø IFASC(R$(1,DN))=192THENPRINT"**DELETE**"
     :GOTO46Ø
```

New Line:

```
41Ø IFASC(R$(1,DR))=192THENPRINT"**DELETE**"
     :GOTO46Ø
```

Old Line:

```
44Ø PRINT@CU(I)-15374,NM$(I);":";R$(I,DN);
```

New Line:

```
44Ø PRINT@CU(I)-15374,NM$(I);":";R$(I,DR);
```

* * * The remainder of the program should be left as is.

On page 30 of the manual, there is a sample of "USING SEQUENTIAL ACCESS"

The following lines should be changed:

Old Line:

```
33Ø DN=1:FORJ=1TONR:IFJ>D(DN)THENDN=DN+1
```

New Line:

```
33Ø DR=1:FORJ=1TONR:IFJ>D(DR)THENDR=DR+1
```

Old Line:

```
34Ø LINEINPUT#DN R$:IFLEN(R$)=ØTHEN34Ø
```

New Line:

```
34Ø LINEINPUT#DR,R$:IFLEN(R$)=ØTHEN34Ø
```

Old Line:

```
36Ø DY!=VAL(MID$(R$,32,6)):PD!=VAL(MID$(R$,32,6))
```

New Line:

```
36Ø DY!=VAL(MID$(R$,52,6)):PD!=VAL(MID$(R$,58,6))
```

* * * The remainder of the program should be left as is.

RSCOBOL (26-2203)

Currently, when RUNCOBOL (Model I/III) attempts to CALL an independent segment overlay, it will have a LOAD FAIL ERROR message and return to TRSDOS. RUNCOBOL is unable to load and execute the machine language program (independent segment). The following patches will correct this problem (Note: a close parenthesis is not required in the PATCH command):

```
PATCH RUNCOBOL/CMD (ADD=9DB8,FIND=CD11AEC2419F,
                    CHG=ØØØØØØØØØØØØ
PATCH RUNCOBOL/CMD (ADD=9DC4,FIND=2A,CHG=21)
PATCH RUNCOBOL/CMD (ADD=ADØD,FIND=19,CHG=Ø9)
PATCH RUNCOBOL/CMD (ADD=ACFE,FIND=D6ØA,CHG=C318)
PATCH RUNCOBOL/CMD (ADD=ADØØ,FIND=3Ø,CHG=9A)
PATCH RUNCOBOL/CMD (ADD=9A18,FIND=ØØØØØØØØ,
                    CHG=FEØADAØ4
PATCH RUNCOBOL/CMD (ADD=9A1C,FIND=ØØØØØØØØ,
                    CHG=ADD611DA
PATCH RUNCOBOL/CMD (ADD=9A2Ø,FIND=ØØØØØØØØ,
                    CHG=11ADFEØ6
PATCH RUNCOBOL/CMD (ADD=9A24,FIND=ØØØØØØØØØØØØ,
                    CHG=D211ADC3Ø2AD
PATCH RUNCOBOL/CMD (ADD=9B65,FIND=2A69AE4E23,
                    CHG=46ED43D1AF
PATCH RUNCOBOL/CMD (ADD=9B6A,FIND=46ED43D1AF,
                    CHG=CD8CA82Ø5C
PATCH RUNCOBOL/CMD (ADD=9DCB,FIND=489F,CHG=D49D)
PATCH RUNCOBOL/CMD (ADD=986Ø,FIND=CD8CA82Ø66,
                    CHG=2A69AE4E23
```

PRINT to LPRINT Improved

Al Reudisuelli
Box 6275
Chattanooga, Tn 37401

Please note that the PRINT to LPRINT and the LPRINT to PRINT conversion techniques which are printed on page 39 of the November '81 issue of the newsletter can cause unexpected problems. BASIC statements are stored in memory as follows:

```
XX (LSB)        : TWO-BYTE ADDRESS OF THE
XX (MSB)        : NEXT PROGRAM LINE.
XX (LSB)        : LINE NUMBER OF THIS
XX (MSB)        : PROGRAM LINE.
XX XX XX ...    : PROGRAM TEXT USING TOKENS.
ØØ              : ZERO INDICATES THE END OF THIS LINE.
```

The series is repeated until the end of the program where 00 00, a pair of hex zeros, tells the BASIC interpreter that there are no more program lines.

I've coded a small program to demonstrate that the number 178 can occur in a BASIC program more often than the number of occurrences of the PRINT statement. Recall that 178 is the token for PRINT.

You must run the program in Level II BASIC for the numbers to demonstrate the problem, but here's what it boils down to:

Line 10 has 179 numbers in it along with the REM command. That means that the starting address of line 20 will be 17330, and when you convert 17330 into the MSB/LSB numbers that BASIC uses, it becomes 67 for the MSB and 178 for the LSB. VOILA! There's that magic number, 178, and it doesn't refer to a PRINT command at all - it's part of an address!

Using the technique in the first "PRINT to LPRINT" article would change the address and give you funny-looking results when you tried to LIST the program. As a matter of fact, you can

```
POKE PEEK(16548)+256* PEEK(16549), PEEK(16548):
POKE PEEK(16548)+1+256* PEEK(16549), PEEK(16549)
```

and make your program list only the first line repeatedly because in effect, you'll be saying to the LIST function that the beginning of the next line is at the beginning of the BASIC program! You can restore the LIST function by POKEing back the values that were there originally or by deleting that first line. Try it. It's fun to experiment with and there are more surprises which I'm not gonna tell y'about.

Now about conversion #2 - Level II BASIC begins at memory address 17129, and you can prove that by executing the statement,

```
PRINT PEEK(16548) + 256 * PEEK(16549)
```

In level II BASIC, those two addresses are the beginning-of-BASIC-program pointer. There is an end-of-BASIC-program pointer located at addresses 16633 and 16634, so you can write a program to examine or change memory just within the limits of your BASIC program. The Conversion #2 program on page 39 of the November '81 Newsletter will begin looking in memory at a place before the author's BASIC program since he's using Disk BASIC which does not start at 17129 like Level II BASIC but at a higher address.

Fortunately, he can use the beginning-of-BASIC pointer to change things from the beginning of his program. The pointer correctly points to the beginning of the program in both Level II BASIC and Disk BASIC. Also, the author's program would scan through all the memory up to address 32767 except that line 65425, as printed, will give a SYNTAX error since the word "TO" is missing in the FOR/NEXT loop. The address, 32767, is far below the top of the memory in a 48K machine which is what the author said he's using, so he might not get all the changes made that he's anticipated.

He could cover the full range of his program if he used the beginning-of-BASIC-program and end-of-BASIC-program pointers. The beginning of the FOR/NEXT statement to cover the range of the BASIC program would be:

```
FOR J=PEEK(16548)+256*PEEK(16549) TO
     PEEK(16633)+256*PEEK(16634)
```

To sum up, both of the conversion routines in the newsletter may work, but both can cause you problems you may not have anticipated, and if you didn't back up your program before you started POKEing around, you could lose parts.

SAMPLE PROGRAM

```
1Ø REM12345678911234567892123456789312345678941234567895123456
     7896123456789712345678981234567899123456789Ø12345678912
     3456789223456789323456789412345678951234567896123456789 7
     12345678981234567899123456 78
2Ø K=PEEK(16548)+256*PEEK(16549)
3Ø IF K<>17129 THEN PRINT "YOU MUST RUN THIS IN LEVEL II
     BASIC"
     : STOP
4Ø FOR J=K TO K+2Ø
5Ø LPRINT PEEK(J);
6Ø NEXT
```

• • •

John C Miller
110 Riverside Drive, #14C
New York, NY 10024

Both short programs for converting PRINT statements to LPRINT and vice versa, published in your November 1981 issue, contain three defects. In order of increasing seriousness, these defects are:

(1) Each revises the entire program, where typically only a portion should be changed,
(2) both will fail with programs whose program text extends above location 32767, and
(3) both will often destroy a program in which the byte being changed (175 or 178) occurs as part of a line number or line pointer in the BASIC program. The programmer normally has no need to be aware of these values, and may be distressed to find that his program has been rendered unusable as he innocently attempts to use these routines.

The enclosed routine avoids these problems by prompting for the range of line numbers in which the changes are desired, and scanning only those lines. Also, it will not give illegal function call errors when the text extends into higher memory. Finally, it corrects only legitimate occurrences of the desired bytes in the text portion of each BASIC line scanned.

```
64999 END
65ØØØ DEFSNG Z
65Ø1Ø ZØ=Ø
     : INPUT"MODIFY LINES STARTING AT NUMBER
     (DEFAULT=Ø)"; ZØ
65Ø2Ø Z1=64998
     : INPUT "ENDING AT NUMBER (DEFAULT=64998)";
     Z1
65Ø3Ø A$="P"
     : INPUT "L=LPRINT (REPLACE PRINT); P=PRINT
     (REPLACE LPRINT) (DEFAULT=P)"; A$
65Ø4Ø N=178+3*(A$="L")
     : M=353-N
     : REM M=VALUE TO BE REPLACED IF FOUND, N=NEW
     VALUE
65Ø5Ø Z=16548
     : GOSUB 6514Ø
     : REM INITIALIZE: ZC=LOC OF START OF BASIC
     PROGRAM
65Ø6Ø Z=ZC+2
     : GOSUB 6514Ø
     : ZL=ZC
     : Z=Z-2
     : GOSUB 6514Ø
     : REM ZL=CURRENT LINE NUMBER, Z=START LOC OF
     CURRENT LINE, ZC=START OF NEXT LINE (OR ZC=Ø
     IF ALREADY FINISHED LAST LINE.)
65Ø7Ø IF ZC=Ø OR ZL>Z1 THEN END
65Ø8Ø IF ZL>=ZØ THEN GOSUB 651ØØ
     : REM CALL REPLACEMENT ROUTINE IF LINE
     NUMBER IS IN TARGET RANGE
65Ø9Ø GOTO 65Ø6Ø
651ØØ FOR Z=Z+4 TO ZC-2
651Ø Ø ZW=Z+65536*(Z>32767)
651Ø Ø IF PEEK(ZW)=M THEN POKE ZW, N
651Ø Ø NEXT
     : RETURN
     : REM FOR LINE IN LOCATIONS Z THROUGH ZC-1,
     REPLACE ALL OCCURRENCES OF BYTE M WITH BYTE
     N
6514Ø ZC=PEEK(Z+65536*(Z>35767)) +
     256*PEEK(Z+1+65536*(Z>32766))
6515Ø RETURN
     : REM ZC=CONTENTS OF MEMORY LOCATIONS Z,Z+1
     AS A 16-BIT UNSIGNED INTEGER, WHERE Z MAY
     EXCEED 32767.
```

Routine to change PRINT statements to LPRINT and vice versa.

&, &H and VAL

Bill Dickson
OFFSHORE NAVIGATION, INC.
P.O. Box 23504
New Orleans, LA 70183

If you have a color computer with Extended BASIC or a Model I, III and Disk BASIC, you may not be aware of a capability you have. (This also applies to Microsoft's Level III BASIC).

The two functions '&' and '&H' are used to convert octal or hexadecimal constants to decimal values. This is a convenient feature but if you want to use this function on a variable within a program, you may be irritated to find that it won't work!! It is only designed to work with a constant.

Because the information following these statements (& and &H) is treated as a string constant without quotation marks, any variable following these statements will be treated the same (e.g. &HA$) and the result will be garbage.

The key to fooling the interpreter is the VAL function. Assuming that A$ = "FF", use this statement to convert A$ to a decimal number.

A = Val ("&H" + A$)

After this statement is executed the variable 'A' will equal 255. Use of the octal function is identical except that you substitute '&' or '&O' for '&H'. Assuming A$ = "7777"

A = Val ("&" + A$)

After the above statement the variable 'A' will equal 4095.

This may not be a source listing for a high powered machine language utility, but in the long run may be as useful if you have a need for this function. None the less I am happy to contribute to the cause and hope your publication encourages more of the same type of helpful hints and fewer "Space Invaders".

FORTRAN/BASIC Data Files

John L. Montgomery
3010 Barcody Road
Huntsville, AL 35802

From time to time I find it necessary to read and write disk files from both BASIC and FORTRAN. I thought a short discussion and two demonstration programs might be of interest to other users of these two languages. I have enclosed two listings, one of the source code for FORTRAN, the

other BASIC. These demonstrate how to read and write files between the two languages.

The technique is really quite straightforward it only requires that you know the precise format of the files that you want to read and write. The BASIC read of a FORTRAN file is the same as the reading of a BASIC file except that you must field the buffer in a manner which represents the way that the FORTRAN file is written. You need to know what type data (integer, char, double, etc) and in what order they were written. The FORTRAN read of a BASIC file is similar, the main point is that the read is done unformatted with the data types specified and positions matched to the BASIC field statement. The unformatted reads and writes in FORTRAN produce the same conversions as the MKD$ and CVD, etc. functions in BASIC.

I hope the programs are sufficiently clear to be understood by most people. One more observation—these programs use the Model I but the lack of user definable record lengths complicates the process. Also, it is not stated in the FORTRAN manual, but if you use Formatted disk I/O then the specified logical record length in the Call Open statement must be at least one greater than the sum of the Format Field widths, in the examples the OPEN would have to have been for a logical record length of 40 or more. In the version for TRSDOS 1.2 random access to files with logical record lengths of 256 just did not work correctly, consequently the programs described above may not work correctly on TRSDOS 1.2.

FORTRAN PROGRAM

```
10 C      THIS PROGRAM IS TO DEMONSTRATE READING
    AND WRITING
11 C      FORTRAN AND BASIC COMPATIBLE FILES,
    FORTRAN USES THE
12 C      UNFORMATTED READ AND WRITES
13 C      CHARACTER DATA TYPED AS BYTE
30        DOUBLE PRECISION C
40        INTEGER D
50        REAL B
60        BYTE A(25)
70 C      ASK IF READING OR WRITING
80        WRITE(1,900)
90 10     READ(1,910) X
100       IF (X.EQ.1.) GOTO 100
110       IF (X.EQ.2.) GOTO 200
120       GOTO 10
129 C     READ A BASIC FILE AND PRINT DATA TO
    SCREEN
130 100   CALL OPEN(6,'BASFILE/DAT',39)
140       DO 110 I=1,25
149 C     READ 25 CHARACTERS 1SNGL,1DBL, 1 INT
    NUMBERS
150       READ(6,REC=I,ERR=9999)
    (A(J),J=1,25),B,C,D
159 C     WRITE RESULTS OF DISK READ TO SCREEN
    NOTE FORMAT
160       WRITE(1,1000)  (A(J),J=1,25),B,C,D
170 110   CONTINUE
180       GOTO 9999
199 C     WRITE A FORTRAN FILE 25 CHARACTERS F
    SINGLE,DOUBLE,INTEGER
200 200   CALL OPEN(6,'FORFILE/DAT',39)
210       DO 210 I=1,25
220 C     BUILD CHARACTER STRING 25 F'S
230       DO 205 J=1,25
240       A(J)='F'
250 205   CONTINUE
260 C     MAKE UP NUMBERS
270       B=12.*I
280       C=100.*I
290       D=I
300 C     WRITE DATA ON SCREEN
310       WRITE(1,1000)  (A(J),J=1,25),B,C,D
320 C     WRITE DATA TO DISK 25 CHAR 1 SNGL,
    1DBL,1INT NUMBERS
330       WRITE(6,REC=I,ERR=9999)
    (A(J),J=1,25),B,C,D
340 210   CONTINUE
520 900   FORMAT(1X,'  SELECT THE OPTION  1-READ
    BASIC, 2--WRITE FORTRAN',/)
530 910   FORMAT(F6.0)
540 1000  FORMAT(1X, 25A1, F6.0, F10.0, I2)
550 C     EXIT HERE NORMALLY OR IF ERROR
560 9999  ENDFILE 6
570       END
```

BASIC PROGRAM

```
1 '       THIS PROGRAM IS TO DEMONSTRATE READING
    AND WRITING
2 '       FORTRAN AND BASIC COMPATIBLE DISK
    FILES
3 '       BASIC USES DIRECT ACCESS FILES 39
    BYTES LONG
4 '       TO ACCOMMODATE A 25 BYTE STRING 1SNGL,
    1DBL, 1 INT NUMBER
5 '       THE READS AND WRITES ARE THE SAME AS
    STANDARD BASIC
10 CLEAR 1000
20 PRINT " SELECT AN OPTION, 1- WRITE BASIC, 2-
    READ FORTRAN ";
    : INPUT X
30 IF X=1 THEN 100
40 IF X=2 THEN 200
50 GOTO 20
100 OPEN "R", 1, "BASFILE/DAT", 39
110 FIELD 1, 25 AS E$, 4 AS F$, 8 AS G$, 2 AS H$
120 FOR I=1 TO 25
129 '   WRITE STRING OF B'S 1 SNGL, 1DBL, 1INT
    NUMBER
130 A$=STRING$(25,"B")
    : B$=MKS$(3.2*I)
    : C$=MKD$(112*I)
    : D$=MKI$(I)
140 LSET E$=A$
150 LSET F$=B$
160 LSET G$=C$
170 LSET H$=D$
180     PUT 1, I
190 NEXT
    : GOTO 260
199 '   READ FILE CREATED BY FORTRAN
200 OPEN "R", 1, "FORFILE/DAT", 39
210 FIELD 1, 25 AS A$, 4 AS B$, 8 AS C$, 2 AS D$
220 FOR I=1 TO 25
230    GET 1, I
240 PRINT A$; CVS(B$); CVD(C$); CVI(D$)
250 NEXT
260 CLOSE
    : END
```

Instant Recall

Dwight Dager
14 Bernard Drive
Morrisville, PA 19067

Instant Recall is a BASIC program designed to test one's mental recall ability by matching the hidden characters that are briefly displayed behind 26 lettered blocks. These characters are randomly shuffled after each complete game, thus insuring a variety of different starting locations.

Up to two players can compete. Starting players are randomly chosen and will repeat play for every MATCH that is made. If you have a speaker/amplifier connected to the cassette port, a single tone will sound after each NO-MATCH.

All inputs are accomplished by pressing the proper let-

tered key. A score is kept with the overall high score updated after each complete game.

The program will run on a Model I (16K) Level II computer.

Possible modifications:

1. To change characters, modify line 320. A total of 13 characters must be used.
2. To change the length of time which characters are displayed, modify TD value in line 1170.

