Volume 4, Issue 3 **MARCH, 1982**

Price $1.50

TRS-80® Microcomputer News

Information Published for TRS-80 users.

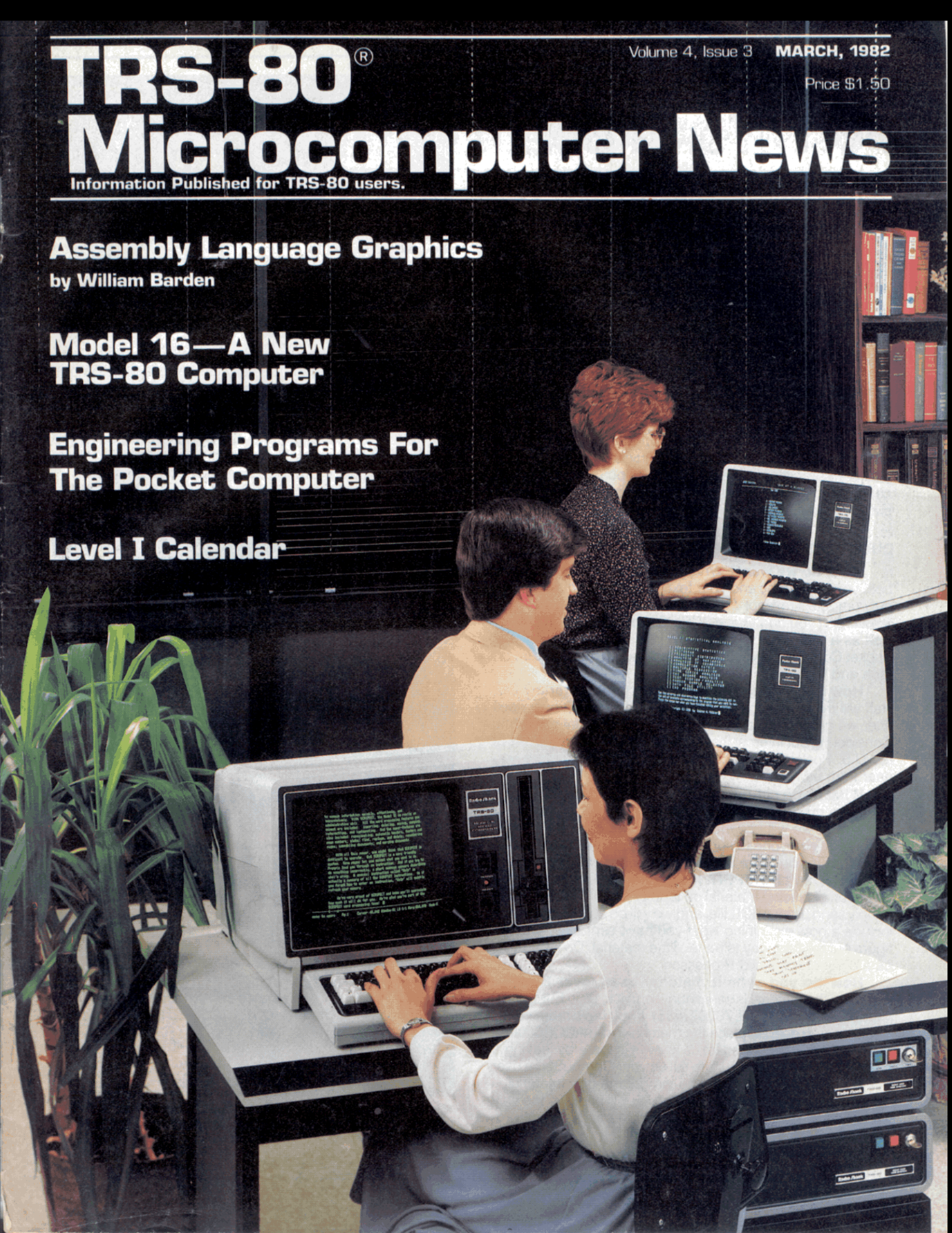
Assembly Language Graphics

by William Barden

Model 16—A New TRS-80 Computer

Engineering Programs For The Pocket Computer

Level I Calendar



Fort Worth Scene

A LITTLE FANFARE, PLEASE

Beginning the first of February, 1982 we will be rewarding those of you who submit articles on cassette tapes or diskettes (5¼" or 8" as appropriate) of programs and/or text material. We are also classifying CompuServe as a magnetic media!

If you submit material on magnetic media, and it is accepted for publication (which means we think the material MAY be used at some point in the future), we will send you two cassettes or two diskettes for each one you sent us. If we send you cassettes, they will come from our box of mixed blank cassettes. If you submit material on CompuServe which we think we may use, we will extend your Microcomputer News subscription by six months for each article accepted. Note: If you are submitting material over CompuServe, please include your name and address (including zip code) or your subscription number so we can find you.

QUESTIONS AND ANSWERS

What do we want? What we want is for you to submit both your programs and your text on magnetic media. All of our computers (except the Pocket Computer) have SCRIPSIT available. If you don't have SCRIPSIT, put the text on tape or disk as a data file and give us a little routine to retrieve it. Pocket Computer owners—please send us the program on tape and the article on paper.

How will you know if we accept your material? Our staff will make reasonable efforts to decide within one week after we get the material whether or not we may publish it. Considering our internal mail structure and the promptness of the U.S. Postal System, it may be eight weeks before you hear from us.

What if you submit material, and we accept it, but it is not on magnetic media? In that case, you will have our thanks and we will use the material as we can. We want to encourage everyone to submit material on magnetic media.

What if you have a word processor that you could use, but it isn't SCRIPSIT. Most word processors give you an option of placing material on tape or disk in ASCII format. Simply send us an ASCII file, and we will be able to read it. Do include a note explaining what you have sent us so we can figure out the best way to retrieve the material. Do NOT send us a copy of the Word Processor!

How do you submit material to us over CompuServe? If the material is very short, send it to us in E-Mail (our CompuServe number is 70007, 535). If you have more than a few lines, you need to place the material in the ACCESS area of CompuServe and then let us know it is there by leaving a message on E-Mail (be sure to tell us your CompuServe user number and the file name(s) so we get the right material.

We are hoping that this new policy will improve the way we handle material, and also improve the accuracy of programs and text in the News.

As always, correspondence about your subscriptions should be directed to:

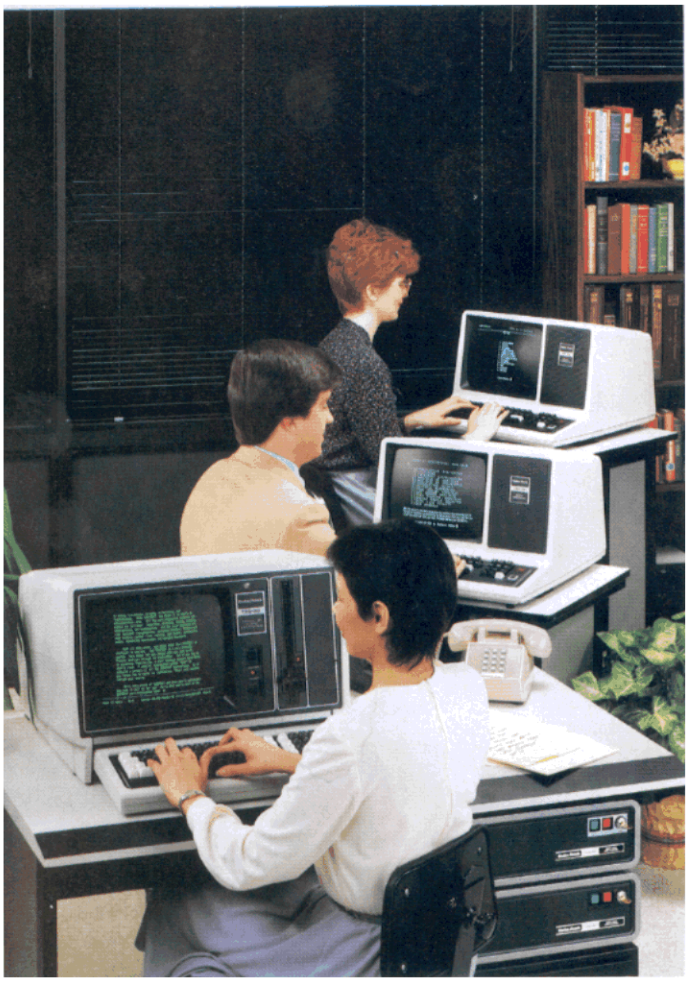
Microcomputer News
P.O. Box 2910
Fort Worth, Texas 76113-2910

TWO LAST-SECOND ITEMS

First, you may have noticed that our CompuServe user ID number has changed. The new number is 70007,535. We are assured that E-Mail to the old number will still reach us, but please change your records and use the new number.