```
1Ø '*** INSTANT RECALL ***
2Ø '.........by DWIGHT DAGER
3Ø '.........MORRISVILLE,PA.
5Ø CLEAR 2ØØ
     : DEFSTR P, L, S, B, G, I
     : DEFINT X-Z, Q, N, R, M, T, O
     : DIM S(14), B(27), Q(27), N(27), L(27)
9Ø G1=STRING$(5, 191)
     : G2=STRING$(36, 14Ø)
3ØØ DATA 71, 77, 83, 89, 193, 199, 2Ø5, 211, 217,
     223, 321, 327, 333, 339
31Ø DATA 345, 351, 449, 455, 461, 467, 473, 479,
     583, 589, 595, 6Ø1
32Ø DATA #, $, %, &, *, +, ?, =, "1", "3", "5",
     "7", "9"
33Ø DATA A, B, C, D, E, F, G, H, I, J, K, L, M,
     N, O, P, Q, R, S, T, U, V, W, X, Y, Z
5ØØ '* PLAYER NAMES *
51Ø CLS
     : O(1)=Ø
     : O(2)=Ø
     : INPUT "HOW MANY PLAYERS - (2 MAX.)"; N
     : PRINT
     : PRINT "PLEASE ENTER YOUR NAME (14 LETTERS
     MAX.):"
     : PRINT
     : FOR X=1 TO N
52Ø PRINT "PLAYER #" X;
     : INPUT P(X)
53Ø IF LEN(P(N))>14 THEN PRINT
     : PRINT "** 14 LETTERS MAX. **"
     : GOTO 52Ø
54Ø PRINT
     : NEXT X
     : CLS
6Ø5 PRINT CHR$(23)
     : PRINT@466, "T H I N K I N G"
61Ø '>PUT PRINT LOCATIONS IN Q()
62Ø RESTORE
     : FOR X=1 TO 26
     : READ Z
     : Q(X)=Z
     : NEXT X
63Ø '>PUT SYMBOLS/NUMBERS IN S()
64Ø FOR X=1 TO 13
     : READ S
     : S(X)=S
     : NEXT X
65Ø '>PUT LETTERS FOR BLOCKS IN I()
66Ø FOR X=1 TO 26
     : READ L
     : L(X)=L
     : NEXT X
67Ø 'RND 2 SETS OF #'S 1-13
68Ø RANDOM
     : FOR Z=1 TO 26
     : N(Z)=Ø
     : NEXT Z
     : FOR Z=1 TO 26
69Ø X=RND(13)
7ØØ N(Z)=X
     : Y=Ø
     : FOR Z1=1 TO 26
71Ø IF N(Z1)<>N(Z) THEN NEXT Z1, Z
     : GOTO 74Ø
72Ø Y=Y+1
     : IF Y=>3 THEN N(Z)=Ø
     : GOTO 69Ø ELSE NEXT Z1, Z
73Ø '>PUT RND SYMBOLS/NUMBERS FROM N() INTO B()
     VIA S()
74Ø FOR Z=1 TO 26
     : B(Z)=S(N(Z))
     : NEXT Z
75Ø '* DRAW BOARD OUTLINE *
76Ø CLS
     : FOR Y=Ø TO 64Ø STEP 128
     : PRINT@Y, G2;
     : NEXT Y
     : FOR X=Ø TO 72 STEP 12
77Ø FOR Y=1 TO 31
     : SET(X, Y)
     : SET(X+1, Y)
     : NEXT Y, X
78Ø PRINT@65, G1;
     : PRINT@95, G1;
     : PRINT@577, G1;
     : PRINT@6Ø7, G1;
79Ø '* PUT LETTERS INTO BLOCKS *
8ØØ FOR X=1 TO 26
     : PRINT@Q(X)+2, L(X);
     : NEXT X
92Ø '>INITIAL START
93Ø PRINT@43, "INSTANT  RECALL";
     : PRINT@1Ø3, STRING$(23, 43);
94Ø PRINT@773, "PLAYERS      TRIES MATCHES SCORE
     (HIGH SCORE)";
95Ø Z1=839
     : FOR Z=1 TO N
     : PRINT@ABS((INT((LEN(P(Z)))/2))-Z1),  P(Z);
     : Z1=9Ø3
     : NEXT Z
96Ø IF N=1 THEN POKE 162Ø8, 17Ø
     ELSE POKE 162Ø8, 17Ø
          : POKE 16727, 138
97Ø '>SET ALL COUNTERS TO ZERO/RANDOM START
98Ø FOR Z=1 TO 2
     : R(Z)=Ø
     : M(Z)=Ø
     : T(Z)=Ø
     : NEXT Z
99Ø RANDOM
     : IF N>1 THEN N1=RND(2) ELSE N1=1
1ØØØ '* FUN BEGINS *
1Ø1Ø PRINT@295, "PLAYER:";
     : PRINT@ABS(INT((LEN(P(N1))/2)-31Ø)), P(N1);
1Ø2Ø ')INPUTS
1Ø3Ø PRINT@487, "(PRESS KEY ONLY)";
     : PRINT@359, "ENTER 1ST LETTER";
1Ø4Ø I=INKEY$
     : IF I<>"" THEN X1=ASC(I)
     : GOTO 1Ø6Ø
1Ø5Ø POKE 15718, 143
     : FOR TD=1 TO 5Ø
     : NEXT TD
     : POKE 15718, 128
     : FOR TD=1 TO 1Ø
     : NEXT TD
     : GOTO 1Ø4Ø
1Ø6Ø IF X1<65 OR X1>9Ø THEN 1Ø4Ø ELSE X1=X1-64
1Ø7Ø IF Q(X1)=Ø THEN 1Ø4Ø ELSE PRINT@Q(X1)+2,
     B(X1);
     : PRINT@Q(X1), CHR$(14Ø);
     : PRINT@Q(X1)+4, CHR$(14Ø);
     : PRINT@423, "ENTER 2ND LETTER";
1Ø8Ø I=INKEY$
     : IF I<>"" THEN X2=ASC(I)
     : GOTO 11ØØ
1Ø9Ø POKE 15782, 143
     : FOR TD=1 TO 5Ø
     : NEXT TD
     : POKE 15782, 128
     :  FOR TD=1 TO 1Ø
     : NEXT TD
     : GOTO 1Ø8Ø
11ØØ IF X2<65 OR X2>9Ø THEN 1Ø8Ø ELSE X2=X2-64
111Ø IF Q(X2)=Ø THEN 1Ø8Ø ELSE PRINT@Q(X2)+2,
     B(X2);
     : PRINT@Q(X2), CHR$(14Ø);
```

```
     : PRINT@Q(X2)+4, CHR$(14Ø);
     : FOR Y=359 TO 487 STEP 64
112Ø PRINT@Y, CHR$(2Ø9);
     : NEXT Y
     : PRINT@167, CHR$(215);
113Ø '>MATCH/NO MATCH
114Ø IF B(X1)=B(X2) THEN M=1
     : M(N1)=M(N1)+1
     : PRINT@616, "*** M A T C H ***"; ELSE M=Ø
     : PRINT@617, "NO - M A T C H";
     : GOSUB 211Ø
115Ø R(N1)=R(N1)+1
     : IF M(N1)>Ø THEN T(N1)=(M(N1)/R(N1))*1ØØ
117Ø GOSUB 2ØØØ
     : FOR TD=1 TO 15ØØ
     : NEXT TD
118Ø IF M=1 THEN PRINT@Q(X1), G1;
     : PRINT@Q(X2), G1;
     : Q(X1)=Ø
     : Q(X2)=Ø
     : GOTO 12ØØ
119Ø PRINT@Q(X1), CHR$(128);
     : PRINT@Q(X1)+4, CHR$(128);
     : PRINT@Q(X1)+2, L(X1);
     : PRINT@Q(X2), CHR$(128);
     : PRINT@Q(X2)+4, CHR$(128);
     : PRINT@Q(X2)+2, L(X2);
12ØØ PRINT@616, CHR$(21Ø);
121Ø '>END ?
122Ø FOR Y=1 TO 26
     : IF Q(Y)<>Ø THEN 124Ø ELSE NEXT Y
1222 FOR Z=1 TO N
     : IF T(Z)>O(Z) THEN O(Z)=T(Z)
1226 NEXT Z
     : GOSUB 2ØØØ
     : GOTO 13ØØ
123Ø '>REPEAT PLAYER ?
124Ø IF N>1 THEN IF M=1 THEN PRINT@167, "SAME
     PLAYER PLAYS AGAIN";
     : GOTO 1Ø3Ø
125Ø '>ADVANCE PLAYER
126Ø IF N>1 THEN IF N1=1 THEN N1=2 ELSE N1=1
127Ø PRINT@295, CHR$(215);
     : GOTO 1Ø1Ø
129Ø '>NEXT GAME ?
13ØØ FOR Y=167 TO 615 STEP 64
     : PRINT@Y, CHR$(215);
     : NEXT Y
131Ø PRINT@359, "PRESS ENTER";
     : INPUT XX
     : CLS
132Ø PRINT CHR$(23)
     : PRINT
     : PRINT "TO PLAY AGAIN PRESS > A
133Ø PRINT
     : PRINT "TO CHANGE PLAYERS PRESS > C
134Ø PRINT@514, "WAITING:";
135Ø I=INKEY$
     : IF I="A" THEN CLS
     : GOTO 6Ø5
136Ø IF I="C" THEN 51Ø
137Ø PRINT@512, CHR$(143);
     : FOR TD=1 TO 5Ø
     : NEXT TD
     : PRINT@512,  CHR$(128);
     : FOR TD=1 TO 1Ø
     : NEXT TD
     : GOTO 135Ø
199Ø '* SCORING SUBROUT *
2ØØØ Z1=849
     : FOR Z=1 TO N
     : PRINT@Z1, CHR$(222);
     : PRINT@Z1, R(Z);
2Ø1Ø PRINT@Z1+7, M(Z);
     : PRINT@Z1+15, T(Z);
     : PRINT@Z1+25, O(Z);
2Ø2Ø Z1=913
     : NEXT Z
     : RETURN
21ØØ '* SOUND > NO-MATCH SUBROUT *
211Ø FOR TD=1 TO 5Ø
     : OUT 255, 2
     : OUT 255, Ø
     : NEXT TD
     : RETURN
```

Stunt Racer

York Maksik
2055 Center Avenue
Fort Lee, NJ 07024

The instructions for operating this program are enclosed in the program itself. Stunt Racer is a game in which you control a car at the bottom of the screen and walls come speeding down to the car. The object of the game is to speed your car through as many walls as possible in the time chosen by the one playing the game.

Please publish this program for your readers as I think they will enjoy it. Note: If there are any questions or comments about this program please send them to: York Maksik, 2055 Central Avenue, Fort Lee, N.J., zip code is 07024.

Editors Comment: In the program listing, Mr. Maksik has used words and numbers to indicate the proper number of spaces to insert in literals. For instance, in line 10 you will find the literal"

"7SPACESSTUNT RACER"

You should replace the phrase "7SPACES" with seven blanks.

PROGRAM LISTING

```
1 DIM A$(9)
     : CLEAR 5ØØØ
1Ø CLS
     : PRINT CHR$(23); "7SPACESSTUNT RACER";
2Ø FOR T=1 TO 1ØØØ
     : NEXT
     : CLS
3Ø INPUT "DO YOU WISH INSTRUCTIONS"; A$
4Ø IF A$="YES" OR A$="Y" THEN GOSUB 3ØØØØ
5Ø INPUT "HOW MANY TIME UNITS DO YOU WANT (25 TO
     1ØØØ)"; A
6Ø A1=.9*(A*14)
     : PRINT "IF YOU SCORE "; A1; " POINTS YOU
     WILL RECEIVE ANOTHER"
7Ø PRINT A; " TIME UNITS"
     : FOR Y=1 TO 15ØØ
     : NEXT Y
     : CLS
8Ø INPUT "HOW MANY UNITS SHOULD EACH WALL BE (4
     TO 8)"; B
9Ø IF B<4 OR B>8 THEN PRINT "NOT A GOOD NUMBER,
     TRY AGAIN"
     : GOTO 8Ø
92 INPUT "SKILL LEVEL 1 OR 2"; Q
     : IF Q<>1 AND Q<>2 THEN 92
1ØØ CLS
11Ø FOR X=1 TO B
     : AQ$=AQ$+CHR$(191)
     : NEXT X
114 K$=CHR$(176)+CHR$(188)+CHR$(191)+CHR$(188)+
     CHR$(176)
12Ø A$(1)=AQ$
13Ø A$(2)="1Ø SPACES "+AQ$
14Ø A$(3)="22 SPACES              "+AQ$
15Ø A$(4)="37 SPACES
     "+ AQ$
16Ø A$(5)="4Ø SPACES
     "+AQ$
17Ø A$(6)="53 SPACES
     "+AQ$
```

```
185 B$="63 SPACES
    "
19Ø FOR T=1 TO A
2ØØ A2=RND(6)
2Ø2 A3=RND(6)
2Ø4 A1$=A$(A3)
21Ø A$=A$(A2)
211 IF A2 = A3 THEN 2Ø2
22Ø FOR T1=64 TO 96Ø STEP 64
221 PRINT@Ø, "TIME = "; T;
    : PRINT@5Ø, "SCORE = "; SC;
    : PRINT@25,PQ;
222 PRINT@T1, A$;
    : PRINT@T1-64, B$;
224 IF T1+32Ø <=96Ø THEN PRINT@T1+32Ø, A1$;
    : PRINT@T1+256, B$;
225 PRINT@96Ø, B$;
226 GOSUB 1ØØØØ
25Ø NEXT T1
26Ø NEXT T
27Ø IF AØ<>1 THEN IF SC>=A1 THEN CLS
    : PRINT CHR$(23); "EXTRA TIME";
    : FOR YU = 1 TO 15ØØ
    : NEXT YU
    : CLS
    : AØ=1
    : GOTO 19Ø
28Ø CLS
    : PRINT "HOPE YOU HAD A GOOD TIME"
    : PRINT "BYE!!!!"
    : PRINT "YOUR SCORE IS ="; SC+S1
    : END
1ØØØØ FOR W9=1 TO Q
    : X8=X9
    : PE=PEEK(144ØØ)
    : IF PE=32 THEN X9=X9-5 ELSE IF PE=64 THEN
    X9=X9+5
1ØØ1Ø IF X9>1Ø18 THEN X9=96Ø ELSE IF X9<96Ø THEN
    X9 = 1Ø18
1ØØ2Ø PRINT@X9, K$;
    : IF X8<>X9 THEN PRINT@X8, "5SPCS";
1ØØ21 PQ=PEEK(1536Ø+X9+2-64)
    : IF PQ=191 THEN SC=SC+7
    : PRINT@X9-64, "******"
1ØØ22 NEXT W9
1ØØ3Ø RETURN
3ØØØØ CLS
    : PRINT "THESE ARE THE INSTRUCTIONS FOR
    STUNT RACER"
3ØØ1Ø PRINT
    : PRINT "THE OBJECT OF THE GAME IS TO STEER
    YOUR CAR"
3ØØ2Ø PRINT "(WHICH IS AT THE BOTTOM OF THE
    SCREEN), INTO AS MANY WALLS AS POSSIBLE"
3ØØ3Ø PRINT "YOU CAN STEER THE CAR RIGHT WITH THE
    RIGHT ARROW AND"
3ØØ4Ø GOSUB 65ØØØ
3ØØ5Ø PRINT "YOU CAN STEER THE CAR LEFT WITH THE
    LEFT ARROW (NOTE: THERE IS A WRAP AROUND"
3ØØ6Ø PRINT "FEATURE WHICH ALLOWS YOU TO MOVE
    FROM THE LEFT SIDE OF THE SCREEN TO THE
    RIGHT SIDE AND VICE"
3ØØ63 PRINT "VERSA. THIS IS THE SECRET TO WINNING
    THIS GAME"
3ØØ64 PRINT "YOU WILL BE ASKED A SERIES OF
    QUESTIONS"
3ØØ7Ø PRINT "THE FIRST IS HOW LONG (IN TIME
    UNITS)"
3ØØ8Ø PRINT "THE GAME SHOULD BE. BETWEEN 25 AND
    1ØØØ TIME UNITS"
3ØØ9Ø PRINT "25 UNITS IS ABOUT 7 MINUTES LONG
    WHILE 1ØØØ UNITS IS "
3Ø1ØØ PRINT "ABOUT 1 HOUR LONG.  A NUMBER WILL
    THEN BE DISPLAYED"
3Ø11Ø PRINT "SHOWING HOW MANY POINTS YOU MUST
    ACHIEVE IN ORDER TO"
3Ø12Ø PRINT "RECEIVE A BONUS ROUND. (YOU CAN ONLY
    GET 1 BONUS ROUND)"
3Ø13Ø GOSUB 65ØØØ
3Ø14Ø PRINT "THE SECOND QUESTION YOU WILL BE
    ASKED IS ABOUT HOW MANY"
3Ø15Ø PRINT "UNITS LONG EACH WALL SHOULD BE
    BETWEEN 4 AND 8"
3Ø16Ø PRINT "8 UNITS IS THIS LONG ________ WHILE
    4 UNITS IS THIS LONG ____"
3Ø17Ø PRINT "THE THIRD QUESTION RELATES TO THE
    SKILL LEVEL"
3Ø18Ø PRINT "YOU WILL BE ASKED IF YOU WANT LEVEL
    1 OR 2.  LEVEL 1 IS"
3Ø19Ø PRINT "THE HARDER OF THE TWO AND YOUR CAR
    WILL MOVE SLOWER"
3Ø2ØØ PRINT "ON THIS LEVEL"
3Ø21Ø PRINT
    : PRINT "HOPE YOU HAVE FUN"
    : GOSUB 65ØØØ
    : CLS
    : RETURN
65ØØØ PRINT "HIT SPACE BAR TO CONTINUE"
65ØØ1 IF INKEY$<>CHR$(32) THEN 65ØØ1 ELSE CLS
    : RETURN
65ØØ2 END
```

AUTHOR'S NOTE: This program was originally written for a TRS-80 Model I or Model III with 16K of RAM but I think if you delete the instructions it might run on a 4K level 2 system instead of 16K level 2 system.

Quick Label

Roy W. MacLean
RADIO SHACK 1016
Capetown Shopping Mall
Route 132
Hyannis, MA 02601

I will share with you one of those programs that is easy enough to do, but some of us just cannot find the time for.

Have you ever had the need to reproduce labels on a repetitive basis, but need only a few? Here is a simple program that works on just about any printer. The variables used are pretty self-explanatory. Spaces enclosed in quotes are used in lines 1100-1120 to adjust the print head to center the label. In my line listing, one blank worked in the printer I used. In lines 1130-1150, I used blanks to force line feeds. You may have to adjust this to your own printer.

There may also come the time when you need just one label and you are not drawing it from any data base. If that is the case, then delete 1060-1090 and 1160. This can be called in very fast from disk and can be used to address letters in a simple word processing workstation setup.

```
1ØØØ CLEAR 2ØØ
1Ø1Ø INPUT "NAME "; NA$
1Ø2Ø IF NA$="ØØ" THEN END
1Ø3Ø INPUT "ADDRESS "; AD$
1Ø4Ø INPUT "CITY STATE ZIP "; CT$
1Ø5Ø CLS
1Ø6Ø INPUT "NUMBER OF LABELS "; L$
1Ø7Ø IF L$=CHR$(13) THE L=1 ELSE L=VAL(L$)
1Ø8Ø REM ***** START OF LOOP *****
1Ø9Ø FOR T=1 TO L
11ØØ LPRINT " "; NA$
1111Ø LPRINT " "; AD$
1112Ø LPRINT " "; CT$
1113Ø LPRINT " "
1114Ø LPRINT " "
1115Ø LPRINT " "
1116Ø NEXT T
1117Ø REM ***** END OF LOOP *****
1118Ø GOTO 1Ø1Ø
```

Essential Math Programs

Microcomputer-based basic skills instruction in mathematics isn't only for the elementary school students, although Radio Shack's K-8 Math Program has been well-received and well-publicized. Secondary students too can use the microcomputer to sharpen their mathematical skills. C.A.I. (computer assisted instruction) programs can help secondary students master addition, subtraction, multiplication, and division skills, as well as fractions, decimals, percent, and pre-algebra concepts. C.A.I. can help secondary students build the foundations they'll need for algebra and other high school and college math courses.

The Radio Shack Essential Math Program is Radio Shack's secondary school counterpart to the Radio Shack K-8 Math Program. The Essential Math Program, Volume One, is now available through your local Radio Shack Store or Computer Center. Volume Two will be available soon. Together, the two volumes provide a wide range of supplementary exercises designed to complement your local secondary school's math curriculum.

The Essential Math Program, Volume One, consists of eight computer programs. Each program contains a series of lessons. One program each is included for Addition, Subtraction, Multiplication, and Division. Lessons in each of these programs are sequenced in order of increasing difficulty, with each lesson building upon skills mastered in the preceding lesson.

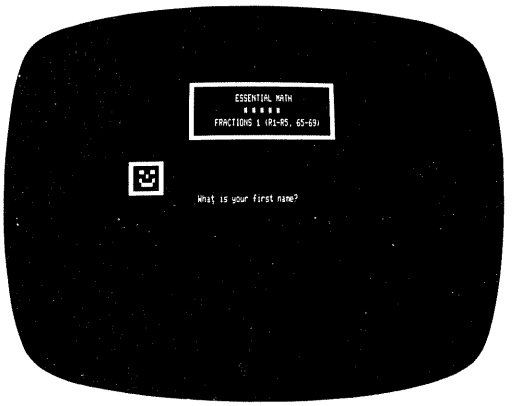
The other four programs in Volume One of the Essential Math Program provide "Number Concepts" lessons in a variety of skills. For example, Number Concepts 1, the first of the four programs, includes lessons in ordering numbers, identifying place value, rounding, using expanded notation, and finding common factors. Other Number Concepts programs cover topics including prime numbers, exponents, and square roots.

Volume Two of the Essential Math Program consists of a total of ten programs in three subject areas: Fractions, Decimals and Percent, and Pre-Algebra concepts. Again, lessons build upon one another. The first Fractions lesson asks the student to identify the fraction represented by a diagram on the screen. Later the student compares and reduces fractions and multiplies, divides, adds, and subtracts fractions and mixed numbers.

The Decimals and Percent lesson sequence begins with problems in adding, subtracting, and multiplying decimal numbers that represent money. Later the student writes decimals as fractions, determines place values in decimal numbers, and multiplies, divides, adds, and subtracts decimals. Percent exercises include converting fractions and decimals to percents, finding percentages of whole numbers, and finding particular numbers by knowing percentage relationships.

Lessons in the Pre-Algebra sequence relate familiar mathematics concepts to negative numbers and to equations containing variables.

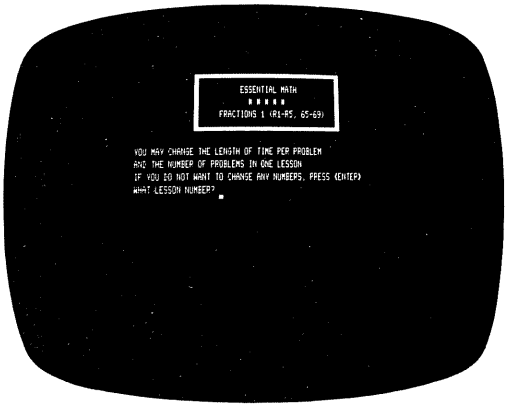
Now let's take a look at how you can use the two volumes of Essential Math. We'll use the Fractions 1 program from Volume Two as an example. As soon as the Fractions 1 program is loaded and running, you'll see the title screen appear:



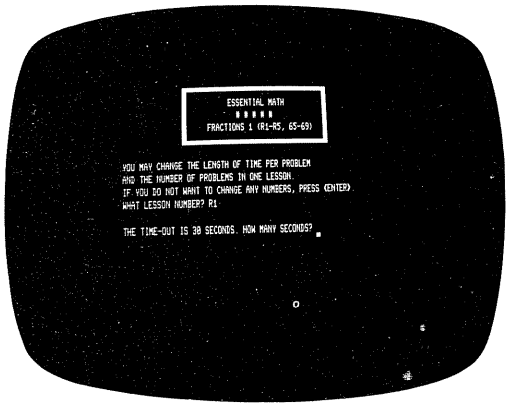
THE PLACEMENT MODE

Notice the question "What is your first name?" in the middle of the screen. If you typed the student's name and pressed ⟨**ENTER**⟩, you could start the student in the lesson of your choice. Alternatively, you could use the placement mode to determine where in the lesson sequence a student should begin. If you select the placement mode, the computer will measure the student's current level of ability and place him or her in the appropriate lesson for further practice.

To select the placement mode, you'd type ⟨**SHIFT**⟩ ⟨↑⟩ instead of a name at the question, "What is your first name?". You'll then see the question, "DO YOU WANT THE PLACEMENT MODE (Y/N)?". Type ⟨**Y**⟩ for "Yes" and press ⟨**ENTER**⟩. You'd then see the question, "DO YOU WANT AUTOMATIC PROMOTION AND DEMOTION (Y/N)?". You'd type ⟨**N**⟩ for "No" and press ⟨**ENTER**⟩. (When setting up for the placement mode, you do not need this feature.) Next the question "WHAT LESSON NUMBER?" appears:

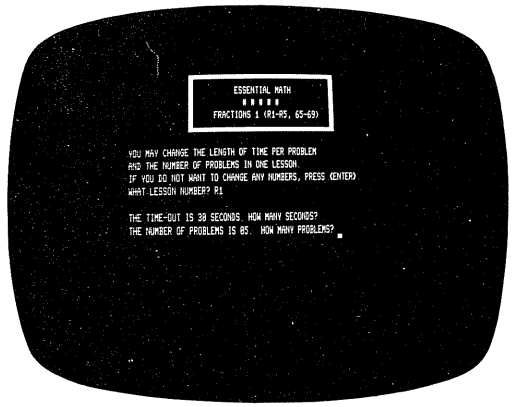


The Fractions 1 program contains ten lessons: review lessons R1 through R5, and lessons numbered 65 through 69. (The entire Fractions lesson sequence contains thirty-one lessons). If you don't know the student's ability in fractions, you can start him or her with the first lesson in a program in order to avoid the discouragement of too many demotions. (Lesson content summaries are included in each volume to help you make such decisions.) To choose lesson R1, you would type ⟨**R**⟩⟨**1**⟩ at this point, and press ⟨**ENTER**⟩. Then you would see the message, "THE TIME-OUT IS 30 SECONDS. HOW MANY SECONDS?":



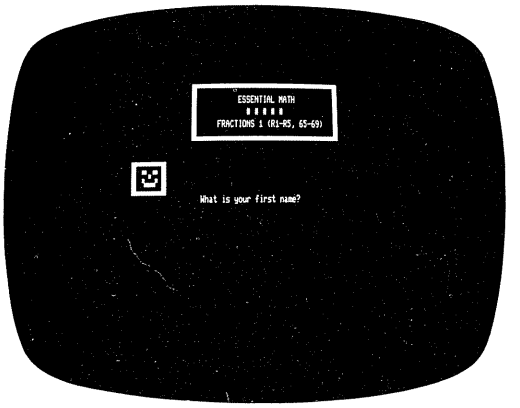
For lesson R1, the computer will normally allow the student 30 seconds to answer the problem before the word "TIME" is displayed on the screen. (The "TIME" message is used to encourage quick thinking, and is not considered in evaluating the student's response. The student can go on to answer the question after "TIME" is displayed.) If you wanted to change the number of seconds, you'd type in the new number of seconds and press ⟨**ENTER**⟩. If you decided to leave the time-out as it is, you'd simply press ⟨**ENTER**⟩.

Next you'd see the message, "THE NUMBER OF PROBLEMS IS 30. HOW MANY PROBLEMS?":

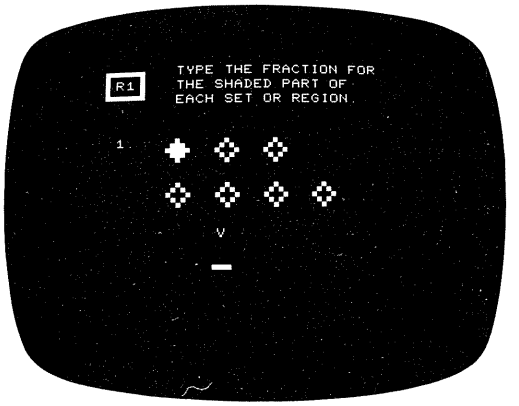


This message tells you that the computer will normally present 30 problems from lesson R1 when the lesson is run. If you wanted to change the number of problems, you could type in the desired number and press ⟨**ENTER**⟩. To leave the number of problems at 30, you'd simply press ⟨**ENTER**⟩. When you are setting up for the placement mode, it should be sufficient to have the student work just a few problems from each lesson. You might, for example, type ⟨**5**⟩ at this point and press ⟨**ENTER**⟩.

Once you have changed the number of problems, you are finished setting up for the placement mode and are ready for the student to begin working with the computer. At this point, the title screen reappears, with the question, "What is your first name?":

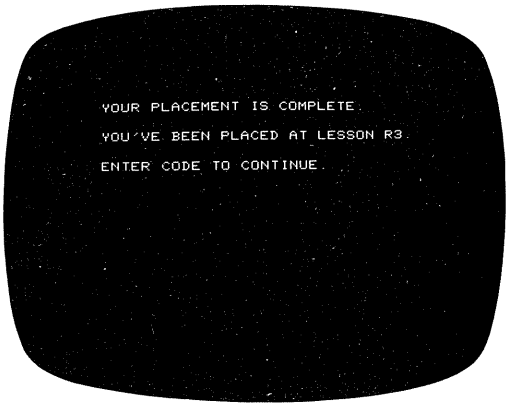


When the teacher or student types in the student's first name and presses ⟨**ENTER**⟩, the first problem appears on the screen. An almost infinite number of different problems can be presented in a given lesson, since the computer generates problems randomly according to a set of rules for each lesson's content:



To work this problem, the student types in the fraction that he or she thinks the diagram represents. First the numerator, then the denominator, is typed in. If the student answers correctly, a positive reinforcement message such as "GOOD JOB!" may appear, and a new problem will be presented. If the student enters an incorrect numerator or denominator, a "PLEASE TRY AGAIN" message is displayed. If the student twice enters an incorrect numerator or denominator, the correct answer is displayed, and the student must type the correct answer in order to continue. Once the entire problem is completed, the program moves on to the next problem.

In the placement mode if the student answers the five problems from lesson R1 with a score of 90% or above, the computer will present five problems from lesson R2. The computer will continue to promote the student until the student scores less that 90% on a set of problems from a given lesson. (If the student starts with a lesson other than the first lesson of the program and scores less than 70% on the first set of problems, the computer will demote the student until he or she scores above 70% on a set of problems from a given lesson.) Once placement is made, a placement report appears:

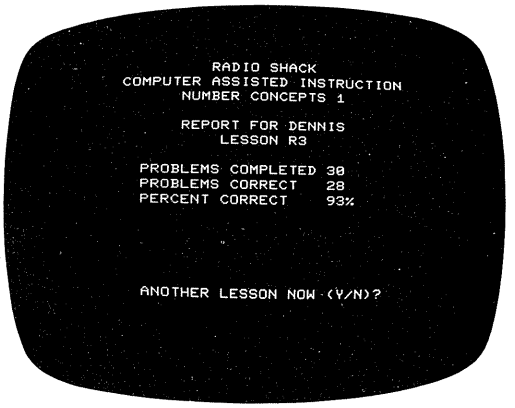


To allow the student to work skill-building problems in the lesson in which he or she is placed, simply press ⟨SHIFT⟩ ⟨S⟩ when a placement report is on the screen. Alternatively, you could press ⟨SHIFT⟩ ⟨T⟩ to end the program, and then type ⟨R⟩⟨U⟩⟨N⟩ ⟨ENTER⟩ to start the program over and place another student.

THE SKILL BUILDING MODE

If you choose to press ⟨SHIFT⟩ ⟨S⟩ when a placement report is on the screen, the computer will present a lesson at the level at which the student has been placed. The computer will present as many problems as are usually given for that lesson (remember that for lesson R1, the usual number is 30). If you want to change the number of problems in a skill building lesson, you can start the program over and change the number of problems per lesson when you see the question "HOW MANY PROBLEMS?".

After the student finishes a skill building lesson, a report similar to the following will appear on the screen:



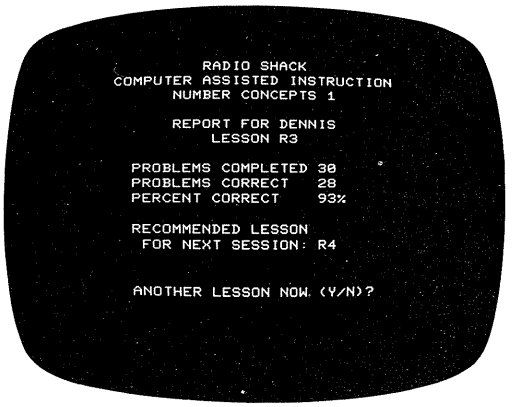
Information from this report can be recorded on a Student Record Sheet for the student. A sample student record sheet and blank student record sheets for you to photocopy are included with each volume of the Radio Shack Essential Math Program.

When the skill building report is being displayed, you can press ⟨Y⟩ to take another lesson, type ⟨N⟩ to return to the title screen, or type ⟨SHIFT⟩ ⟨T⟩ to end the program.

AUTOMATIC PROMOTION AND DEMOTION

The promotion and demotion feature is automatically in effect if you start a student in a skill building lesson without first viewing the question "DO YOU WANT AUTOMATIC PROMOTION AND DEMOTION (Y/N)?" and typing ⟨N⟩ for "No." (This feature is not in effect during the placement mode, but will be in effect if the student takes a skill building lesson after placement has been made.)

Automatic promotion and demotion allows a student to progress forward or backward in a lesson sequence after placement has been made. At the end of each skill building lesson, the computer recommends the next lesson based upon the student's performance on the lesson just taken. For example:



In this example, if the student presses ⟨Y⟩ to take another lesson the computer will present the recommended lesson (lesson R4). If automatic promotion and demotion had not been in effect, the student would simply repeat lesson R3 regardless of his or her performance on lesson R3. With automatic promotion and demotion, a student can work his or her way through all of the lessons in a program by moving forward one lesson each time he or she demonstrates mastery of the lesson just taken.

DEVELOPMENT OF THE ESSENTIAL MATH PROGRAM

The Radio Shack Essential Math Program, Volumes One and Two, grew out of a computer assisted instruction project that curriculum designer Dr. Max Jerman was working on at Stanford in the mid 1960s. Since then, the program has gone through many developmental stages, and has been updated in structure and content to fit the current basal mathematics textbooks.

Dr. Jerman has used the program to teach all types of secondary students, and he estimates that in development the program gave a total of one million lessons to ten thousand students. Dr. Jerman has also used the program for demonstration purposes in teaching math education classes at the college level, and he reports that a junior college in his area is using one version of the program in its remedial math classes.

The program takes advantage of the individualization of instruction and the immediacy that microcomputer-based instruction can provide. Through positive reinforcement messages, an immediate sense of achievement is given to the student when he or she does well. The student can see his or her progress through the lessons and may be motivated by the fact that these milestones are so tangible.

Immediacy also helps the student who is having trouble with lesson material. The program corrects mistakes immediately rather than allowing the student unknowingly to reinforce errors while completing an assignment and waiting for

feedback. Students are given corrections while they are still in the midst of problem-solving thought processes. Because classroom use of computer assisted instruction materials can save time for the teacher, the teacher can often give extra help to students who do not understand a lesson.

Finally, a program like Volumes One and Two of Essential Math allows almost every student to experience success. The automatic promotion and demotion feature will place most students in a lesson in which they can solve at least 70% of the problems correctly. Even if the student is not promoted immediately, he or she will see positive reinforcement messages for individual problems which are worked correctly. Through the intensive practice provided by each Essential Math lesson, most students will eventually score 90% or above on the lesson and will be promoted.

WHO USES THE RADIO SHACK ESSENTIAL MATH PROGRAM?

David Long, the administrator in charge of special projects at Mount Airy City Schools in Mount Airy, North Carolina, reports that his district has had success using both the Essential Math Program, Volume One, and the K-8 Math Program. This year the district set up three microcomputer centers, each containing a Radio Shack Network System with seven or eight student stations. Each one of the district's three school building complexes has its own center.

The center that serves the district's junior high and high school complex was used almost exclusively by Title I exceptional students throughout the first quarter of this school year. Students met in the center daily or on alternate days for math practice sessions with Essential Math and K-8 Math.

According to Long, the Title I students were able to make acceptable math grades during the time that they were working regularly with the computers. In the second term, however, the Title I students had less time to work with the math programs because other students were using the computers and because of other scheduling problems. Long reports that Title I math grades were down in the second semester, and he sees the lower grades as a result of less time spent with the math practice programs.

Long identifies the motivational approach of these Math practice programs as a strong point, and commented that the computer provided a "constant stimulus." Because students quickly got used to following the pointer symbol or cursor, they learned to "attack the problem in the correct manner." In this aspect, Long said, a computer program does more than a workbook, because the computer shows students how to solve the problem step by step.

The technicians who operate the district's three centers were hired by the district and then trained through the free computer courses for educators that are available at Radio Shack Computer Centers. Long also credits Gerald Moore, the educator at their local Radio Shack Computer Center in Winston-Salem, with being especially helpful. Long says, "Our main reason for going with Radio Shack was the service, which has been excellent."

Steve Drosdek, who teaches math and computer programming at Eagle Valley High School in Gypsum, Colorado, has recently started using Volume One of the Essential Math Program to reinforce students' basic math skills. Drosdek expressed concern that many students come to high school without having had the opportunity to sit down and work math exercises on paper with "enough repetition to master basic facts." He is very pleased with the Essential Math Program so far, and thinks it can help students "really develop basic skills" by "giving the practice that is necessary."

Drosdek has also used Essential Math's placement mode as a diagnostic tool to supplement the usual standardized tests. He has found the placement mode helpful in highlighting specific skills problems.

Students using Essential Math regularly take lessons in Eagle Valley's computer room, where they work intensively on problems from one Essential Math segment at a time—Addition, Subtraction, Multiplication, or Division. Drosdek says that students who do not enjoy doing math worksheets are motivated by the computer. And the response from some students who tend to be "nervous about new things" has included comments like "boy, that was fun!"

Eagle Valley's TRS-80 microcomputers are also used by Drosdek's programming students. One of Drosdek's objectives for the year is to extend microcomputer use to more classrooms, including using the SCRIPSIT™ word processing program in high school business classes.

HOW YOU CAN USE THE ESSENTIAL MATH PROGRAM

The Radio Shack Essential Math Program can be used in the classroom or in the home to build secondary math skills.

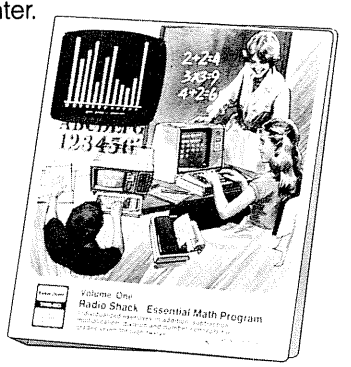
The program runs on a minimum TRS-80 Model III or Model I 16K tape or 32K disk system. Essential Math is also an ideal program for use with a Radio Shack Network 2 Controller.

Using the Controller, a teacher can load a program into a central disk-based TRS-80 Model III or Model I, and then send the program through the network to as many as sixteen cassette-based TRS-80s. This gives the classroom the increased efficiency of the disk system at low cost. The teacher can use the Network Controller to load one program into all of the student computers at once or to load different programs into different computers. The ability to load several or all of the student computers simultaneously can be a great time-saver.

The suggested price of the Essential Math Program, Volume One (catalog number 26-1716) is $199.00. Volume Two (catalog number 26-1719) is sold separately at $199.00. Prices may vary at individual stores and retailers.

Because the two volumes of Essential Math do not employ a one-lesson-per-program approach, you may be able to use a single program, once loaded, with a large number of students.

The Essential Math Program, Volume One, is available through any Radio Shack store or Computer Center. Volume Two will be available soon. Also, Radio Shack has Regional Educational Sales Coordinators throughout the country to help schools determine and meet their educational computing needs. For details, call your local Radio Shack store or Computer Center.



Line Printer VI Control Program

H. Richard Priesmeyer
2040 Lisa Lane
Fayetteville, AR 72701

It is a simple but useful adaptation of the software control commands for the Line Printer VI and allows you to easily select any of four character sizes and 3 pitch spacings. I have loaded it onto my SCRIPSIT disk and use it frequently when developing promotional materials.

One interesting note. Item number 4 on the menu creates characters which are not mentioned in the manuals. Try it and see. Use the test routine (item number 8) to review the different combinations of characters and pitch spacings available.