Second, the 3-D Color Graphics program in the February, 1982 Microcomputer News was also published in the December, 1981 issue of Chromasette Magazine. Our thanks to the author, Mark Granger, for an excellent program.

Bruce Elliott, Editor
Linda Miller, Writer



The TRS-80 Model 16 and two of the TRS-80 DT-1 Data Terminals are pictured on the front cover. Information on these two new products can be found in this issue.

TRS-80® Microcomputer News

Volume 4, Issue 3 **MARCH, 1982**

Price $1.50

Information Published for TRS-80 users.

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

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

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

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

The TRS-80 Newsletter welcomes the receipt of computer programs, or other material which you would like to make available to users of TRS-80 Microcomputer systems. In order for us to reprint your submission, you must specifically request that your material be considered for reprinting in the newsletter and provide no notice that you retain copyrights or other exclusive rights to 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 authors' 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 enhance readability, and we separate each program statement onto a separate line. While these techniques increase program readability, they also require more memory, and may execute more slowly 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™	CompuServe, Inc.
DIF™	Software Arts, Inc.
Dow Jones NEWS/RETRIEVAL Service®	Dow Jones & Co., Inc.
Personnel Manager™	Image Producers
Project Manager™	Image Producers
ReformaTTer™	Microtech Exports
Time Manager™	Image Producers
VisiCalc®	VisiCorp™
Program Pak™	Tandy Corporation
SCRIPSIT™	Tandy Corporation
TRSDOS™	Tandy Corporation
TRS-80®	Tandy Corporation

Contents:

View From the Seventh Floor

by Jon Shirley,
Vice President, Radio Shack Computer Merchandising

What's new for March? A brand new computer catalog expanded to 56 pages to hold all the new products! This full color catalog (RSC-7) should be available in your nearest Radio Shack within two weeks. There is something new for every one of you TRS-80 owners no matter what model you have, plus there are two new computers and a terminal. Let's take a quick look at the new goodies.

Model I Owners. Well, gang, we know we are late but the new catalog has a Radio Shack Model I Double Density kit for you disk system owners. It's only $149.95 plus installation. It includes a new double density TRSDOS that has most of the Model III additions and can copy to and from single density diskettes. It allows 40 or 35 track operation. It also allows you to boot up and run any single density disks you have on any operating system.

If you are a Model I owner and do not have a disk system this is the last catalog that will list the expansion interface. It is out of production. There are still some left but they won't last long.

Model II Owners. Some of you asked for better graphics so we have added SUPER graphics. For $499 plus installation we will put a board in your Model II that adds 32K of RAM memory to support graphics with a 640 by 240 resolution. It comes with a graphics BASIC that adds a lot of commands to generate the graphics (and many of those are very much like the Extended Color Computer graphics without the color) plus machine language subroutines that can be called by your software. For those of you who do not understand 640 by 240 it means that there are 153,600 points (pixels) on your screen and you can turn each one on or off.

There is also another new goody but please look under new products, below, to find it.

Model III Owners. Sorry gang, no new hardware, but there is a lot of new software including our first series of disk based games which includes ZORK, the adventure game so tough it had to be on disk. Also new is Desktop/Plan-80, a neat planning software package. There is also a lot of new educational software.

Color Computer Owners. There are four new Program Paks for you, but you will have to get the catalog to see what they are. There is also the first of our educational cassette software line for the Color Computer that includes some really unique programs. If your Color Computer spends its time in the hands of the young set, take a look at the educational pages.

Pocket Computer Owners. More software for you, too. Engineering Math is a series of four programs that are really useful. Also check out the "book" page—there is a book of Pocket Computer programs for $1.95.

New Products, Part 1. For all of those who have been waiting for the Model IV, sorry, there is none. There is a Model 16 that we introduced in Fort Worth on January 19 so it may not be "new" news. The Model 16 is a 16 bit computer (surprise?) based on the MC68000 processor. It comes in a minimum version for $4999 that includes 128K RAM, a green monitor, and one 8 inch drive that holds 1.25 meg. It expands up to 512K RAM and two drives for 2.5 meg, all in one case. That big version sells for $7495. There is a story inside on this new machine but please read on as there are some items not covered inside.

What is unique about the Model 16 is that it also contains a Z80A processor so it can run any and all Model II software. That software cannot get to the expanded memory, sorry, but it's still a neat trick and very unusual in our industry. Now if we can make a Model 16 be a Model II can we make a Model II be a Model 16? Yes! For $1499 you can buy a kit to be installed in your Model II to make it a Model 16, 128K RAM, MC68000. There are some limitations to this. One is that you cannot expand your memory to 512K, only to 256K and the other is that you stay with the same disk drives that you have. It means that you will be able to run the new software that will be coming for the Model 16.

The Model 16 also provides one more feature—multitasking. This means you can connect one or two terminals to a Model 16 or a Model II turned 16, and run two or three programs all at the same time! Sound neat? We think it is, and for more details please visit your Computer Center.

New Products, Part 2. Also introduced at our January 19 show was the Pocket Computer PC-2. This is an all new Pocket Computer with so many new features I won't even try to list them all here. It sells for $279.95 and its matching four-color printer/two-cassette interface is $239.95. It's expandable with plug in RAM or ROM cartridges and it will be able to talk to the outside world in the future. I don't want to tell it all so watch for a column from the Pocket Computer Product Manager with all the details.

New Products, Part 3. We also introduced a neat new Terminal, the DT-1. This is a slightly intelligent terminal with a 12-inch screen, 80 characters by 24 lines, a full terminal keyboard and both serial and parallel printer ports. It can be connected to the Model 16 or to any computer that uses a dumb terminal or to a modem. It is very state-of-the-art, using an EEPROM to store its settings instead of the usual tiny dip switches. An EEPROM is an EPROM that can be erased electrically. On the DT-1 you enter all the terminal configuration from the keyboard setting baud rate, stop bits, and a whole bunch of user controlled parameters, and they are stored in the EEPROM. Turn off the power and you have lost nothing—it's still all there and without battery backup. And you can reprogram it whenever you like.

The DT-1 sells for $699 and you can also tell it to be an ADDS, a Hazeltine, a Televideo or a Lear Sigler terminal and it will do it automatically. If you have use for a terminal, take a look—this one is winner.

Final Important Note. Not every item I have described here is available at your nearest Radio Shack—the catalog is what is available. Some of these items carry "late availability" dates in the catalog, which were based on our best estimate when we went to press in early January. So please don't write in and say I said it was available because it was in this column. All I am trying to do is show you that we are hard at work in Fort Worth adding lots more items to the biggest microcomputer line in the world. I hope you find at least one of them is just what you were waiting for.

Until next month . . .

Guess the Winning Number

Richard Huttenbrauck
429 Rainey Court
Virginia Beach, VA 23452

With this program you have to guess the correct four digit number that has been picked by the computer. All you do is try to figure out which four numbers ranging from 0 - 9 the computer has picked, and in what order. Further instructions are given in the program text itself. (This program was written on a Model I Level II).