```
100 CLS
    : PRINT "LINE PRINTER CONTROL PROGRAM"
110 PRINT "LINEPRINTER VI"
112 PRINT "................................."
115 PRINT "CHARACTER SIZE OPTIONS"
120 PRINT "  (1) Expanded Characters"
130 PRINT "  (2) Condensed Characters"
140 PRINT "  (3) Normal Characters"
142 PRINT "  (4) Expanded/Condensed Characters"
    : PRINT
145 PRINT "LINE PITCH OPTIONS"
150 PRINT "  (5) 12 Line/inch pitch"
160 PRINT "  (6) 8 Line/inch pitch"
170 PRINT "  (7) 6 Line/inch pitch"
175 PRINT "  (8) TEST ROUTINE"
180 PRINT "................................."
190 INPUT "ENTER SELECTION"; A
200 ON A GOTO 300, 350, 400, 560, 450, 500, 550,
    600
300 LPRINT CHR$(27); CHR$(15); CHR$(31)
    : GOTO 100
350 LPRINT CHR$(30); CHR$(27); CHR$(14)
    : GOTO 100
400 LPRINT CHR$(30); CHR$(27); CHR$(15)
    : GOTO 100
450 LPRINT CHR$(27); CHR$(28)
    : GOTO 100
500 LPRINT CHR$(27); CHR$(56)
    : GOTO 100
550 LPRINT CHR$(27); CHR$(54)
    : GOTO 100
560 LPRINT CHR$(27); CHR$(15); CHR$(31)
    : LPRINT CHR$(27); CHR$(14)
    : GOTO 100
600 FOR N=1 TO 5
    : LPRINT "THIS IS a test."
    : NEXT
    : GOTO 100
```

Print Videotex Material

Jorge Mir
12851 W. Balboa Drive
New Berlin, WI 53151

I have outlined some simple procedures and written a short program to enable Color Computer users to print data gathered while using Videotex. You may want to publish these procedures and program in a future Newsletter.

VIDEOTEX/PRINT
By JORGE MIR

The procedures outlined below will enable you to get out of the Color Computer Tape Videotex program without having to turn the machine off. *(Editor's Comment: Videotex for the CC was released first on tape. We now supply the same program in a Program Pak. Mr. Mir's procedure is only applicable to the tape version.)* In addition, a short BASIC program is included to enable you to view the videotex pages stored in memory and select any of them for printing on a TRS-80 Line Printer VII.

First, you need to change the videotex program so that you can press the 'reset' button on back of the color computer to return to BASIC without turning the machine off.

Load the Videotex program the normal way (i.e., 'CLOADM'). Do not type 'EXEC' at this time. Instead, type the following:

```
FOR X=2102 TO 2110:POKE X,18:NEXT
```

The above procedure causes 'NO OP' codes to be inserted in the section of videotex that causes the machine to return to videotex after pressing the reset button. Now, the machine will return to BASIC instead.

At this point, you are now ready to run videotex. You start the program by typing 'EXEC'. When you have stored any information you want from the HOST computer, go 'offline' in the normal manner.

Once you are offline, you can print any of the pages still contained in memory (last 26 pages on a 16K system, last 4 if using a 4K system). You can search through memory by using the 'up' and 'down' arrows to view those pages which are still stored in memory.

If, after viewing those pages still in memory, you decide you want hard copies made, then press the reset button to return to BASIC. If you have Extended BASIC, you should type 'PCLEAR1' to enable you to load BASIC programs at the beginning of RAM. You should also disconnect the modem RS-232 cable and connect your printer to the serial port.

Once the machine is back to BASIC, you can load the 'V/PRINT' program shown at the end of these procedures. Load the program and type 'RUN'.

The program causes the pages stored in memory to be poked into the video screen, one at a time. To go to the next page, just press the **⟨SPACE BAR⟩** key.

Once you find a page which you wish printed, press the 'P' key and the printer will then print what is shown on the video screen.

Please note that if the memory area you are viewing does not contain any videotex pages, then the program will print some funny things on the screen. What you will see is pure garbage which is placed in memory when the computer is first turned on.

If you want to see what the programs stored in memory look like, then change line 20 in the program to 'FOR X=1536 TO 16383' (or 4095 for a 4K system).

You will not be able to go back to videotex at this point since, most likely, some of the pages stored in memory will have wiped out part of the videotex program. So, to go back to videotex, you must reload the program and follow the procedures outlined above to enable you to go back to BASIC when pressing the 'RESET' button.

Here is the program that enables you to print those videotex pages still stored in memory. (Continued on page 38)

Model II Hard Disk System

Using Radio Shack Application Programs With the HD System

I am getting numerous phone calls from the field about several key features on the 26-4150 Hard Disk. I will try to cover at least some of the questions.

First, what software will run or not run? Nearly all of the programs will run as they are. A few require minor changes to a BASIC module (3-4 line) or a patch to a machine language module.

A good example is Profile-II. Existing Profile data bases can be moved to the hard disk and will operate fine, the problem shows up when you try to create a NEW data set. The "DEFINE FILE" module asks which drive you wish to place the data on but will not accept "4" for an answer. A one byte patch fixes this.

Some of our programs have hard coded drive numbers for speed of data handling. Before these programs can be run, you will have to issue a "FLOPPY OFF" command from TRSDOS-HD. This tells the operating system to disregard any hard coded drive numbers and use the first drive that it finds containing the filename it needs or which is available for creating a new file. With a single hard disk, this would be drive 4.

Most of our programs fit into the above categories. The throughput of the programs will increase from 2 to 12 times and the capacities of the programs increase drastically. However, four programs will not change their capacities because they are using an internal memory index or a limiting counter set to a hard number. These programs are:

26-4501 General Ledger
26-4502 Inventory Management
26-4505 Accounts Payable
26-4507 Mailing List-II

Four programs will be re-released because of needing more than just a few patches to make them operate on both hard disk and floppy. These will be available shortly as a no-charge replacement for users that need to move the program to hard disk. There will be no enhancements made to the program other than making it operate on the hard disk. These programs are:

26-4710 Program Editor
26-4714 ReformaTTer
26-4715 Bisync 3270
26-4716 Bisync 3780

One program (26-4602 Inventory-Multi drive) will be released in 2 versions. The original version uses 4 floppies to handle up to 9000 inventory items, and a rework of the original program (no catalog number yet) to take full advantage of the hard disk would handle (I'm told) about 57,000 items if you wanted to dedicate most of the hard disk storage to it.

And SCRIPSIT 2.0??? A version will be released with a new catalog number as a hard disk only version. Much of the speed of SCRIPSIT 2.0 is due to the fact that all available memory was used including some that is needed by the hard disk system. The new version will have a few overlays that will be called in only as needed by the user. This should not cause a noticeable delay from hard disk but would have caused very noticeable delays if done from a floppy.

How about the SCRIPSIT dictionary? Well, as soon as SCRIPSIT HD is completed, we'll know. It should cause no problems. The only module that is known to need a change is INSTALL, and a patch should correct it. If a new version IS needed, it will be done. The Dictionary is a disk I/O intensive program and should really whiz on the hard disk system.

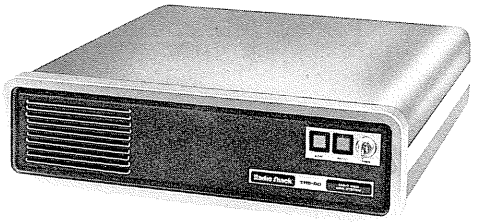
The next item I get a lot of questions about is Hard Disk BACKUP. Hard Disk BACKUP uses two new utilities that were added to the TRSDOS operating system called "SAVE" and "RESTORE".

SAVE writes to a floppy in a special (extended density) format and is much faster than the standard floppy backup. Standard floppy format takes about 3 minutes and backup from one floppy to another takes another 8½ minutes for a total of about 11½ minutes for 460K. The SAVE utility writes the track and data in one pass then verifies it. Each full disk takes only 2½ minutes to SAVE 660K. That amounts to 30 percent more data per disk and about ⅕ of the time. Saving 5 megabytes of data will only take eight floppy diskettes and 20 minutes.

The SAVE command can also be passed parameters that will cut down on the amount of data needing to be backed up. With the options, you may save only data that has been created after a certain date or data that has been updated since a certain date. Using the BUILD utility, you may create a file of filenames to be SAVEd. Since you also have full wildcarding, you can SAVE only filenames that begin with ABC or end with a particular extension like /BAS or /DAT.

RESTORE operates about the same way and with the same speed as SAVE. Even if you have SAVED an entire 5 to 6 megabytes of your system, you may only need to restore filenames with a specific extension or creation date. As each disk is created by SAVE, it is assigned a volume number and an ID. RESTORE will prompt you for the correct volume of the disk set needed and will verify that it is correct.

That's about it for now, by the time this is printed, you should be able to drop by any of the Radio Shack Computer Centers and enjoy HANDS ON experience with the hard disk unit.



Model II Bugs, Errors and Fixes

PROFILE II (26-4512)

It's been reported that Profile II (Version 1.0) doesn't work properly with a serial printer.

To correct this problem, follow these steps:

1. At TRSDOS READY apply the following patches to a backup copy of your Profile II diskette.