Have fun and good luck!

```
1Ø CLS
     : PRINT TAB(24)"NUMBER SEQUENCE"
     : PRINT TAB(24)"RICH'S PROGRAM"
     : PRINT
     : PRINT"I HAVE SELECTED A RANDOM 4 DIGIT
     NUMBER FOR YOUR TO DE-CODE,"
     : PRINT "THE DIGITS ARE FROM Ø - 9. SELECT
     YOUR COMBINATION OF 4 DIGITS & ";
2Ø PRINT "TRY TO MATCH YOUR SEQUENCE WITH MINE.
     YOU HAVE 1Ø TRIES."
     : PRINT "ALONGSIDE OF YOUR PICKED NUMBER
     YOU WILL SEE 2 DIGITS"
     : PRINT "DISPLAYED. THE FIRST DIGIT STATES
     THE AMOUNT OF CORRECT NUMBERS THAT";
3Ø PRINT " YOU'VE CHOSEN AND THE SECOND DIGIT
     STATES THE AMOUNT OF "
     : PRINT"CORRECT POSITIONS YOU HAVE ENTERED
     THE CORRECT NUMBER IN."
     : PRINT"GOOD LUCK!"
     : PRINT
35 PRINT"TYPE IN 4 NUMBERS AND PRESS ENTER!"
     : PRINT
     : FOR X=1 TO 4
4Ø A(X)=RND(1Ø)-1
     : FOR Y=1 TO 4
     : IF Z(Y)=A(X) GOTO 4Ø
     : NEXT
     : Z(X)=A(X)
     : NEXT
     : FOR Y=1 TO 1Ø
     : PRINT Y"";
     : INPUT A
     : FOR X=1 TO 4Ø
     : PRINT".";
     : NEXT
     : A=A*.ØØ1
     : B(1)=INT(A)
     : T=A-B(1)
     : M=T*1Ø
     : FOR X=2 TO 4
     : B(X)=INT(M)
     : M=M-B(X)
     : M=(M+.ØØ3)*1Ø
     : NEXT
     : H=Ø
     : G=Ø
     : FOR X=1 TO 4
     : IF A(X)=B(X) H=H+1
6Ø IF A(X)=B(1) OR A(X)=B(2) OR A(X)=B(3) OR
     A(X)=B(4) G=G+1
7Ø NEXT
     : IF H+G=7 H=4
8Ø PRINT TAB(4Ø)G;
     : PRINT TAB(48)H
     : IF H+G = 8 PRINT"WHAT LUCK!  YOU WIN"
     : GOTO 1ØØ
85 NEXT
9Ø PRINT
     : PRINT"SORRY MAYBE NEXT TIME!"
     : PRINT"YOUR NUMBERS WERE";
     : PRINT TAB(36)">>>>";
     : FOR X=1 TO 4
     : PRINT A(X);
     : NEXT
     : PRINT" <<<<"
1ØØ PRINT
     : INPUT" TO PLAY AGAIN PRESS ENTER";A$
     : CLS
     : CLEAR
     : GOTO 35
```

Coded Message

For Kids Only

David Snyder
Route 4, Box 952
Bainbridge, GA 31717

Here is a coded message for you to decode. There is only one hint: If you cannot decipher this message, say to yourself, "Aw, hex . . . this guy must be a real ASCII!"

434F4E47524154554C4154494F
4E53212020594F552752452053
4D41525445522054484414E2059
4F55204C4F4F4B212020544841
5427532050524F4241424C5920
57485920594F55205245414420
524144494F20534841434B204D
4943524F434F4D505554455220
4E4557534C45545445522E

P.S. I have a TRS-80 16K Level II microcomputer.

Double Dump to Tape

Robert Pollock
Homewood, IL

Like many Model I/III users, I started with tape and advanced to Disk BASIC. In looking back, the worst part of using tape for program storage was the major inconvenience of waiting for the machine to complete the double and triple program dumps. I since have thought of a simple way to eliminate the waiting. Why I didn't think of it earlier, I'll never know.

To do a double dump to tape, simply type the following:

CSAVE "filename": CSAVE "filename" ⟨ENTER⟩

Now while the computer records two copies of that long program, go walk the dog.

Assembly Language Graphics

by William Barden, Jr.

If you want to be an assembly language programmer, you must learn to follow the rules. In this column we'll provide some of the basic programming rules. This is not a comprehensive list.

RULE #1: No true assembly language programmer **ever** used "code" written by someone else when he could write his own.

Picture this scene: The programming manager is confronting one of his charges: "I thought that Radio Shack already had a package for predicting pork belly commodity prices that runs on the Model III?"

"Uh, well, uh, I don't know if it's exactly what we're looking for... We could add our own sort routines... uh... I don't like the way they do their menus... I think we could speed it up... It wouldn't take long..."

True to form, your assembly language columnist is reinventing the wheel. I **know** Color Computer BASIC has some excellent SET/RESET and PSET/PRESET commands for color graphics. However...

In fact, there are some advantages to using assembly language code for graphics. The BASIC commands are excellent, but for high-speed games, animation, simulation, and other uses, assembly language can be much faster. Of course, one never gets something for nothing. If you will give a little sweat and tears, I'll show you some assembly language code that can be used for a variety of high-speed graphics. This month we'll research the problem and learn about low-resolution graphics; next month I'll show you how the routines can be used for a practical game.

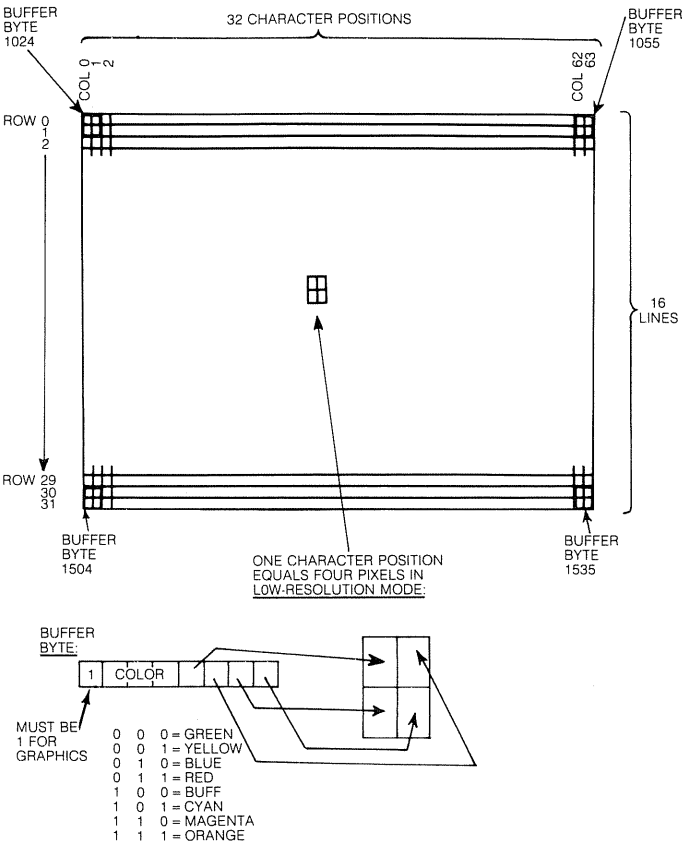


Figure 1. Color Computer Low-Resolution Graphics Mode

LOW-RESOLUTION GRAPHICS

The programs we'll be talking about here run on low-resolution color graphics. There are a number of graphics modes on the Color Computer, ranging all the way up to "high-res" of 256 horizontal elements by 192 vertical elements. The "low-res" graphics here offer 64 horizontal elements by 32 vertical elements. We'll cover the high-resolution graphics in a later column. The low-res graphics mode is a good starting point.

As you know, the Color Computer has 16 lines of 32 characters each, a total of 512 characters, as shown in Figure 1. Each character is held in one byte. The normal "buffer" for the text screen is located starting at location 1024 and continuing through location 1535. (This corresponds to $400 through $5FF, where the "$" is the prefix for hexadecimal.)

Line 0, character position 0 is at RAM location 1024, line 0, character position 1 is at location 1025, and so forth, on through location 1535, which is line 15, character position 31.

Color BASIC uses a low-resolution mode that divides each of the 512 character positions into 4 "pixels," or "picture elements." In this mode, each character position can represent a text character or 4 color elements.