```
PATCH CLERK/EFC A=3411, F=3E64, C=3E7E
PATCH CLERK/EFC A=4F11, F=Ø664, C=Ø67E
PATCH CLERK/EFC A=4F2Ø, F=Ø665, C=Ø67D
PATCH CLERK/EFC A=56BC, F=Ø664, C=Ø67E
PATCH CLERK/EFC A=56C3, F=Ø665, C=Ø67D
PATCH CLERK/EFC A=5D66, F=ØEØØ, C=ØE42
```

* * * After applying these patches, you should be able to use a serial printer with Profile II.

SCRIPSIT VERSION 1.0 (26-4530)

To circumvent widow line handling during repagination SCRIPSIT version 1.0. Follow these steps:

1. Reset the system at swap diskettes.
2. Answer the Date Prompt, press ⟨**ENTER**⟩.
3. Press ⟨**HOLD**⟩ and then press ⟨**ENTER**⟩ for the Time Prompt.
4. Type in the following patch:

```
PATCH SCRIPSIT/6 A=E3B3 F=C5 C=C9
```

5. Type STARTUP and press ⟨**ENTER**⟩.
6. SCRIPSIT is now patched and you can continue operation.
7. Make a copy of the corrected diskette.

SCRIPSIT VERSION 2.0 (26-4531)

The initial release of 2.0 SCRIPSIT has the following known bugs:

1. Sheet feeder error never gets reset.
2. Cosmetic problem in user key execution of define text block.
3. Intermittent problem in D option in save/recall format line.
4. Exiting from a printer error in Merge hangs system.
5. The following patches will correct these errors.

```
PATCH SCRIPSIT A=D6FØ F=ØØØØØØØØØØØØØØØØ
                       C=AF32Ø3DB3AFDDAC9
PATCH SCRIPSIT A=BA84 F=C5Ø6ØØ C=C3EØD6
PATCH SCRIPSIT A=D6EØ F=ØØØØØØØØØØØØ C=D516ØØ3EØACF
PATCH SCRIPSIT A=D6E6 F=ØØØØØØØØØØØØØØ
                       C=D1C5Ø6ØØC387BA
PATCH SCRIPSIT A=D6F8 F=ØØØØØØØØØØØØØØØØ
                       C=ED73FBDACD15D9C9

PATCH SCRIPSIT/SYS R=155 B=223 F=3AFDDA C=CDFØD6
PATCH SCRIPSIT/SYS R=139 B=225 F=15D9 C=F8D6
PATCH SCRIPSIT/SYS R=154 B=124 F=3ØØ332D4DA
                                C=F5ØØØØØØØØ
PATCH SCRIPSIT/SYS R=154 B=132 F=3AD4DAB7 C=F1ØØØØØØ

PATCH SCRIPSIT/SYS R=158 B=45 F=AF C=ØØ

PATCH STARTUP A=EØB3 F=1Ø C=CØ
PATCH STARTUP A=E4ØE F=AAAAAAAAAAAAAAAAAAAA
                       C=B1BØAFB1B6AFB1B9B8B1
```

When these patches have been made, the SCRIPSIT initialization screen will show the date 10/16/1981 at the bottom. After the first 1000 or so, the disks had these patches made at the factory. So, if that date is visible, no patches need to be made.

It has been discovered that SCRIPSIT will not format text correctly when a large outline format is used.

For example:

```
--------o--------------------(-------)------------------------
```

The following patches will correct this problem:

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

NOTE: The patches listed just above must be applied prior to these patches.

```
PATCH SCRIPSIT A=9EF8 F=FD7571 C=22749Ø
PATCH SCRIPSIT A=9F5D F=7DFD9671 C=AFC3DØD6
PATCH SCRIPSIT A=9FBE F=ØØØØØØØØØØØØØØØØ
                       C=D5ED5B749ØED527D
PATCH SCRIPSIT A=D6D8 F=ØØØØØØØØØØØØ C=D1E1E5C3619F
PATCH STARTUP A=E4ØE F=B1BØAFB1B6AFB1B9B8B1
                       C=B1B1AFB1B1AFB1B9B8B1
```

2) When these patches are made the SCRIPSIT initialization screen will show the date 11/11/1981 at the bottom.
3) You are now ready to resume use of SCRIPSIT 2.0.

BI-SYNC. 3270 (26-4715)

The following patches correct the polling problem with the Bi-Sync Version 1.0 package. The problem is characterized by slow response to Model II input.

1. At TRSDOS READY apply these patches to a backup copy of 4715.

```
PATCH BIS327Ø A=DC15, F=ØØØØØØØØØØØØØØØØØØØØØØØØØØØØ,
                       C=3AB192FEØØC83ACD92C35D7A
PATCH BIS327Ø A=7A5A, F=3ACD92, C=C315DC
```

2. You are now ready to continue use of 4715.

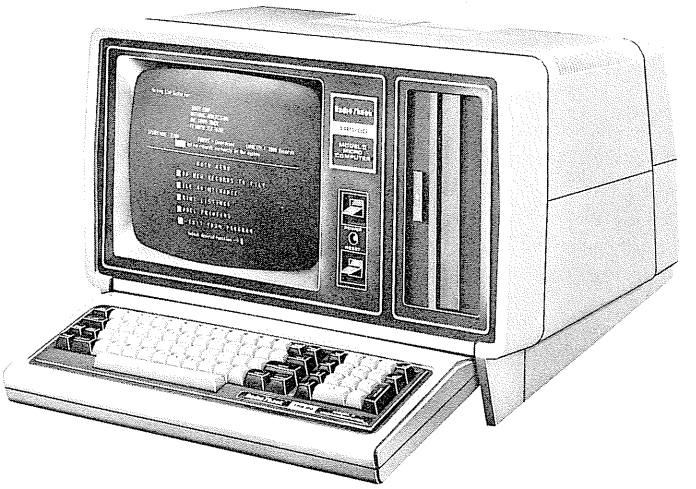
BI-SYNC. 3780 (26-4716)

The following patches correct the duplicate ACK problem with 26-4716 package. The problem is characterized by duplicate print lines, duplicate disk data, or high retransmission rates.

1. At TRSDOS READY apply the following patches to a backup copy of the program diskette.

```
PATCH BIS378Ø A=7CEE F=ØØØØØØØØØØØØØØØØØØØØØØØØØØØØ
                       C=AF324A813EØØ326481C3635Ø
PATCH BIS378Ø A=5Ø5E F=3EØØ326481 C=C3EE7CØØØØ
```

2. You can now continue to use BI-SYNC 3780 (4716).



Typewriter

David F. Salisbury
2991 Folsom Street
Boulder, CO 80302

I am a journalist by profession and use my Model II mainly for writing. There is absolutely no doubt that a microcomputer with a word processor like SCRIPSIT is as great an advance beyond the typewriter as the typewriter exceeded the pen and pencil. However, I have found there are certain cases - addressing a single envelope, for instance - where I have longed for the simplicity of slipping a piece of paper into a typewriter and punching away.

That feeling was the genesis for the following, exceedingly modest program. This transforms my Model II/Printer into an electric typewriter clone. It handles margins up to 255 characters and automatically returns the carriage following the last word of each line. It has proven perfect for tasks where full word processing is overkill.

If I have felt this way I imagine others have too. So I thought I would share it with you. It is listed on the second page of this letter.

By tying this into a simple diskette directory program, it becomes faster and much less cumbersome to use than it is to load SCRIPSIT, open a file, et cetera, et cetera.

Now my typewriter can truly rest in peace.

```
10Ø '*****TYPEWRITER MODE*****
2ØØ CLS
    : PRINT@112, "TYPEWRITER MODE"
3ØØ PRINT@272, "TO EXIT USE ^9"
4ØØ PRINT@346, "SET PRINTER MARGIN(Ø=255)";
    : INPUT A
5ØØ CLS
    : F=Ø
6ØØ IF A>79 THEN 16ØØ
7ØØ C$=INKEY$
8ØØ IF C$="" THEN 7ØØ
9ØØ IF C$=CHR$(92) THEN 2ØØ
1ØØØ IF POS(X)>=A THEN F=1
11ØØ IF F=1 THEN IF C$=CHR$(32) THEN C$=CHR$(13)
12ØØ IF C$=CHR$(13) THEN F=Ø
13ØØ PRINT C$;
14ØØ LPRINT C$;
15ØØ GOTO 7ØØ
16ØØ C$=INKEY$
17ØØ IF C$="" THEN 16ØØ
18ØØ IF C$=CHR$(92) THEN 2ØØ
19Ø IF ROW(X)/2<>INT(ROW(X)/2) AND POS(X)>=A-79
    THEN F=1
2ØØØ IF F=1 THEN IF C$=CHR$(32) THEN C$=CHR$(13)
21ØØ IF C$=CHR$(13) THEN F=Ø
22ØØ PRINT C$;
23ØØ LPRINT C$;
24ØØ GOTO 16ØØ
```

Garbage Collection

William L. Pierce
THE NATIONAL ALLIANCE
P.O. Box 3535
Washington D.C. 20007

I noted with interest your information on minimizing the "hang ups", during which the computer is cleaning up string space, on page 3 of the October issue.

We have developed our own BASIC programs for manipulating fairly large files of disk-based personnel records with our Model II, and one of the problems which formerly caused us a great deal of grief was the interminable "hang ups" or "coffee breaks" (sometimes as long as 30 minutes) which the computer took in going from one file to another. We learned several tricks for reducing the length of these coffee breaks somewhat, until we finally decided that the only way to eliminate them (almost) completely is to CLEAR and reDIMension all string arrays each time we load a new file into memory.

For example, a portion of the program line which sets up memory for our string arrays is:

```
CLEAR 9ØØØ: DIM R(5ØØ,6)
```

We found that while it helped a little to insert an:

```
ERASE R: DIM R(5ØØ,6)
```

command sequence before loading each new file into memory, by far the better procedure is to use:

```
CLEAR 9ØØØ:DIM R(5ØØ,6)
```

each time. (Of course, all other arrays must also be redimensioned after the CLEAR command.) This caused a few minor program complications, but it was not too difficult to work around them. The result of the change is that programs which formerly took 12 hours to work their way through 26 alphabetical files now take about 40 minutes.

When one is working with smaller arrays, the "hang up" problem is not so severe, but if one is writing a program which will be used with large arrays at some time in the future, it is wise to avoid the problem from the beginning. We would have saved ourselves a great deal of trouble if we had known how long and frequent the Model II's "coffee breaks" would be when our files grew in size.

BASICally Speaking . . .

The information for this month relates to some of the things that can be done with the BASIC resident in the Color Computer and some of the things you need to be careful about when using that BASIC.

As you probably know, the Color Computer includes a random number generator which enables you to create a series of random numbers. Unlike the Models I/III or II, the Color Computer does not have a command to randomize the random number "seed" (e.g. a RANDOM command.) What this means is that the first time you turn on the computer, load in your random number program (or a program that uses random numbers), the first and subsequent numbers, although appearing random, are the same set of numbers. Sound confusing? Maybe this little program will explain it better:

```
10 FOR X = 1 TO 5
20 A = RND(10)
30 PRINT A
40 NEXT X
```

Run the program. If you just turned the computer on, keyed in and ran the program, the sequence of numbers will probably be 5-4-2-7-5. If you run the program again, the sequence of numbers will be different. If you run this program, say, three times, write down the sequence of numbers you get, turn the machine off and start again from scratch, the three sequences of "random numbers" will be the same. A random number is not of very much use when it can be predicted. So . . . .try entering the following in your random number program:

If you have standard color BASIC or Extended BASIC, at the beginning of your program, (or somewhere that this routine will be executed prior to selecting random numbers, i.e.: a subroutine, GOTO, etc.) add a line like this to your program:

```
1 A$=INKEY$:IF A$="" THEN A=RND(0):GOTO 1
```

If you turn on (after you have turned it off) the Color Computer and enter the random number program including the program line #1, run it, and check the results, you will find what is accepted in the computer world as a "random sequence of numbers". Line #1 will force you to press ENTER (or any other key) to proceed with the program. All subsequent random numbers will be unpredictably random.

If you have Extended BASIC, another method of generating a "random seed" for the random number generator is as follows:

```
1 A = RND(TIMER)
```

Using this method under Extended BASIC will give you over sixty-five thousand different starting sequences possible (65536 to be exact). Once the starting sequence is randomized, all subsequent numbers will also be unpredictably random.

As you can see, using either one of these routines will give you vast starting sequences with the odds against duplicating the same sequence consistently being astronomical.

A couple of Color Computer owners have brought to our attention the following limitation for the Color Computer. When using the PRINT USING function in Extended BASIC along with the Exponential function, you are limited to exponents of nine or less. If you try to use an exponent of 10 or larger, the resulting answer will not be correct. My suggestion here is to not use PRINT USING when using exponents and if you must, limit them (exponents) to less than 10.

Some other users have commented that when using the PCLEAR command in their programs, they don't always work the way they should. First, let's see what is happening inside the machine. We'll use the YO-YO program from the Going Ahead with Extended BASIC manual. The program is entered after power-up. This would have us in PCLEAR 4, with four "pages" of memory reserved for graphics displays. As we begin to enter the program lines, the computer starts storing the program in "free" memory. After the program is entered, we run the program and it "crashes." Before we get angry, we try to run the program again. This time it works! This is what has happened. The computer stored the program in free memory. When the program is executed, the PCLEAR 8 command in line 10 is encountered. Since eight graphic "pages" require more memory than four graphic "pages," the computer moves the program to a location not used by the graphics memory grabbed by the Extended BASIC ROM. This is why the program worked properly the second time it was run. After the first attempt at running, the program was moved, so any additional attempts at running the program will be successful.

Based on this information, here are some things to watch out for and some things you might use to avoid any possible problems:

The major thing to avoid is the use of multiple PCLEAR statements inside your program. If you initialize the program with a PCLEAR 1, start storing variables, then switch to a PCLEAR 4, store some more variables, and finally switch to a PCLEAR 8, by the time you go into the graphics display, the machine could have lost track of where it is, and more important, where and what the variables are!

The best suggestion under this type of circumstance is to figure out the maximum number of graphics "pages" needed at any one time for the entire program and PCLEAR that many pages from the very start.

Even simpler than that is to use the PCLEAR statement BEFORE you run your program. Simply type in PCLEAR 8, then run the program WITHOUT the PCLEAR statement inside your program. When you are writing your program, use the PCLEAR statement prior to entering the program, so the program will be stored where it does not need to be moved in order to make room for additional graphics memory "pages". The same suggestion holds true for loading a program. Prior to loading the program, use the PCLEAR statement required for that program.

Does that make sense? Always keep in mind when

you're writing programs how the PCLEAR statement works and be careful with how you use it in your programming.

Now, on to a possibly obvious but still frequently asked question about the Color Computer and BOOKS. Got your attention, well read on. We at Radio Shack have been besieged by requests for books on the Color Computer containing games and other assorted programs and hints. Well, I can't say when, but we will have some in the "future." In the mean time, might I suggest that you try using some of the existing books available for the Model I. Many of the programs in these books are in BASIC and with a few modifications will probably work on the Color Computer. I wrote a solitaire card game for the Model I "many moons ago" and when the CC came out, I decided to convert it to run on the Color Computer. Naturally I had to change the screen print locations, respace some of the FOR-NEXT-STEP routines, delete (or otherwise sidestep) various PEEK and POKE locations, but after making my "Getting Started" and "Going Ahead" manuals dogeared, I finally got it to work right. (How do you think I found my solution for the "random seed" routine?)

Maybe one of these days I'll get up the nerve to put the program listing in the Newsletter, but don't hold your breath . . .

One last tidbit of information: For the early purchasers of our TRS-80 Videotex packages. If, when you purchased your package, for whatever machine, you did NOT receive a Dow Jones manual/ID number, drop by your local Shack, (it's best to go to the store where you bought the package) and pick up the Dow Jones upgrade package (stock number 700-2300). It is free to you when you show proof of purchase. The store may have to order you one, but the wait is worth it! Just like CompuServe, you get the first hour of non-prime time use free. Well, bye again and happy computing.

Print Videotex

(From page 33)