First the bad news: The 4 color elements must be the same color, or black. You can't have two pixels of red, one of green, and one of buff, at least for the same character position. Adjacent character positions can have different colors with no problem. The good news is that this mode is easy to set up and that text and graphics characters can be intermixed. You can have a text "WTB JR" surrounded by red, for example.

Each of the 512 bytes corresponding to a character position is marked as either a text byte or a graphics byte by the first bit, bit 7. (See Figure 1). If this bit is a one, the remainder of the byte represents a text character in ASCII-like code. If bit 7 is a zero, the remaining bits represent a color code and four pixels, as shown in the figure.

The color codes are 0 through 7 and represent green, yellow, blue, red, buff, cyan, magenta, and orange. Note that these are in the same order, but one less than the standard BASIC color codes. The color code is in bits 6 through 4 of the character position byte.

The remaining four bits, bits 3 through 0, represent the four pixels of the character position. If a pixel bit is a one, the pixel is on, and the color is the same as defined in the color code. If the pixel bit is a zero, the pixel is off, or black.

BASIC SET/RESET

The Color BASIC command for setting any pixel is "SET (X,Y,C)," where X is the horizontal coordinate of 0 through 63, Y is the vertical coordinate of 0 through 31, and C is the color code of 1 through 8. RESET is similar, except that the pixel is turned off and no color code need be specified: RESET (X,Y).

We'll emulate the BASIC SET/RESET in our assembly language code.

WASN'T GENERAL ALGORITHM A FRIEND OF PATTON'S?

In theory, all we must do to set a pixel is to find the buffer byte associated with the pixel position and set the bit for the pixel. Simple? Which brings us to our second rule of assembly language programming:

RULE #2: In theory, everything is easy. Implementation is horrendous.

The complicating problem here is the "mapping" of the pixels is not as easy as we would like it to be. Instead of four bits representing four horizontal pixels along a row, four bits represent two rows of two columns as we saw in Figure 1. Can we figure out a general algorithm, or plan, for finding the proper byte in the text buffer for any given x,y pixel? When I first sent this column in, I admitted I couldn't, and ended it at this point. Prompting by Bruce Elliott, however, gave me the motivation to proceed.

If you give it some thought, you'll see that the starting byte number of the line holding the pixel is given by:

LINE ST = INT(Y/2)*32

A pixel of (12,11), for example, is found in the line starting at (INT(11/2)*32), or byte 160. The quantity INT(Y/2) is the "integer" form, the result to the next lower integer. (See INT in BASIC.)

The actual byte holding the pixel is this starting byte number plus INT(X/2). Since each byte holds 2 columns, the X value must be divided by 2; byte 31 along the line holds the pixel bit for X = 62 and X = 63, for example. The "byte displacement" from the start of the text buffer is therefore:

BYTE DISP = INT(Y/2)*32 + INT(X/2),

where both the INT(Y/2) and INT(X/2) are to the next lower integer. Try some examples of this for various X and Ys, and you'll see how the byte can be located.

Once the byte displacement is found, the starting location of the buffer, decimal 1024 can be added to produce the actual RAM location that contains the pixel in question.

The byte in the RAM location can then be picked up, and a one bit put into the pixel bit to set the pixel to the color. Prior to the first SET, the text buffer should be cleared to all graphics by POKEs of 128 + c*16, where C is the color code of 0 through 7. This POKE sets the graphic bit, bit 7, and the color code in bits 6 through 4, but leaves all pixels off, or black. Individual SETs will be used to set each pixel.

A second difficulty is determining which pixel to SET from the X,Y coordinates. Again, it's not obvious without some thought. As it turns out, when Y is even and X is even, bit 3 should be set. When Y is even and X odd, bit 2 should be set. When Y is odd and X even, bit 1 should be set. When Y is odd and X odd, bit 0 should be set. (To that reader in Dubuque - nobody promised you a rose garden, fellah . . .). This can all be summed up as follows:

		LS Bits		Mask
Y	X	Y	X	3 2 1 0
even	even	0	0	1———
even	odd	0	1	—1——
odd	even	1	0	——1—
odd	odd	1	1	———1

The "mask" value represents which bit of bits 3 through 0 should be set.

A RESET is very similar except that instead of a one bit being put into the pixel bit, a 0 bit is stored instead.

THE ASSEMBLY LANGUAGE CODE

Armed with our trusty algorithm, we'll surge forward into the coding. The result of hours of work is shown in Listing 1. Lots of assembly language programs are done "inward out." And so it was here—the core routine is the ADDMSK subroutine. ADDMSK finds the text buffer address for a given X,Y pixel and also supplies the mask value. The SET/RESET action is taken in the SET and RESET code.

THE ADDMSK SUBROUTINE

ADDMSK assumes that the X,Y coordinates are in the A and B registers of the 6809E. Remember that the A and B registers are 8-bit registers that can be grouped together into a 16-bit register called the D register.

The assembly language code is on the right of the listing; the resulting "machine-language" bytes are the hexadecimal values on the left. Let's follow ADDMSK.

First, the D register (A and B) is saved in the stack. This is the "hardware" stack, the RAM area used to store temporary results and subroutine return addresses, among other things. (Recall that we've got a second stack area called the "user" stack, which is pointed to by the U register. We won't complicate things by using the user stack in these subroutines.)

The X and Y values are saved in the stack so that we can restore them a little later.

The LSRB shifts Y in the B register right one bit position. This divides Y by 2 with the least significant bit being discarded, effectively an "INT" action.

Next, A is loaded with decimal 32. The "#" indicates "immediate" addressing, where the operand is in the instruction bytes themselves ($20 is decimal 32). A multiply multiplies 32 times y/2 and puts the result in A,B (D). We've found the start of the line containing the pixel.

Next, hex 400 (1024 decimal) is added to the result. We're now pointing to the actual buffer byte that represents the start of the line. This value is saved on the stack by the PSHS D.

Now here's a subtle instruction: The LDB 2,S takes the stack pointer S register, adds 2 to its contents, and then uses the result as a memory address for the load into B. The S register always points to the last stored value on the stack. Remember that the stack builds down in memory. This instruction picks up the X value from the first PSHS.

The CLRA clears the A register. The LSRB shifts B right one bit position. We've now got X/2 in D (A,B). The ADDD ,S++ adds X/2 to the previous result, and we're now pointing to the actual byte in the buffer containing the pixel. The ADDD adds two bytes from the stack; the stack pointer S still

Listing 1

```
3FØØ                ØØ1ØØ            ORG     $3FØØ
                    ØØ11Ø   ************************************************
                    ØØ12Ø   * SET SUBROUTINE. SETS X,Y POINT IN LOW-RES MODE    *
                    ØØ13Ø   * CALLED FROM BASIC WITH X,Y IN ARGUMENT 8,8        *
                    ØØ14Ø   ************************************************
3FØØ    17    74EA  ØØ15Ø   SET     LBSR    $B3ED     GET ARGUMENTS
3FØ3    8D    25    ØØ16Ø            BSR     ADDMSK    FIND ADDRESS, MASK

3FØ5    1F    89    ØØ17Ø            TFR     A,B       SAVE MASK IN B
3FØ7    A4    84    ØØ18Ø            ANDA    ,X        TEST ALREADY SET
3FØ9    27    ØA    ØØ19Ø            BEQ     SETØ1Ø    GO IF NOT SET
3FØB    E6    84    ØØ2ØØ            LDB     ,X        SET, FIND COLOR
3FØD    C4    7Ø    ØØ21Ø            ANDB    #$7Ø      MASK OUT COLOR
3FØF    54          ØØ22Ø            LSRB              ALIGN FOR 1-7 CODE
3F1Ø    54          ØØ23Ø            LSRB
3F11    54          ØØ24Ø            LSRB
3F12    54          ØØ25Ø            LSRB
3F13    2Ø    Ø5    ØØ26Ø            BRA     SET9Ø     RETURN
3F15    EA    84    ØØ27Ø   SETØ1Ø    ORB     ,X        SET PIXEL
3F17    E7    84    ØØ28Ø            STB     ,X        RESTORE
3F19    5F          ØØ29Ø            CLRB              Ø FOR SET ACTION
3F1A    4F          ØØ3ØØ   SETØ9Ø    CLRA              CLEAR A FOR CONVERT
3F1B    17    75D6  ØØ31Ø            LBSR    $B4F4     CONVERT ARGUMENT TO INTEGER
3F1E    39          ØØ32Ø            RTS               RETURN
                    ØØ33Ø   ************************************************
                    ØØ34Ø   * RESET SUBROUTINE. RESETS ONE PIXEL IN LOW RES MODE*
                    ØØ35Ø   * CALLED FROM BASIC WITH X,Y IN ARGUMENT 8,8        *
                    ØØ36Ø   ************************************************
3F1F    17    74CB  ØØ37Ø   RESET   LBSR    $B3ED     GET ARGUMENTS
3F22    8D    Ø6    ØØ38Ø            BSR     ADDMSK    FIND ADDRESS, MASK
3F24    43          ØØ39Ø            COMA              FLIP MASK FOR RESET
3F25    A4    84    ØØ4ØØ            ANDA    ,X        RESET PIXEL
3F27    A7    84    ØØ41Ø            STA     ,X        RESTORE BYTE
3F29    39          ØØ42Ø            RTS               RETURN
                    ØØ43Ø   ************************************************
                    ØØ44Ø   * FIND LOW-RES ADDRESS, MASK                       *
                    ØØ45Ø   *       INPUT: A,B=X,Y                            *
                    ØØ46Ø   *       OUTPUT:X=BUFFER ADDRESS                   *
                    ØØ47Ø   *              A=MASK                             *
                    ØØ48Ø   ************************************************
3F2A    34    Ø6    ØØ49Ø   ADDMSK   PSHS    D         SAVE X,Y
3F2C    54          ØØ5ØØ            LSRB              FIND START OF LINE
3F2D    86    2Ø    ØØ51Ø            LDA     #32       32 IN A, Y IN B
3F2F    3D          ØØ52Ø            MUL               START OF LINE
3F3Ø    C3    Ø4ØØ  ØØ53Ø            ADDD    #$4ØØ     ADD START OF BUFFER
3F33    34    Ø6    ØØ65Ø            PSHS    D         NOW ON STACK
3F35    E6    62    ØØ55Ø            LDB     2,S       GET X
3F37    4F          ØØ56Ø            CLRA              NOW IN D
3F38    54          ØØ57Ø            LSRB              X/2
3F39    E3    E1    ØØ58Ø            ADDD    ,S++      BUFFER+LINE ST+X/2
3F3B    1F    Ø1    ØØ59Ø            TRF     D,X       NOW IN X
3F3D    A6    E4    ØØ6ØØ            LDA     ,S        GET X
3F3F    43          ØØ61Ø            COMA              COMPLEMENT BIT
3F4Ø    84    Ø1    ØØ62Ø            ANDA    #1        FIND ODD/EVEN
3F42    4C          ØØ63Ø            INCA              EVEN TO 1Ø, ODD TO Ø1
3F43    A7    E4    ØØ64Ø            STA     ,S        SAVE
3F45    E6    61    ØØ65Ø            LDB     1,S       GET Y
3F47    C4    Ø1    ØØ66Ø            ANDB    #1        GET ODD/EVEN
3F49    26    Ø2    ØØ67Ø            BNE     ADDØ9Ø    GO IF ODD (MASK=1Ø OR Ø1)
3F4B    48          ØØ68Ø            LSLA              ALIGN (1ØØØ OR Ø1ØØ)
3F4C    48          ØØ69Ø            LSLA
3F4D    32    62    ØØ7ØØ   ADDØ9Ø   LEAS    2,S       RESET STACK
3F4F    39          ØØ71Ø            RTS               RETURN
          ØØØØ      ØØ72Ø            END
ØØØØØ  TOTAL ERRORS
```

points to the previous result. The " + + " increments the stack pointer by 2 after the add, "resetting the stack pointer" so that it now points to the data pushed on the first PSHS. The TFR transfers the result in D to the X register, the 16-bit index register.

We now have the address of the buffer byte containing the pixel in X. We need to find which bit should be set or reset. To explain this, we need RULE 3.

RULE #3: Always try to be clever, even if it takes hours.

The code from line 600 through 690 produces a "mask value" in A. The mask value is binary 00001000, 00000100, 00000010, or 00000001, depending upon which pixel is involved. The code is not as straightforward as it could be. Your columnist fell headlong into rule number 3 and must now take 8 paragraphs to hem and haw about why he did it this way. (The truth is I wanted to speed up the code. Ahem . . .)

The LDA ,S gets the original X value from the stack. This value is complemented by the COMA - all ones are changed to zeroes and all zeroes to ones. Next, the least significant bit, bit 0, is found by the ANDA #1. One is added to this bit by the INCA. The result is 10 if X is even, and 01 if X is odd. The result is stored back in the stack.

The LDB 1,S gets the original Y. An ANDB #1 finds the least significant bit. If this bit is not 0 (BNE) the two "shift left" instructions are bypassed, and the mask remains 0010 or 0001. If the bit is 0 (Y even), the mask becomes 1000 or 0100.

We now have the address of the byte containing the pixel in X and the mask for the set/reset in A.

The instruction at ADD090 resets the stack pointer by adding 2 to the contents of S. This makes S point back beyond the data "pushed" in this subroutine.

THE RESET SUBROUTINE

This subroutine is "called" from a BASIC program to reset a pixel. It, in turn, calls ADDMSK. The call to reset uses the USR call in Color BASIC or the USRn call in Extended Color BASIC. The form of this call is "X = USRn(N)." The N parameter is a 16-bit "integer" value of -32768 through +32767. In fact, the upper 8 bits of this value represent X, and the lower 8 bits represent Y. A call to RESET 20,30, for example, could be made by

Z = USRn(20*256+30)

Multiplying by 256 aligns X into the upper 8 bits. The "Z" parameter is a "dummy" and can be any variable.

The protocol for passing the argument to RESET says that a call to the integer convert subroutine at ROM location $B3ED must be done. This converts a "floating-point" form of the argument into a 16-bit value (see Extended BASIC manual). After the LBSR $B3ED, the A and B registers contain X and Y, respectively, from BASIC.

The BSR ADDMSK now finds the byte location of the pixel in X and the mask in A. The mask is complemented for the RESET. For example, a mask of 00001000 becomes 11110111. The ANDA passes all bits except for the pixel involved, resetting that pixel with a zero. A STA then stores the modified byte back into the text buffer.

The RTS instruction returns to the BASIC statement following the USR call; the pixel has now been reset by storing a 0 in the appropriate bit.

THE SET SUBROUTINE

The SET subroutine is somewhat more involved than RESET. Again, the X,Y parameters are passed from BASIC by a call to INTCNV at ROM location $B3ED, and the byte location and mask are found by a call to ADDMSK.

SET then tests the bit to be set. If it is already set (BEQ SET010) the byte containing the pixel is ANDed with #$70. This value (binary 01110000) masks out the color code. Four LSRBs shift the color code four bit positions to the right and it is returned to BASIC as a parameter. If the bit is not already set, it is set to a one and a zero is returned to BASIC.

The LBSR $B4F4 is the standard way to pass back an argument from machine language to BASIC. The ROM routine at $B4F4 converts an integer value of -32768 through +32767 to a floating point value for BASIC. The value to be converted is in A,B (D) before the LBSR. The value is set equal to the variable equated to the USR call in BASIC. Z=USRn(X), for example, returns the argument as variable Z.

The idea behind returning a color code is to detect "boundary colors" which may be useful in game playing. More on that aspect in the next column.

ASSEMBLING THE PROGRAM

The SET/RESET/ADDMSK program was assembled by the Color Computer Editor Assembler. You may choose to key in the "source code," the assembly language statements on the right side of the listing, and reassemble it for practice. After keying in the source code, assemble and compare the machine code on the left side of the listing. It should be identical. Now assemble and write an object file to cassette.

The object file can be loaded from cassette by the following procedure:

1. CLEAR 200,&H3EFF from BASIC. This reserves the $3F00 area for the machine code.
2. CLOADM the object file. This loads the machine code into the $3F00 area.