```
1Ø Y=2336
     : CLS(Ø)
2Ø FOR X=336Ø TO 16383
3Ø POKE X-Y, PEEK(X)
4Ø IF X-Y<>1535 THEN 9Ø
5Ø Y=Y+512
6Ø I$=INKEY$
     : IF I$="" THEN 6Ø
7Ø IF I$="P" THEN GOSUB 1ØØ
8Ø CLS(Ø)
9Ø NEXT X
     : END
1ØØ L=Ø
     : FOR P=1Ø24 TO 1535
11Ø A=PEEK(P)
12Ø IF A>9Ø THEN A=A-64
13Ø A$=A$+CHR$(A)
     : L=L+1
     : IF L<>32 THEN 16Ø
14Ø PRINT#-2,A$
15Ø L=Ø
     : A$=""
16Ø NEXT P
17Ø RETURN
```

Joystick Draw Routine

Jim Ebbert
1680 N. Page Drive
Deltona, FL 32725

I have found that my Color Computer can do more than I thought. It seems that every time I write a new program I find something new about the computer. I tried the POKEs for the color computer listed in the September, 1981 issue. They worked O.K. but most of my long programs use high res. graphics so I do not have much of a use for them.

I have included a short program that uses the right joystick to draw on the T.V. screen. You move the joystick and the dot moves. If you press the red button it draws a line from the center of the screen to where the dot was. When you press the button the next time the line will go from the position the dot was in the last time you pressed the button to your current position etc . . .

To fill in an area drawn on the screen, put the dot in the area and press P for paint. To change screens press 1 for screen 1 or 0 for screen 0.

That is it for that program, it is not complicated but it works.

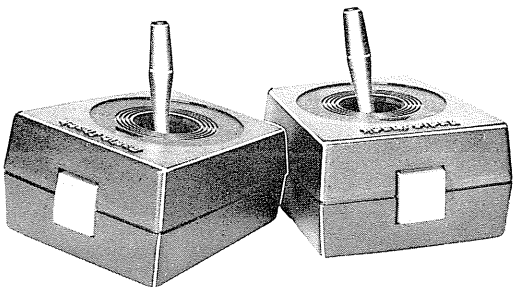
Here is the program:

```
1Ø DIM AC(1, 1)
2Ø PMODE 4, 1
     : PCLS
     : SCREEN 1, 1
3Ø A=JOYSTK(Ø)*4
     : B=JOYSTK(1)*3
4Ø GET(A, B)-(A, B), AC, G
     : PSET(A, B)
     : P=PEEK(6528Ø)
5Ø IF P=126 OR P=254 THEN LINE-(A, B), PSET
     : GOTO 9Ø
6Ø A$=INKEY$
     : IF A$="P" THEN PAINT(A+1, B+1), 5, 5
7Ø IF A$="1" THEN SCREEN 1, 1
8Ø IF A$="Ø" THEN SCREEN 1, Ø
9Ø PUT(A, B)-(A, B), AC, PSET
     : GOTO 3Ø
```

I also thought you might be interested in some good music so I included this song:

```
1Ø PLAY "T4 V31 L2 O2 G O3 L2. F L6 E D C O2 B L2
     B- L1 B- L2 G L2. O3 G L6 F E D C O2 L2 B L1
     B L4 B- L2. A L4 B L6 O3 C# D E F# G L2 A
     L1. B- O2 L2. B- O3 L4 C L6 D E- F G- A L2
     B- L1 B O2 L2 G O3 L2. F L6 E D C O2 B L2 B-
     L1 B- L4 A- L2 G L1 O3 G L6 F E D C O2 L2 B
     L1 B L4 B- L2 A L4 B L6 O3 C D E F E L2 G L4
     G L2 G L1 C L4 D D D L1..C"
```

Note: Eliminate the spaces when you enter this line.



3-D Color Graphics

Mark Granger

I have included two programs, both were written on the TRS-80 Extended Color Computer and both use Hi-Res graphics to the fullest.

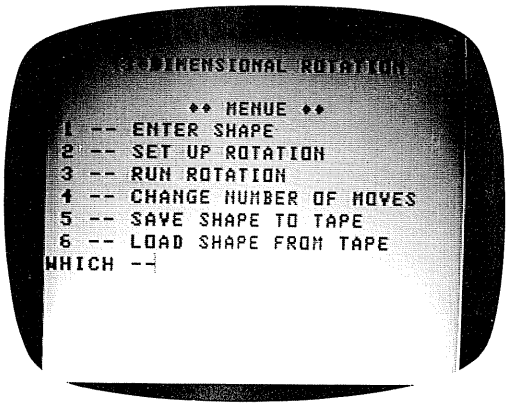
I have been interested in animation and 3-Dimensional graphics for a long time and I would like to share some of my ideas with you and the readers.

The first program is an all-purpose 3-Dimensional rotation program which I wrote from scratch on the Color Computer.

I have tried to make the program as easy to use as possible but it still requires instructions.

This is how to operate the program and create your own shapes:

First of all, the program does have its limitations. For example, do not use more that 15 lines because of memory limitations. Also, do not make more than 36 steps of rotation.



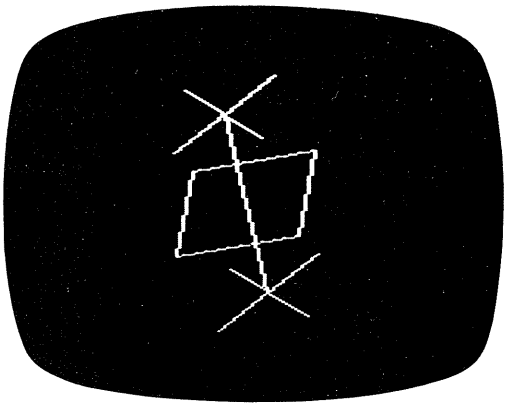
To enter a shape for rotation, press #1. The computer will then ask you how many lines you wish to enter and the x,y and z coordinates for the end points of each lines. Do not enter any number greater than 95 or less than -95. Generally numbers around 40 or -40 work best. A simple shape to enter is the 3 axis lines. Use the following data:

```
 50,    0,    0
-50,    0,    0
  0,   50,    0
  0,  -50,    0
  0,    0,   50
  0,    0,  -50
```

When you have finished entering the lines, the menu will reappear. The next thing you will need to do is to set up the rotation and compile the strings. Enter #2 and the computer will give you the prompt 'RX, RY, RZ'. Enter degrees for the shape to be rotated per step in each plane. (10, 10, 10 works best.) It will then ask you how many steps to move the shape. If you are unsure, type 36. For now, however, enter 21.

The method I used to generate "fast" motion is to 'compile' graphic strings and then draw them one after another while flipping the graphic pages creating constant motion. The end effect is almost as fast as machine code.

As the string is compiled, the shape will be plotted on the screen. This is the time when the program does the actual 'work' so it can take quite a long time to finish.



When it is done, type #3 to run the rotation. Using #4 you can adjust the number of moves.

When you decide you like what you see, you can save the shape to tape and load it again whenever you feel like it.

The second program is simple to use. It is just an old 3D plot formula which I wrote for the Color Computer. When you run it, type 'M' and a Hi-Res picture will be loaded off tape (if you have saved a copy! The first time through, you will have to use the 'P' option.) If you run the program and type 'P', you can see the program draw the picture. It takes about 2 hours to finish. After it is done, you can type

```
CSAVEM"SCREEN",1536,7679,Ø
```

to save the shape on tape for "instant" load later.

One last thing I would like to tell you is how to 'PCLEAR 0'. It is very simple. All you need to do is type the following when you turn on the computer:

```
POKE 25, 6
POKE 27, 6
POKE 29, 6
POKE 31, 6
```

Type 'PRINT MEM' and you will discover that you now have 14631 free bytes to play around with. I have been using this and several RAM cram techniques to fit several basic micro adventures and Star Trek onto my Color Computer. All seem to operate very well. This seems to work on ALL Extended Color Computers.

Program 1

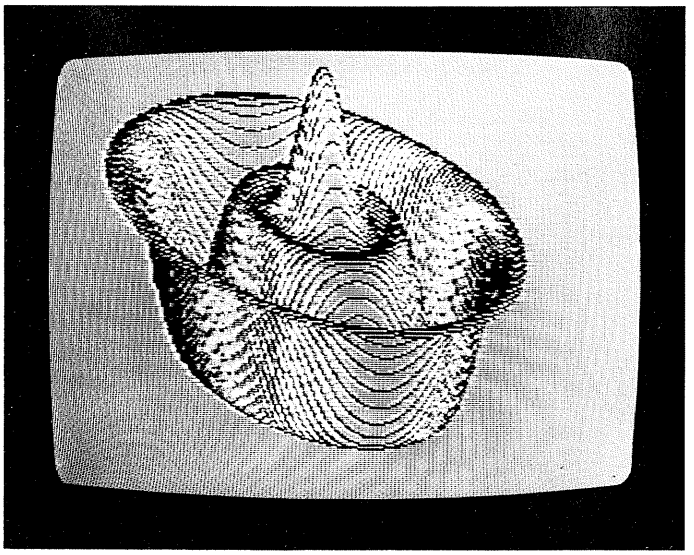
```
1Ø CLEAR 55ØØ
2Ø PI=3.1415927
3Ø T1=1
5Ø DIM P$(36)
55 GOTO 2ØØØ
56 CLEAR 55ØØ
57 DIM P$(36)
     : PI=3.1415927
     : T1=1
6Ø CLS
     : INPUT "NUMBER OF LINES"; L
7Ø DIM A(L,6)
8Ø FOR X=1 TO L
9Ø INPUT "X1,Y1,Z1"; X1, Y1, Z1
1ØØ INPUT "X2, Y2, Z2"; X2, Y2, Z2
11Ø A(X, 1)=X1
     : A(X, 2)=Y1
     : A(X, 3)=Z1
     : A(X, 4)=X2
     : A(X, 5)=Y2
     : A(X, 6)=Z2
```

```
12Ø NEXT X
125 GOTO 2ØØØ
13Ø CLS
     : INPUT "RX, RY, RZ"; RX, RY, RZ
     : INPUT "HOW MANY MOVES"; M
14Ø PMODE 4, 1
     : PCLS
     : SCREEN 1, 1
15Ø RX=RX*PI/18Ø
     : RY=RY*PI/18Ø
     : RZ=RZ*PI/18Ø
16Ø FOR I=1 TO 36
     : P$(I)=""
     : NEXT I
17Ø FOR I=1 TO M
18Ø P$(I)=""
19Ø FOR X=1 TO L
2ØØ N1=A(X, 1)
     : N2=A(X, 2)
     : T=RZ
21Ø GOSUB 1ØØØ
22Ø A(X, 1)=N1
     : A(X, 2)=N2
23Ø N1=A(X, 4)
     : N2=A(X, 5)
24Ø GOSUB 1ØØØ
25Ø A(X, 4)=N1
     : A(X, 5)=N2
26Ø N1=A(X, 1)
     : N2=A(X, 3)
     : T=RY
27Ø GOSUB 1ØØØ
28Ø A(X, 1)=N1
     : A(X, 3)=N2
29Ø N1=A(X, 4)
     : N2=A(X, 6)
3ØØ GOSUB 1ØØØ
31Ø A(X, 4)=N1
     : A(X, 6)=N2
32Ø N1=A(X, 3)
     : N2=A(X, 2)
     : T=RX
33Ø GOSUB 1ØØØ
34Ø A(X, 3)=N1
     : A(X, 2)=N2
35Ø N1=A(X, 6)
     : N2=A(X, 5)
36Ø GOSUB 1ØØØ
37Ø A(X, 6)=N1
     : A(X, 5)=N2
39Ø P1=INT(127+A(X, 1))
4ØØ P2=INT(96-(A(X, 2)))
41Ø P3=INT(127+A(X, 4))
42Ø P4=INT(96-(A(X, 5)))
43Ø P$(I)=P$(I)+"BM"+
     RIGHT$(STR$(P1),LEN(STR$(P1))-1)+ ","+
     RIGHT$(STR$(P2),LEN(STR$(P2))-1)+ "M"+
     RIGHT$(STR$(P3),LEN(STR$(P3))-1)+ ","+
     RIGHT$(STR$(P4), LEN(STR$(P4))-1)
44Ø NEXT X
45Ø DRAW P$(I)
     : NEXT I
47Ø GOTO 2ØØØ
48Ø CLS
     : INPUT "NUMBER OF MOVES"; M
49Ø FOR I=1 TO M
     : DRAW P$(I)
     : SCREEN 1, 1
5ØØ T1=-T1
     : IF T1=1 THEN PMODE 2, 1 ELSE PMODE 2, 3
51Ø IF INKEY$="" THEN PCLS
     : NEXT I
     : GOTO 49Ø
52Ø GOTO 2ØØØ
1ØØØ D=SQR(N1^2+N2^2)
1Ø1Ø A=ATN(N1/(N2+.ØØØØ1))
1Ø2Ø A=A+T
1Ø3Ø IF N2<Ø THEN A=A+PI
1Ø4Ø N1=D*SIN(A)
1Ø5Ø N2=D*COS(A)
1Ø6Ø RETURN
2ØØØ CLS
2Ø1Ø PRINT@ 37, "3-DIMENSIONAL ROTATION"
2Ø2Ø PRINT@ 1Ø6, "** MENU **"
2Ø3Ø PRINT " 1 -- ENTER SHAPE"
2Ø4Ø PRINT " 2 -- SET UP ROTATION"
2Ø5Ø PRINT " 3 -- RUN ROTATION
2Ø55 PRINT " 4 -- CHANGE NUMBER OF MOVES"
2Ø6Ø PRINT " 5 -- SAVE SHAPE TO TAPE"
2Ø7Ø PRINT " 6 -- LOAD SHAPE FROM TAPE"
2Ø8Ø LINEINPUT "WHICH --"; Q$
     : Q=VAL(Q$)
     : IF Q<1 OR Q>6 THEN 2ØØØ
2Ø9Ø ON Q GOTO 56, 13Ø, 49Ø, 48Ø, 3ØØØ, 4ØØØ
3ØØØ CLS
     : INPUT "FILE NAME"; F$
     : CLS
     : INPUT "SAVE -- HIT RETURN WHEN READY"; Q$
31Ø CLS
     : PRINT "NOW SAVING"
32Ø OPEN "O", -1, F$
33Ø PRINT#-1, L, M
34Ø FOR I=1 TO L
     : FOR J=1 TO 6
     : PRINT#-1, A(I, J)
     : NEXT J, I
3Ø45 IF M=Ø THEN 36Ø
35Ø FOR I=1 TO M
     : PRINT#-1, P$(I)
     : NEXT I
36Ø CLOSE
     : GOTO 2ØØØ
4ØØØ CLEAR 55ØØ
     : T1=1
     : PI=3.1415927
     : CLS
     : INPUT "FILE NAME"; F$
     : CLS
     : INPUT "LOAD -- HIT ENTER WHEN READY"; Q$
41Ø CLS
     : PRINT "NOW LOADING"
43Ø OPEN "I", -1, F$
44Ø INPUT#-1, L, M
45Ø DIM A(L, 6), P$(36)
46Ø FOR I=1 TO L
     : FOR J=1 TO 6
     : INPUT#-1, A(I, J)
     : NEXT J, I
4Ø65 IF M=Ø THEN 48Ø
47Ø FOR I=1 TO M
     : INPUT#-1, P$(I)
     : NEXT I
48Ø CLOSE
     : GOTO 2ØØØ
```

Program 2

```
1Ø CLS
     : INPUT "PROGRAM OR MFILE (P OR M)"; A$
     : IF A$="M" THEN 37Ø ELSE IF A$<>"P" THEN 1Ø
2Ø DIM M(256, 1)
3Ø PMODE 4, 1
     : PCLS 1
     : COLOR Ø, 1
     : SCREEN 1, 1
4Ø P=16Ø
     : Q=1ØØ
5Ø XP=144
     : XR=1.5*3.1415927
6Ø YP=56
     : YR=1
     : ZP=64
7Ø XF=XR/XP
     : YF=YP/YR
     : ZF=XR/ZP
```