THE CODE INCORPORATED AS DATA STATEMENTS

An alternative to reassembling is to use BASIC DATA statements for the code. Listing 2 shows this method. Listing 2 shows the code as a series of DATA statements of 10 per line. The code is moved from the DATA statements to the $3F00 area by the first part of the BASIC program. Each DATA value corresponds to one machine language byte from Listing 1.

Listing 2

```
100 ' MOVE MACHINE LANGUAGE
110 DATA 23,116,234,141,37,31,137,164,132,39
120 DATA 10,230,132,196,112,84,84,84,84,32
130 DATA 5,234,132,231,132,95,79,23,117,214
140 DATA 57,23,116,203,141,6,67,164,132,167
150 DATA 132,57,52,6,84,134,32,61,495,4
160 DATA 0,52,6,230,98,79,84,227,225,31
170 DATA 1,166,228,67,132,1,76,167,228,230
180 DATA 97,196,1,38,2,72,72,50,98,57
190 FOR I=&H3F00 TO &H3F4F
200 READ A
210 POKE I,A
220 NEXT I
230 ' DEFINE MACHINE-LANGUAGE
240 DEFUSR0=&H3F00
250 DEFUSR1=&H3F1F
260 ' NEXT GRAPHICS CLEAR SCREEN WITH ORANGE
270 FOR I=1024 TO 1024+511
280 POKE I,240
290 NEXT I
300 ' NEXT SET ALL 2048 PIXELS
310 FOR X=0 TO 63
```

```
320 FOR Y=Ø TO 31
330 A=USRØ(X*256Z+Y)
340 NEXT Y
350 NEXT X
360 ' NEXT RESET ALL PIXELS
370 FOR Y=Ø TO 31
380 FOR X=Ø TO 63
390 A=USR1(X*256+Y)
400 NEXT X
410 NEXT Y
420 ' NOW SET ONE PIXEL AND TEST
430 SET (RND(63),RND(31),8)
440 FOR X=Ø TO 63
450 FOR Y=Ø TO 31
460 A=USRØ(X*256+Y)
470 IF A<>Ø THEN STOP
480 NEXT Y
490 NEXT X
```

CALLING THE SUBROUTINES FROM BASIC

Listing 2 shows a BASIC test driver that will help you get familiar with the program. This version is for Extended Color BASIC, but we'll explain the modifications required for Color BASIC.

After relocating the code, the BASIC driver defines the SET and RESET subroutine starting points by the DEFUSR statements. DEFUSR0 is SET and DEFUSR1 is RESET.

Next, the screen is cleared to all graphics characters by a POKE of 240 into locations 1024 through 1535. This resets all pixels and sets the color codes to orange, or 7.

Next, SET is called by USR0 for all pixels. The coloring is done as a series of vertical bars that sweep from left to right.

Next, RESET is called by USR1 for all pixels. The clearing is done as a series of horizontal bars that sweep from top to bottom.

Finally, the BASIC SET command is used to set one pixel at random. SET is then called by USR0 to set pixels unless the pixel is already set. In this case the program STOPs.

PASSING THE ARGUMENTS FOR X AND Y

The USR calls in the program show the procedure I described earlier for passing X and Y. The X parameter is multiplied by 256 and the Y parameter is added to it. This puts X in the "most significant" 8 bits and Y in the "least significant" 8 bits of the argument.

PROCEDURE FOR COLOR BASIC

Color BASIC uses USR in place of USRn. The USR command assumes that the starting location of the subroutine has been stored in locations 275 and 276. Do a POKE 275,63 and a POKE 276,0 before each call for SET and a POKE 275,63 and POKE 276,31 before each call for RESET. Also, change the USR0 and USR1 to "USR." Everything else remains the same.

HOW DOES SET/RESET COMPARE?

To evaluate SET/RESET, we need Rule #4. RULE #4: Never admit that your program may be not quite as good as another.

If you run the normal SET/RESET in Color BASIC and time it, and then run our assembly language version, you'll see that our version runs at about the same rate . . . All right, it **is** slightly less rapid . . . That is to say, somewhat slower . . . Aw shucks, it just isn't as fast as Color BASIC.

Honestly though, the somewhat slower speed of our SET/RESET isn't the point. We now have our own code which can be interfaced with other assembly language code. The result will be very powerful assembly language code that will provide functions that are not implemented in Extended Color BASIC or that will be significantly faster. We'll see examples of some of these in the next column.

THE OBLIGATORY END OF COLUMN SERMON

In the meantime, study the examples here. As I mentioned in the first column, I can't provide a series of assembly language tutorials in this column. You'll need to do a significant amount of study on your own. What I will do is present a detailed example of an assembly language application for each column and show you how I went about solving the problem. We'll get into more and more aspects of 6809E assembly language with each column. Between columns, crack those books!

DELTASIN

David Sligar
882-D Revere Village Court
Centerville, OH 45459

I am sure that Color Computer owners will enjoy the impressive graphics this program generates. This program is a modification of sample program #9—Sine Wave, listed in the back of Going Ahead With Extended Color Basic. My program, which I call "DELTASIN" will produce a three-dimensional sine wave.

By increasing the value of N in statement 35 (try N = 720 or N = 960) you can increase the density of the points plotted, but the program will take longer to run. (Lower values of N have the opposite effect.)

By changing the value of R in statement 40, you can vary the amplitude of the three-dimensional sine wave.

```
1 '***DELTASIN***
2 '
5 PMODE 4,1
10 PCLS
15 SCREEN 1,1
20 PI=3,14159
24 Q=1
25 A1=-4*PI
30 A2=4*PI
35 N=360
40 R=20
41 IF A>1 THEN R=R+Q
45 X=(A2-A1)/N
50 F=255/(A2-A1)
55 FOR I=A1 TO A2 STEP X
60 X=I*F
65 Y=R*SIN(I+Q)
70 PSET
75 NEXT I
77 Q=Q+7
78 IF Q>56 THEN GOTO 78
80 GOTO 25
```

Relocating Machine Language Programs

This month we have some information for you that should reinforce your confidence in Radio Shack's wanting to support what we sell. One question which is asked several times a day is, "Why won't Pyramid (or Raaka-Tu, or Eliza, or any other Machine Language Tape Based program) run from my disk system? I just paid $400.00 to upgrade to disk and nothing will run."

The answer to that question is very simple. These games were designed to load into a 16K computer, and therefore, load very low in user RAM. That just happens to be the same area where TRSDOS loads. When you load a game on top of TRSDOS, the computer will either re-boot or just "hang up."

First, let me assure you that it has never been a matter of not wanting to help you. Due to the work of many dedicated individuals, we are now able to give you a routine which will allow you to relocate the programs to an area of memory which does not interfere with TRSDOS. So much for the answer. Now on to the solution.

In general, to get a tape based, machine language program onto disk, you are going to go into DEBUG to enter two very short machine language programs. The first will move your program up into high memory (out of the way of TRSDOS) and the second will move it back to its normal place and execute it.

The Hexadecimal values shown in the tables are the actual program addresses. For a machine language program, any address used must have the LSB (Least Significant Byte) first, then the MSB (Most Significant Byte) when entered into the two move routines.

Example:

Desired MSTART - 7C20	Enter as 20 7C
Desired LENGTH - 0D10	Enter as 10 0D
Desired ENTRY - 4AF6	Enter as F6 4A

To help, we've broken them into columns and labeled them as S1 & S2 (the 2 byte start address), E1 & E2 (the 2 byte execute address), L1 & L2 (the 2 byte length address), and M1 & M2 (the 2 byte Move Start address). Remember, these are only the "variable names." Use the values from the Address Allocation Table in this article.

Here is the Move-Up routine:

```
21 S2 S1     LD HL,START
11 M2 M1     LD DE,MSTART
Ø1 L2 L1     LD BC,LENGTH
ED BØ        LDIR
C3 ØØ ØØ     JP REBOOT
```

So, what does this mean? Let's take an example. For Pyramid, we would type in, under DEBUG, 21 00 43 11 20 80 01 FE 3C ED B0 C3 00 00. We will continue to use Pyramid in the examples, but you should follow the table for any of the other programs. Just change the values for S2, S1, M2, M1, L2 and L1.

Now, the Move-Back routine:

```
21 M2 M1     LD HL,MSTART
11 S2 S1     LD DE,START
Ø1 L2 L1     LD BC,LENGTH
ED BØ        LDIR
C3 E2 E1     JP ENTRY
```

From the table:

```
START        43ØØ
END          7FFE
ENTRY        43ØØ
LENGTH       3CFF
MSTART       8Ø2Ø
MEND         BD4Ø
MENTRY       BD3Ø
MOVE-UP      8Ø1Ø
```

1. From TRSDOS Ready, type: (For Model I, press ⟨**BREAK**⟩ after pressing ⟨**ENTER**⟩.

```
DEBUG <ENTER>
M
8Ø1Ø <SPACE BAR> (Move up from table)
```

2. Enter the Move-Up routine: (For Model I, press ⟨**SPACEBAR**⟩ after each pair of numbers.)

```
21 ØØ 43 11 2Ø 8Ø Ø1 FF 3C ED BØ C3 ØØ ØØ
```

3. Press ⟨**ENTER**⟩ to finish that entry and ⟨**M**⟩ again to enter the Move-Back routine:

```
<ENTER>
M
BD3Ø <SPACE BAR> (MENTRY from the table)
```

4. Enter the Move-Back program:

```
21 2Ø 8Ø 11 ØØ 43 Ø1 FF 3C ED BØ C3 ØØ 43
```

5. Exit to ROM BASIC by holding down the BREAK key while pressing RESET.
6. Press ⟨**ENTER**⟩ to answer the prompt(s), then type SYSTEM ⟨**ENTER**⟩. When *? appears, type PYRMD ⟨**ENTER**⟩. When *? appears again, type /32784 ⟨**ENTER**⟩. (The address 32784 is from the Exec Moveup Column of the table, and causes execution of the Move-Up program.)

```
<ENTER>
SYSTEM <ENTER>
*? PYRMD <ENTER>
*? /32784 <ENTER>
```

(TRSDOS will automatically re-boot)

7. Then, at TRSDOS Ready, type the following to create your disk file:

(for a Model III)

```
DUMP PYRMD/CMD (START=8Ø2Ø,END=ØBD4Ø,TRA=
ØBD3Ø) <ENTER>
```

or (for a Model I)

```
DUMP PYRMD/CMD (START=X'8Ø2Ø',END=X'ØBD4Ø',
TRA=X'ØBD3Ø') <ENTER>
```

Notice that a zero has been added before any number which begins with a letter.

Now let's do another example—Invasion Force. From the table:

```
START     5ØØØ
END       71ØØ
ENTRY     5ØØØ
LENGTH    21Ø1
MSTART    712Ø
MEND      924Ø
MENTRY    923Ø
```

1. From TRSDOS Ready, type:

```
DEBUG <ENTER>
M
711Ø <SPACE BAR> (Move up from the table)
```

2. Enter the Move-Up routine:

```
21 ØØ 5Ø 11 2Ø 71 Ø1 Ø1 21 ED BØ C3 ØØ ØØ
```

3. Press ⟨**ENTER**⟩ to finish that entry and ⟨**M**⟩ again to enter the Move-Back routine:

```
<ENTER>
M
923Ø <SPACE BAR> (MENTRY from the table)
```

4. Enter the Move-Back program:

```
21 2Ø 71 11 ØØ 5Ø Ø1 Ø1 21 ED BØ C3 ØØ 5Ø
```

5. Exit to ROM BASIC by holding down the BREAK key while pressing RESET.
6. Answer the Prompt(s). When Ready appears, type in SYSTEM ⟨**ENTER**⟩. After *? appears, type in INVADE ⟨**ENTER**⟩ and let the program load. When *? appears, type in /28944. This will cause the Move-Up program (which is located at 7110 Hex or 28944 Decimal) to execute.

```
<ENTER>
SYSTEM <ENTER>
*? INVADE <ENTER>
*? /28944 <ENTER>
```

(TRSDOS will automatically re-boot)

7. Then, at TRSDOS Ready, type the following to create your disk file:

```
DUMP INVADE/CMD (START=712Ø,END=924Ø,
TRA=923Ø) <ENTER>
```

Or, on a Model I, when DOS READY appears, type:

```
DUMP INVADE/CMD (START=X'712Ø',END=
X'924Ø',TRA=X'923Ø') <ENTER>
```

If you follow the procedure outlined here EXACTLY as shown, you shouldn't have any trouble. If your "moved" program fails to execute, check the following:

1. Correct addresses used for the program moved, M1,M2,S1,S2,L1,L2,E1 and E2.Correct addresses used and correct programs entered for Move-up and Move-Back.
2. Correct decimal execute address used for Move-up

Address Allocation Table

NOTICE TO MODEL I OWNERS: This information has not been tested on a Model I. We know that you will have problems on programs like Raaka-Tu and Pyramid. A follow-up article in the near future will give you a reliable way to move these other programs.

Program Name	START S1 S2	END	ENTRY E1 E2	LENGTH L1 L2	MSTART M1 M2	MEND	MENTRY Move Back	Move Up	Exec Moveup	Required Memory Size
EDITOR ASSEMBLER	43 00	5D 40	46 BA	1A 41	70 00	8A 60	8A 50	60 00	24576	32K
SERIES I EDITOR ASSEMBLER	4A EE	63 10	4B EA	18 23	70 00	88 40	88 30	63 20	25376	32K
T-BUG	43 80	48 24	43 A0	04 A5	70 00	74 C0	74 B0	60 00	24576	32K
PROGRAM CONVERSION	4A F6	4F 7D	4A F6	04 88	70 00	74 A0	74 90	60 00	24576	32K
DATA CONVERSION	43 00	45 25	43 00	02 26	70 00	72 40	72 30	60 00	24576	32K
IN MEMORY-INIT	43 80	47 8C	43 80	04 0D	70 00	74 20	74 10	60 00	24576	32K
IN MEMORY-RETREV	43 80	4B 8C	43 80	08 0D	70 00	78 20	78 10	60 00	24576	32K
IN MEMORY-SORT	43 80	46 DE	43 80	03 5F	70 00	73 70	73 60	60 00	24576	32K
TERM	50 00	50 BF	50 00	00 C0	70 00	70 E0	70 D0	60 00	24576	32K
ELIZA	50 00	78 00	50 00	28 01	78 20	A0 40	A0 30	78 10	30736	32K
TALKING ELIZA	46 00	7C 00	46 00	36 01	7C 20	B2 40	B2 30	7C 10	31760	32K
MICRO MOVIE	43 00	4C FF	43 00	0A 00	70 00	7A 20	7A 10	60 00	24576	32K
MICRO MUSIC	43 00	49 70	43 00	06 71	70 00	76 90	76 80	60 00	24576	32K
MICRO MARQUEE	4A 00	4F FF	4B 00	06 00	70 00	76 20	76 10	60 00	24576	32K
INVASION FORCE	50 00	71 00	50 00	21 01	71 20	92 40	92 30	71 10	28944	32K
CHECKERS-80	50 00	77 00	50 00	27 01	77 20	9E 40	9E 30	77 10	30480	32K
PYRAMID	43 00	7F FE	43 00	3C FF	80 20	BD 40	BD 30	80 10	32784	32K
RAAKA-TU	43 00	7F FF	43 00	3D 00	80 20	BD 40	BD 30	80 10	32784	32K
HAUNTED HOUSE	42 E9	4F FF	42 E9	0D 17	70 00	7D 30	7D 20	60 00	24576	32K
SNDFLR	43 5E	4E A0	43 5E	0B 43	70 00	7E 60	7E 50	60 00	24576	32K
ASTROLOGY	43 00	7F FF	43 00	3D 00	80 20	BD 40	BD 30	80 10	32784	32K
PADDLE PINBALL	50 00	74 C1	60 B0	24 C2	75 00	99 E0	99 D0	74 D0	29904	32K
FLYING SAUCER	42 EA	4F FA	42 EA	0D 11	70 00	80 30	80 20	70 00	28672	32K
TAPE SCRIPSIT MI	43 00	69 C5	43 00	26 C6	70 00	96 E0	96 D0	69 D0	27088	32K
TAPE SCRIPSIT MIII	42 E9	6C A0	43 03	29 B8	70 00	99 D0	99 C0	6C B0	27824	32K
TINY PASCAL MI	58 40	7F FD	58 40	27 BE	80 00	A7 D0	A7 C0	A7 D0	42960	32K
PAS32K (32K Tiny Pascal)	4D 90	73 C6	4D 90	26 37	74 00	9A 50	9A 40	73 D0	29648	32K

after you loaded your program from disk.