```
8Ø FOR ZI=Q-1 TO -Q STEP -1.5
9Ø IF ZI<-ZP OR ZI>ZP THEN 17Ø
1ØØ ZT=ZI*XP/ZP
     : ZZ=ZI
11Ø XL=INT(.5+SQR(XP*XP-ZT*ZT))
12Ø FOR XI=-XL TO XL
13Ø XT=SQR(XI*XI+ZT*ZT)*XF
     : XX=XI
14Ø YY=(COS(XT)+.4*COS(3*XT))*YF
15Ø GOSUB 21Ø
16Ø NEXT XI
17Ø X2=Ø
     : Y2=Ø
18Ø NEXT ZI
19Ø POKE 65494, Ø
2ØØ GOTO 2ØØ
21Ø T=(YY-ZZ)*.9
22Ø I=(XX+ZZ+P)*.7
     : IF I>255 THEN RETURN
23Ø IF M(I, 1)=Ø AND M(I, Ø)=Ø THEN M(I, 1)=T
     : M(I, Ø)=T
24Ø IF T>=M(I, Ø) THEN M(I, Ø)=T
     : GOTO 27Ø
25Ø IF T<=M(I, 1) THEN M(I, 1)=T
     : GOTO 27Ø
26Ø X2=Ø
     : Y2=Ø
     : GOTO 36Ø
27Ø X1=(XX+ZZ+P)*.7
28Ø Y1=(YY-ZZ)*.9+Q+2Ø
29Ø '
3ØØ IF Y1=Ø THEN X2=Ø
     : Y2=Ø
     : GOTO 36Ø
31Ø IF X1>255 OR Y2>191 THEN X2=Ø
     : Y2=Ø
32Ø IF X2=Ø OR Y2=Ø THEN 35Ø
33Ø IF Y1<Ø OR Y1>191 THEN X2=Ø
     : Y2=Ø
     : GOTO 363
34Ø LINE(X2, 191-Y2)-(X1, 191-Y1), PSET
35Ø X2=X1
     : Y2=Y1
36Ø RETURN
37Ø PMODE 4, 1
     : PCLS
     : SCREEN 1, 1
     : CLOADM"SCREEN"
38Ø GOTO 38Ø
39Ø SCREEN 1, 1
     : GOTO 39Ø
```



Space Alert

J.W. Myers
Route 2, Box 29
Bastrop, LA 71220

Enclosed is a program "Space Alert" which I have been working on for the past two months. Complete documentation is included.

This program was written specifically for the TRS-80 4K Color Computer and I have made full use of color and sound in the program.

PROGRAM DOCUMENTATION

NAME OF PROGRAM:	SPACE ALERT
PROGRAMMER:	J.W.MYERS
DATE WRITTEN:	JULY 10, 1981
CURRENT VERSION:	SEPTEMBER 18, 1981

SPACE ALERT
By
J.W. MYERS

BACKGROUND:

Your spaceship is returning to Earth after a successful journey to Jupiter. On approaching Mars your radar indicates four Martian pirate spaceships are preparing to launch an attack. The "Space Alert" signal is activated and all hands rush to their "Battle Stations".

The number of attack missiles you have in your arsenal to defend against this attack is determined by the Level of Challenge selected. The four enemy ships are using destructive laser beams to try and destroy your ship. You have a firing advantage over your adversaries which is also determined by the Level of Challenge, but the laser weapons are more accurate.

If you have an enemy ship within firing range and fail to fire your missile within the time allotted then he is going to "ZAP" you. To fire your missile in a horizontal plane, press the (R) Key. To fire your missile in a vertical plane, press the (U) Key. To get a stop action picture of the situation, press the (Shift) Key and then the (@) Key. To restart the action, press any key.

Both you and the enemy must have the same horizontal or vertical coordinates before either of you can make a "hit".

Once you have fired a missile your radar automatically switches to a position that enables you to follow the path of the missile and see if you have made a "hit" or a "miss." If you record a "miss," a two second delay keeps the shot on the screen so you can see how far you were off target.

Your radar indicates how many "hits" and "misses" you make, so you can plan your strategy based on the number of missiles you have left.

Your spaceship shows up on the radar screen in an orange color while the enemy may be any color except orange.

You and your enemy are able to maneuver at the speed of light to change positions in outer space.

You win if you destroy all four enemy spaceships. They win if they destroy you or if you expend all of your missiles without destroying their four ships.

You have three (3) Levels of Challenge from which to choose:

Number 1 is for beginners—This level allows you four (4) seconds for firing the missile and gives you fourteen (14) missiles.

Number 2 is for amateurs — This allows you two (2) seconds for firing and seven (7) missiles.

Number 3 is for experts — This level allows you one (1) second for firing and five (5) missiles.

These levels can be changed as desired in lines 5000 through 5050.

THE PROGRAM'S OPERATION

Since this program is composed in BASIC, even a beginner should have no problem understanding its operation.

There are some unusual effects created in the program, so this is a brief description of how it works.

This program was written specifically for the 4K RAM TRS-80 Color Computer. Since this computer uses 1.657 K for color and sound basics, this leaves only 2.343 K for the programmer's use.

It was a real challenge to program the "Space Alert" game with the limited amount of memory. With all the shortcuts utilized, the program required exactly 2K which left .343 K to actually run the program.

Sound and color were used extensively throughout the program, to add excitement to the game.

VARIABLES USED IN THE PROGRAM

A1 = Games Won

A2 = Games Lost

(W,X,8) = (W,X) are the coordinates of your spaceships - (8) colors your spaceship orange

O = Number of enemy ships destroyed

If O = 1:Then one enemy ship is deleted from radar screen

If O = 2:Then two enemy ships are deleted from radar screen

If O = 3:Then three enemy ships are deleted from radar screen

If O = 4:You win the game

(R,S,7) = Enemy spaceship colored magenta

(P,Q,4) = Enemy spaceship colored red

(E,F,1) = Enemy spaceship colored green

(G,K,3) = Enemy spaceship colored blue

J = Hits scored

M = Misses

J + M = Number of missiles fired

C-(J + M) = Number of missiles in your arsenal

Y1 = Level of Challenge

Y2 = Time allowed for firing missile

F$ = INKEY$: Gives you the triggers for firing your missiles

"U" to fire vertically

"R" to fire horizontally

The (X$ = INKEY$) instruction in the program prevents wasting a missile should you inadvertently hit the F$ keys during other portions of the program.

When the spaceships appear on your radar screen, you have an alloted time, determined by the level of challenge, within which to determine a go or a no-go on firing a missile. The F$ = INKEY$ command is inserted in a position of the program so that the missile is fired instantly upon pulling the trigger, rather than waiting for the time to elapse. After the delay, if you have not fired a missile, the program moves through lines 300 through 310 to determine if you made the right decision. If your decision to not fire was correct, the program goes back to line 210 and the spaceships reappear in different positions on the radar screen.

If your decision was wrong then you go to line 2000 for your epitaph. (Note the special effects created by lines 2000 through 2030.)

* Lines 300 through 320 determine if you hit or missed the enemy ship.
* Lines 370 and 380 fire the horizontal shot.
* Lines 380 through 400 determine if you hit or missed the enemy ship.
* On hits you GOTO line 500 for your tally.
* On a hit one of the enemy ships disappears from the radar screen. This is accomplished in lines 5500, 5510 and 5520.

  You do not get credit for a double hit.

  After four hits you GOTO line 3000 for your just reward.
* On misses you GOTO 600 for a comment.

  If you run out of missiles you GOTO line 4000 for instructions.
* Lines 5540, 5585, 5635 and 5685 eliminate the possibility of an enemy ship and your ship having both coordinates the same.

Since this TRS-80 has 8 different colors I was able to create eye appealing backgrounds on the screen.

This TRS-80 also has 240 notes of sound — so sound effects are used throughout the program to keep your attention and to amuse you.

THE PROGRAM:

```
10Ø CLS
     : PRINT @ 135, "LEVEL OF CHALLENGE"
11Ø PRINT @ 2Ø2, "1. BEGINNER"
12Ø PRINT @ 266, "2. AMATEUR"
13Ø PRINT @ 33Ø, "3. EXPERT"
14Ø INPUT Y1
15Ø IF Y1<1 GOTO 13Ø
16Ø IF Y1>3 GOTO 13Ø
17Ø CLS(5)
     : PRINT @ 226, "GAMES: WON =" A1 "-- LOST
     ="; A2 ;
18Ø FOR Z=1 TO 25ØØ
     : NEXT
19Ø CLS(4)
     : PRINT @ 234, "SPACE ALERT ";
2ØØ FOR Z=1 TO 6Ø
     : SOUND 215, 1
     : X$=INKEY$
     : NEXT
21Ø CLS(Ø)
     : W=RND(61)
     : X=RND(31)
     : SET(W, X, 8)
22Ø SOUND 1Ø, 1
     : GOSUB 55ØØ
23Ø GOSUB 5ØØØ
24Ø FOR Z=1 TO Y2
25Ø SOUND 19Ø, 1
26Ø F$=INKEY$
```

```
27Ø IF F$="U" GOTO 33Ø
28Ø IF F$="R" GOTO 37Ø
29Ø NEXT
3ØØ IF W=E OR W=G OR W=P OR W=R GOTO 2ØØØ
31Ø IF X=F OR X=K OR X=Q OR X=S GOTO 2ØØØ
32Ø X$=INKEY$
     : GOTO 21Ø
     : NEXT
33Ø RESET(W, X)
     : SET(W, 31, 8)
     : FOR H=29 TO Ø STEP -2
34Ø SET(W, H, 5)
     : SOUND 195, 1
     : NEXT
35Ø IF W=E OR W=G OR W=P OR W=R GOTO 5ØØ
36Ø GOTO 6ØØ
37Ø RESET(W, X)
     : SET(Ø, X, 8)
     : FOR I=3 TO 61 STEP +2
38Ø SET(I, X, 5)
     : SOUND 195, 1
     : NEXT
39Ø IF X=F OR X=K OR X=Q OR X=S GOTO 5ØØ
4ØØ GOTO 6ØØ
5ØØ CLS(8)
     : SOUND 2ØØ, 1Ø
     : J=J+1
51Ø O=O+1
52Ø PRINT @ 168, "A DIRECT HIT! ";
53Ø GOSUB 7ØØ
54Ø GOTO 64Ø
55Ø END
6ØØ FOR Z=1 TO 1ØØØ
     : NEXT
61Ø CLS(7)
     : SOUND 17Ø, 1Ø
     : M=M+1
62Ø PRINT @ 168, "A CLEAN MISS! ";
63Ø GOSUB 7ØØ
64Ø FOR Z=1 TO 25ØØ
     : NEXT
     : IF O=4 GOTO 3ØØØ
65Ø IF J+M=C GOTO 4ØØØ
66Ø X$=INKEY$
     : GOTO 21Ø
7ØØ PRINT @ 26Ø, "HITS ="; J "  MISSES ="; M;
71Ø N=C-(M+J)
72Ø PRINT @ 359, "MISSILES LEFT ="; N;
     : RETURN
2ØØØ FOR Z=1 TO 5
     : CLS(8)
2Ø1Ø FOR H=Ø TO 61
     : SET(H, 15, 5)
     : NEXT
2Ø2Ø PRINT @ 224, "ZAP-SAP-ZAP-SAP-ZAP-
     SAP-ZAP-SAP"
2Ø3Ø SOUND 235, 3
     : NEXT
2Ø4Ø CLS(3)
2Ø5Ø PRINT @ 228, "YOUR SHIP WAS DESTROYED ";
2Ø6Ø FOR Z=1 TO 3ØØØ
     : NEXT
2Ø65 A2=A2+1
     : O=Ø
     : J=Ø
     : M=Ø
2Ø7Ø X$=INKEY$
     : GOTO 17Ø
3ØØØ FOR Z=1 TO 2Ø
3Ø1Ø CLS(4)
     : SOUND 5Ø, 1
     : CLS(5)
     : SOUND 1ØØ, 1
     : NEXT
3Ø2Ø CLS(2)
     : PRINT @ 226, "YOU HAVE DESTROYED THE
     ENEMY";
3Ø3Ø FOR Z=1 TO 4ØØØ
     : NEXT
3Ø4Ø A1=A1+1
     : O=Ø
     : J=Ø
     : M=Ø
     : GOTO 17Ø
4ØØØ CLS(8)
     : PRINT @ 227, "YOU ARE OUT OF AMMUNITION
4Ø1Ø A2=A2+1
     : O=Ø
     : J=Ø
     : M=Ø
4Ø2Ø FOR Z=1 TO 4ØØØ
     : NEXT
     : GOTO 17Ø
5ØØØ IF Y1=1 THEN Y2=3Ø
5Ø1Ø IF Y1=1 THEN C=14
5Ø2Ø IF Y1=2 THEN Y2=13
5Ø3Ø IF Y1=2 THEN C=7
5Ø4Ø IF Y1=3 THEN Y2=6
5Ø5Ø IF Y1=3 THEN C=5
5Ø6Ø RETURN
55ØØ IF O=1 GOTO 557Ø
551Ø IF O=2 GOTO 562Ø
552Ø IF O=3 GOTO 567Ø
553Ø R=RND(61)
     : S=RND(31)
554Ø IF W+S=X+R GOTO 553Ø
555Ø SET(R, S, 7)
     : SOUND 2ØØ, 1
     : GOTO 558Ø
557Ø R=Ø
     : S=Ø
558Ø P=RND(61)
     : Q=RND(31)
5585 IF W+Q=X+P GOTO 558Ø
559Ø SET(P, Q, 4)
     : SOUND 21Ø, 1
     : GOTO 563Ø
562Ø P=Ø
     : Q=Ø
563Ø E=RND(61)
     : F=RND(31)
5635 IF W+F=X+E GOTO 563Ø
564Ø SET(E, F, 1)
     : SOUND 22Ø, 1
     : GOTO 568Ø
567Ø E=Ø
     : F=Ø
568Ø G=RND(61)
     : K=RND(31)
5685 IF W+K=X+G GOTO 568Ø
569Ø SET(G, K, 3)
     : SOUND 23Ø, 1
57ØØ X$=INKEY$
     : RETURN
```

Bouncing Box

W. Tudor Apmadoc
Radio Shack Computer Center 7443

```
10 '
20 ' BOUNCING BOX PROGRAM
30 '
40 '   W. TUDOR APMADOC
50 '   RSCC 7443
60 '
70 PCLEAR 4
80 DIM X(200), Y(200)
90 DIM X1(200), Y1(200)
100 '
110 ' FIN = DENSITY OF BOXES
120 ' FO = # OF BOXES ON SCREEN
130 '
140 FIN = 10
150 FO = 20
160 XTP = 255
    : YTP = 191
170 A = RND(FIN)
    : B = RND(FIN)
180 C = RND(FIN)
    : D = RND(FIN)
190 POKE 65495, 0
200 PMODE 4, 1
210 PCLS 2
220 '
230 ' SET INITIAL POINTS
240 '
250 X = RND(XTP)
    : Y = RND(YTP)
260 X1= RND(XTP)
    : Y1= RND(YTP)
270 SCREEN 1, 1
280 T1 = 2
290 FOR G = 1 TO FO
300 '
310 ' CHECK IF EXCEEDS SCREEN
320 ' LIMITS, IF SO SELECT NEW
330 ' RANDOM INCREMENT
340 '
350 IF X > XTP-FIN THEN A = -RND(FIN)
360 IF X < FIN THEN A = RND(FIN)
370 IF X1 > XTP-FIN THEN B = -RND(FIN)
380 IF X1 < FIN THEN B = RND(FIN)
390 IF Y > YTP-FIN THEN C = -RND(FIN)
400 IF Y < FIN THEN C = RND(FIN)
410 IF Y1 > YTP-FIN THEN D = -RND(FIN)
420 IF Y1 < FIN THEN D = RND(FIN)
430 '
440 ' INCREMENT POSITION
450 '
460 X = X + A
    : X1 = X1 + B
470 Y = Y + C
    : Y1 = Y1 + D
480 '
490 ' DRAW BOX
500 '
510 LINE (X,Y)-(X1,Y1), PSET, B
520 IF FO = 1 THEN 650
530 '
540 ' STORE POINTS IN ARRAY
550 '
560 X(G) = X
    : Y(G) = Y
570 X1(G) = X1
    : Y1(G) = Y1
580 '
590 ' ERASE END LINE
600 '
610 LINE(X(T1),Y(T1)) - (X1(T1),Y1(T1)),PRESET,B
620 '
630 '
640 T1 = T1 + 1
    : IF T1 > FO THEN T1 = 1
650 NEXT G
660 GOTO 280
```

Biorhythms

Kenneth A. Mowen
158 Royal Palm Drive
Leesburg, FL 32748

The theory of BIORHYTHMS is not subscribed to by everyone. However, as there may be some substance to these observations, I am submitting this program for possible inclusion in the TRS-80 4K Color Computer section of your most interesting publication.

The theory of BIORHYTHMS states that there are three cycles to your life, which started the day you were born: The PHYSICAL cycle, which is 23 days long; the EMOTIONAL cycle, which is 28 days long; and the INTELLECTUAL cycle, which is 33 days long.

The first half of each cycle is said to be UP period and the last half the DOWN period.

When the cycle curve crosses the horizontal it is said to be a CRITICAL time.

This program for the 4K Color Computer will give you the number of days from the last critical period and tell you if the period is UP or DOWN.

You will need to know your age, the number of leap years since your birth and the number of days (including your birthday) since your last birthday. The charts below will assist you.

LEAP YEARS					DAYS IN THE MONTHS		
1904	08	12	16	20	Jan 31	Feb 28	Mar 31
1924	28	32	36	40	Apr 30	May 31	Jun 30
1944	48	52	56	60	Jul 31	Aug 31	Sep 30
1964	68	72	76	80	Oct 31	Nov 30	Dec 31

```
10 CLS
15 PRINT "BIORHYTHMS"
20 PRINT
25 PRINT "REFER TO THE INSTRUCTION"
30 PRINT "GUIDE BEFORE RUNNING
35 PRINT
40 PRINT "ENTER YOUR AGE"
    : INPUT A
45 PRINT
50 PRINT "HOW MANY LEAP YEARS"
55 PRINT "SINCE YOUR BIRTHDAY"
    : INPUT L
60 PRINT
65 PRINT "HOW MANY DAYS FROM YOUR"
70 PRINT "BIRTHDAY, INCLUDING YOUR
75 PRINT "BIRTHDAY, TO THE TARGET DATE"
80 INPUT D
85 CLS
90 N=(A*365)+L+D
    : PRINT "N="; N
100 X=N/11.5
    : Y=INT(X)
    : Z=X-Y
110 Z=Z*11.5
    : X=Z
    : X=X+.5
120 X=INT(X)
    : P=X
    : X=N/23
130 IF Z<.5 THEN P$="UP"
```

(Continued on page 46)

Programs for Your Enjoyment

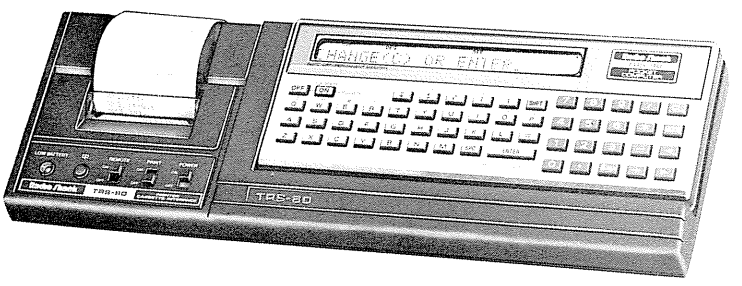
Star Trek Fight

Robert Saccone

I own two computers; the TRS-80 Model III with one disk drive and the Pocket Computer. I am submitting to you a program for the pocket computer that simulates a fight between the Star Trek Enterprise and a Random number of Klingons.

In the beginning of the program you are asked to enter two numbers from 1-100. It is best to enter a large number first then a smaller number next. Make sure there is a significant difference between the numbers. When you enter this program into the computer use all possible abbreviations because this program leaves about 4 steps 0 memories open. Have fun with it.

```
1: PAUSE "STAR TREK"
     : INPUT "ENTER 2 NUMBERS (1-1ØØ)"; N, M
     : GOSUB 5ØØ
     : PRINT "KLINGONS TO KILL"; K
2: E=Ø
     : W=1
     : S=1ØØ
     : P=2Ø
     : T=1Ø
     : D=15
     : V=36
     : FOR Q=1 TO 2
     : BEEP 1
     : PAUSE "RED ALERT !!"
     : NEXT Q
3: PAUSE "ENTER ONE"
     : PAUSE "1=STATUS REPORT"
     : PAUSE "2=FIRE PHOTON TORPEDOS"
4: PAUSE "3=FIRE PHASERS"
     : PAUSE "4=MOVE"
     : PAUSE "5=SELF DESTRUCT"
     : INPUT O
5: IF O=2 THEN 13
6: IF O=3 THEN 19
7: IF O=4 THEN 25
8: IF O=5 THEN 28
9: P=P*5
     : T=T*1Ø
     : PRINT "WARP IS: "; W
     : PRINT "SHIELDS ARE: "; S; "%"
     : PRINT "PHASERS ARE "; P; "%"
1Ø: PRINT "TORPEDOS ARE: "; T; "%"
     : PRINT "DIRECTION IS: "; D; "DEGS"
11: S=S-(O+V)
     : V=V+11
     : E=E+12.4*.3
     : T=T/1Ø
     : P=P/5
     : PRINT "KLINGONS AT: "; E; "KM."
     : IF S<=Ø THEN 8ØØ
12: GOTO 3
13: INPUT "ENTER FIRING ANGLE"; A
     : IF T<=Ø PRINT "OUT OF TORPEDOS"
     : GOTO 3
14: IF A>INT (((V+E)/2)) PRINT "SHOT WIDE! MISS!"
     : T=T-1
     : GOTO 3
15: IF A<INT (((V+E)/2)) PRINT "SHOT TOO LOW!"
     : T=T-1
     : GOTO 3
16: IF (A=INT(((V+E)/2))) +
    ((A+5)=INT(((V+E)/2))) +
    ((A-5)=INT(((V+E)/2))) PRINT "DIRECT HIT
17: PRINT "KLINGON DESTROYED!"
     : K=K-1
     : IF K<=Ø THEN 6ØØ
18: T=T-1
     : GOTO 3
19: INPUT "ENTER FIRING ANGLE"; L
     : IF P<=Ø PRINT "PHASER INOPERATIVE"
     : GOTO 3
2Ø: IF (L<>INT(((V+E)/2))) +
    ((L+5)<>INT(((V+E)/2))) PRINT "MISS!"
     : P=P-1
     : GOTO 3
22: PRINT "DIRECT HIT!"
     : K=K-1
     : PRINT "KLINGON DESTROYED!"
     : P=P-1
     : IF K<=Ø THEN 6ØØ
23: GOTO 3
25: INPUT "ENTER NEW ANGLE "; D
     : INPUT "ENTER NEW WARP"; W
26: PRINT "DIRECTION IS NOW: "; D
     : PRINT "WARP IS NOW"; W
27: GOTO 3
28: FOR Q=1 TO 2
     : PAUSE "SELF DESTRUCTION "
     : BEEP 1
     : NEXT Q
29: IF (D+15<E)+(D-15>E)*(K>Ø) PRINT "KLINGONS
    TOO FAR AWAY!"
     : GOTO 7ØØ
31: PRINT "DESTROYED ALL KLINGONS"
     : PRINT "BUT YOU DESTROYED"
     : PRINT "YOURSELF"
     : GOTO 7ØØ
5ØØ: K=ABS (INT (2*N/M/2))
     : RETURN
6ØØ: PAUSE "YOU HAVE DESTROYED THE"
     : PAUSE "KLINGONS AND HAVE BEEN"
     : PAUSE "PROMOTED TO"
6Ø1: PAUSE "ADMIRAL!"
     : INPUT "PLAY AGAIN?"; Y$
6Ø2: IF Y$="NO" END
6Ø3: GOTO 1
7ØØ: PRINT "YOU HAVE LOST"
     : PRINT "OR SELF DESTRUCTED"
     : INPUT "PLAY AGAIN? "; Y$
     : IF Y$="NO" END
7Ø1: GOTO 1
8ØØ: PRINT "SHIELDS=Ø"
     : GOTO 7ØØ
```



Printer Plot

Herbert G. Dorsey, III
321 East Ojai Avenue
Ojai, CA 93023

Here are some graphic programs for the TRS-80 hand held computer using the printer-cassette interface.

The first program was written for my girl friend who thought that I was paying too much attention to the computer and not enough to her. The dashes are spaces:

```
100: P."---XX-----XX"
101: P."-X----X-X----X"
102: P."X------X------X"
103: P."X----I-LOVE---X"
104: P."-X-----------X"
105: P."--X---YOU---X"
106: P."---X-------X"
107: P."-----X---X"
108: P."-------X"
109: END
```

Theoretically any kind of picture with 16 X N pixel resolution would be possible with this method. Different shades of black could be printed using different letters.

By using the print command as a subroutine I have come up with a method of plotting mathematical functions:

```
399: "GRAPH"
     : P."PLACE F(X) ON LINE 410"
400: INPUT "A=", A, "B=", B, "D=", D, "M=", M
401: X=A
     : I=INT A
     : C=I
     : GOSUB 918
402: GOSUB 410
403: Y=INT(6Y/M+.5)
404: IF C=0 GOTO 406
405: GOSUB(909+Y)
406: X=X+D
     : I=I+D
     : C=C+D
407: IF X>B GOTO 409
408: GOTO 402
409: END
410: Y=SIN X
411: IF C=5 LET C=0
415: IF C=0 GOSUB 917
416: RETURN
900: P."*"
     : RETURN
901: P."[]*"
     : RETURN
902: P."[][]*"
     : RETURN
903: P."[][][]*"
     : RETURN
904: P."[][][][]*"
     : RETURN
905: P."[][][][][]*"
     : RETURN
906: P."[][][][][][]*"
     : RETURN
907: P."   ETC.
915: P."[][][][][][][][][][][][][][][][]*"
     : RETURN
916: P."[]"
     : RETURN
917: P. USING "###";I;"+[][]+[][]+[][]+[][]+"
     : RETURN
918: P."[][]-1[]-.5[]0[][].5[]1"
     : RETURN
```

The function to be plotted is entered on Line 410 of the program. The input constants are A—the beginning value of plot, B—the end value, D—the plotting increment and M, the maximum value of Y. Also, M is the multiplying factor as the graph here goes from -1 to 1.

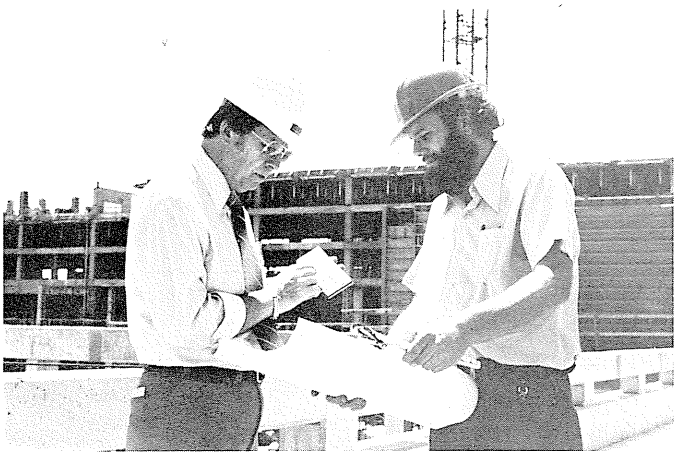
Singularities present problems in this program. For example Tan X goes to infinity at 90 deg. This causes an error signal to be displayed here. Some typical values of constants would be: For SIN X in radian mode: A = 0, B = 4pi, D = 0.5, M = 1; for ExpX: A = 0, B = 5, D = .5, M = 100, etc.

More plotting resolution could obtained for non-negative functions by altering program to start the plot with zero at the -1 position.

Well, somewhere I heard that the handheld TRS-80 did not have graphic capability. I just wanted to point out that this is not entirely true.

Biorhythms (From page 44)

```
140 IF Z>.5 THEN P$="DOWN"
150 X=N/14
     : Y=INT(X)
     : Z=X-Y
160 Z=Z*14
     : X=Z
     : X=X+.5
170 X=INT(X)
     : E=X
     : X=N/28
180 IF Z<.5 THEN E$="UP"
190 IF Z>.5 THEN E$="DOWN"
200 X=N/16.5
     : Y=INT(X)
     : Z=X-Y
210 Z=Z*16.5
     : X=Z
     : X=X+.5
220 X=INT(X)
     : I=X
     : X=N/33
230 IF Z<.5 THEN I$="UP"
240 IF Z>.5 THEN I$="DOWN"
250 PRINT P; "DAYS AFTER THE START OF THE"
260 PRINT "11.5 DAY "; P$; " PERIOD OF THE"
270 PRINT "PHYSICAL CYCLE"
280 PRINT
290 PRINT E; "DAYS AFTER THE START OF THE"
300 PRINT "14 DAY "; E$; " PERIOD OF THE"
310 PRINT "EMOTIONAL CYCLE"
320 PRINT
330 PRINT I; "DAYS AFTER THE START OF THE"
340 PRINT "16.5 DAY "; I$; " PERIOD OF THE"
350 PRINT "INTELLECTUAL CYCLE"
360 END
```



POCKET COMPUTER PROGRAM

BUSINESS STATISTICS

POCKET COMPUTER PROGRAM

CIVIL ENGINEERING

POCKET COMPUTER PROGRAM

REAL ESTATE

POCKET COMPUTER PROGRAM

SURVEYING

POCKET COMPUTER PROGRAM

PERSONAL FINANCE

POCKET COMPUTER PROGRAM

GAMES

POCKET COMPUTER PROGRAM

Electrical Engineering I

NEW LOW PRICE..$
169.95 !

LOW BATTERY REMOTE PRINT POWER

Radio Shack TRS-80 CASSETTE INTERFACE

Radio Shack TRS-80 POCKET COMPUTER

....ONLY $169.95 !!...

Radio Shack TRS-80 CASSETTE INTERFACE

Radio Shack

One Dollar and Ninety-Five Cents
Cat. No. 62-2086

TRS-80 POCKET COMPUTER PROGRAMS

By Jim Cole

50 ready-to-run programs in Pocket-BASIC, including:

BUSINESS Hourly Wages/Overtime • Pocket Datebook • Unit Cost • Commissions • Advertising Cost-per-1000 •

EDUCATION Math Drill • Metric Conversion • Averages • Temperature Conversion • Slope & Distance • Factors •

HOME Compound Interest • Checking Account Balance • Simple Interest • Mortgage Loan • Whodunit? • Craps • Old West Shootout • Klingon Killer • And many more!

Radio Shack

Nine Dollars and Ninety-Five Cents
Cat. No. 62-2512

PROBLEM-SOLVING ON THE TRS-80 POCKET COMPUTER

By Don Inman and Jim Conlan

Radio Shack Computer Center Addresses and Telephone Numbers

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

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

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

CALIFORNIA
ANAHEIM 509 Katella, (714) 776-9540
BERKELEY 1922 Grove St., (415) 848-9170
BEVERLY HILLS 8500 Wilshire Blvd., (213) 659-8870
CANOGA PARK 8371 Topanga Canyon, (213) 347-9800
CARMICHAEL 6305 Fairoaks Blvd., (916) 484-6815
ESCONDIDO 347 W. Mission Ave., (714) 741-6032
FRESNO Princeton S/C, 2721 N. Blackstone Ave., (209) 225-5551
GLENDALE 236 N. Brand Blvd., (213) 246-9310
HAYWARD 20942 Mission Blvd., (415) 278-2888
LAKEWOOD 5830 Lakewood Blvd., (213) 920-9671
LA MESA 5346 Jackson Dr., (714) 460-3610
LONG BEACH 2119 Bellflower Blvd., (213) 597-3377
MONTEREY 484 Washington St., (408) 375-8430
MOUNTAIN VIEW 1933 El Camino Real W., (415) 961-0542
PASADENA 575 S. Lake Ave., (213) 449-5424
RIVERSIDE 3844 La Sierra Ave., (714) 689-0340
SACRAMENTO 4749 J. St., (916) 454-3287
SAN DIEGO 3062 Clairemont Dr., (714) 276-6050
SAN FRANCISCO One Market Place, (415) 777-9810
SAN JOSE 1228 S. Bascom Ave., (408) 297-2603
SAN MATEO 3180 Campus Dr., (415) 573-8607
SANTA BARBARA 4141 State St. A-1, (805) 967-4538
STOCKTON College Sq. S/C, 963 West March Lane, (209) 957-3676
VENTURA 4005 E. Main St., (805) 654-0196
WEST COVINA 2516 E. Workman St., (213) 915-5791

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

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

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

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

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

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

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

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

INDIANA
FT. WAYNE 747 Northcrest S/C, (219) 482-9547
GRIFFITH 208 W. Ridge Rd., (219) 838-3000
INDIANAPOLIS 6242 E. 82nd St., Castleton Plz., (317) 849-6896; Speedway Plaza, 6129 B Crawfordsville, (317) 244-2221

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

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

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

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

MAINE
BANGOR Maine Square, (207) 945-6491

MARYLAND
BALTIMORE 7942 Belair Rd., Putty Hill Plaza, (301) 882-9583
CATONSVILLE One Mile West S/C, 6600 B Balt. Nat'l. Pike, (301) 788-3277
ROCKVILLE Congressional Plaza, 1673 Rockville Pike, (301) 984-0424
SALISBURY Shoppers World S/C, Rt. 50, (301) 546-9223

MASSACHUSETTS
BOSTON 730 Commonwealth Ave., (617) 739-1704
BROCKTON 675 Belmont, (617) 583-2270
CAMBRIDGE Harvard Square, 28 Boylston St., (617) 354-7694
CHESTNUT HILL 200 Boylston St., (617) 969-2031
NATICK 1400 Worcester Rd., (617) 875-8721
SAUGUS 343 Broadway, (617) 233-4985
SPRINGFIELD 1985 Main St., Northgate Plz., (413) 732-4745
WORCESTER Lincoln Plaza, (617) 852-8844

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

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

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

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

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

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

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

NEW JERSEY
E. BRUNSWICK 595 A Rt. 18, (201) 238-7142
E. HANOVER Rt. 10, Hanover Plaza, (201) 884-1200
LAWRENCEVILLE Rt. 1 & Texas Ave., (609) 771-8113
PARAMUS 175 Rt. 17 S., (201) 262-1920
SPRINGFIELD Rt. #22 Center Isle, (201) 467-9827

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

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

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

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

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

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

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

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

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

TENNESSEE
CHATTANOOGA 636 Northgate Mall, (615) 870-1366
JOHNSON CITY Peerless Center, (615) 282-6829
KNOXVILLE Cedar Bluff S/C, 9123 Executive Park Dr., (615) 690-0520
MEMPHIS 4665 American Way, (901) 795-4963; 1997 Union Ave., (901) 272-3055
NASHVILLE 2115 Franklin Pike, (615) 298-5484; Rivergate Plaza, (615) 859-3414

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

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

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

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

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

WISCONSIN
MADISON 57 West Towne Mall, (608) 833-6130
MILWAUKEE 6450 N. 76th St., (414) 353-6790
WEST ALLIS 2717 South 108th St., (414) 327-4240

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