3. Correct addresses used for DUMP to disk, especially the execute address (TRA = xxxx)

NOTE

This procedure WILL NOT make Model I-only programs (PASCAL, TALKING ELIZA, T-BUG, etc.) work on a Model III. Neither will this make your tape program use the disk for anything other than being able to load the program from it. If the program used cassette storage for data before you moved it to disk, it will still use cassette for data storage.

To execute Pyramid from disk at TRSDOS Ready

```
Model III Type:   PYRMD <ENTER>
Model I Type:     LOAD PYRMD/CMD <ENTER>
                  BASIC2 <ENTER>
                  <ENTER>
                  SYSTEM <ENTER>
                  /48432 <ENTER> (Decimal
                  equivalent of MENTRY from the
                  table.)
```

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

"CLEAR"

(This article originally appeared in the Radio Shack Computer Center Newsletter in Rochester, New York.)

The TRS-80 stores string information in a special area in RAM memory called "string storage." When you turn your computer on, this space is initially set to 50 bytes. That is, space is reserved to store 50 characters. If you begin using many strings, or a string longer than 50 characters, you will see an "OS" error which means "Out of String space," or "Oh Shucks!" When this happens, of course, you simply reserve more string space by adding "CLEAR n" to your program. (The "n" is a number which indicates how many bytes of memory you wish to set aside for strings). Everybody knows that! But not everyone is aware of the full effect of CLEARing string space.

CLEAR does more than just reserve space. It also erases all the variable names that you have used in the program up to that point by setting all numeric variables to 0 and all string variables to null. That means that CLEAR should be used very early in the program, preferably in the very first line. You can NOT say CLEAR in the body of your program anytime you happen to think of it. You will erase all your data if you do! Consult your Level II manual, page 4/4. See the Model III manual pages BA 2/2, 4/4, 5/1. Model II users should look at the BASIC section of their manual at page 3/22.

But that is not all that CLEAR does. CLEAR also nullifies the effect of any DIMension statements and DEFine statements. As an example, let us suppose that you need to use the extra accuracy of double precision numbers in your program, and you intend to build an array with 100 double precision numbers in it. Here is what you (mistakenly) write:

```
10 CLS: DEFDBL A,B: DIM A(100)
20 CLEAR 500
```

Now, when your program RUNs you will be surprised to find that the numbers are not accurate. Furthermore, your array will not hold 100 numbers as you expected, but only 10, and your program crashes with a "BS" error. CLEAR caused the problem. The effect of the CLEAR in line 20 is to cancel the DEF and DIM statements in line 10 as if that line did not exist. The cure? Simply write CLEAR 500 as the first statement in line 10 and eliminate line 20.

Furthermore, loading and running a new program does not restore string storage to 50 bytes unless you have a CLEAR statement in the program which specifically does just that. Thus if you load a program early in the day which reserves 10,000 bytes for string storage, that area remains reserved until you CLEAR it again or turn off the computer. Later attempts to load a long program may cause an "OM" error ("Out of Memory" or "Oh My!") which will lead you to believe that your computer needs repair. Actually, all it needs is for you to type in CLEAR 50.

On the Model II TRS-80 and the Color Computer, CLEAR can also be used to reserve areas in RAM memory for other purposes. For example:

```
10 CLEAR 100, 32000
```

sets string space to 100 bytes and specifies 32000 as the highest memory address which BASIC can use to run your program. Memory above this address will not be changed by BASIC and you may use this area to store a machine language program.

Carefully check programs which you copy from magazines and other sources. You sometimes see the CLEAR error made. Now is it all CLEAR?

Shop-at-Home Now Available on CompuServe™

With Comp-U-Star, Shopping May Never Again be the Same

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.

For the past eight years, nearly anyone with a telephone has been able to dial a number and order merchandise of just about any imaginable type, make or model at discounts from 10 to 40 percent. Once ordered, this merchandise, from crystal to television sets, would then be delivered to their door. All of this was through the efforts of a Stamford, Connecticut-based company called Comp-U-Card.

Comp-U-Card is now available on the CompuServe Information Service, enabling CompuServe customers, with the aid of their personal computers, to "scan" the pages of the ultimate catalog, Comp-U-Star. "It's a premier electronic shopping service," remarks Kirk Shelton of Comp-U-Card, developers of the Comp-U-Star service.

Comp-U-Star allows customers to purchase from a selection of more than 30,000 items without ever bothering to set foot outside their front door.

HOW DOES IT WORK?

"Comp-U-Star is really an interactive electronic shopping service. It consists of a database of consumer products which allows a customer to either access an individual item by specifying make and model or to 'shop' in the database by describing the product by its various characteristics," explains Shelton.

After logging into the CompuServe Information Service and selecting Comp-U-Star, a customer can check the price and features of a specific item. He can also enter a model name and receive detailed information about the item. Comp-U-Star gives the manufacturer's suggested retail price as well as the Comp-U-Star member price.

By asking about several different products in the same category and using the prices and features as guidelines, a shopper can choose the item best suited to his needs at the most reasonable price.

For those uncertain of particular makes and models, Comp-U-Star allows the user to present generic characteristics only. It will then search through its database and present all the items that fit those specific characteristics.

For a yearly membership fee plus CompuServe's standard connect charge, shoppers are able to check out the latest merchandise from a selection that spans 18 different categories. Shelton says merchandise on Comp-U-Star is offered through cooperation with a network of vendors that sell only brand name goods. When merchandise is ordered through Comp-U-Star, it is shipped in factory-sealed cartons and includes a full warranty.

Items may be paid for by check or major credit card. If a purchase price is particularly large, Shelton said that payments may be spread among more than one credit card. This is helpful for people who want to charge an item but are faced with a ceiling on individual credit cards.

What does the future hold for Comp-U-Star and its customers? Well, believe it or not, shoppers may never again have to haggle with car dealers for the best price on a new car, because Comp-U-Star will soon offer automobiles. And with the cost of "getting around" continually going up, Comp-U-Star will make you think twice about driving from one retail outlet to another to do your comparative shopping.

Comp-U-Card on CompuServe. Why leave home if you've got it?

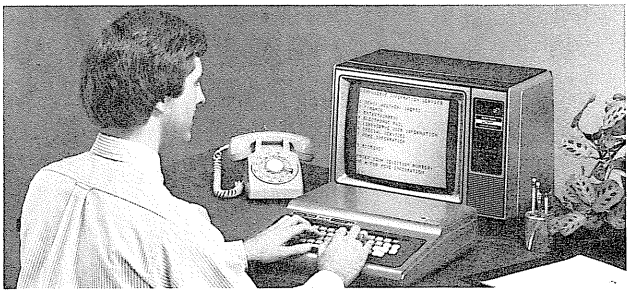
CompuServe ALSO AVAILABLE VIA DATAPAC IN CANADA

The CompuServe Information Service is available to owners of personal computers and computer terminals through Radio Shack stores in Canada.

Using a personal computer or terminal, a telephone and a Western Electric compatible 212 modem or coupler, subscribers in Canada can now access CompuServe's information via the Datapac telecommunications network.

The standard CompuServe fee for service is $5 per hour weekday evenings, all day weekends and holidays. There is an additional communications surcharge of $12.50 per hour for 300 baud rate.

Canadian customers can take advantage of the full range of services on CompuServe including the latest news from major newspapers and a national U.S. wire service, corporate stock and commodities trading information, home banking, electronic mail and real-time communications, computer games, family information and computing power for programming activities.



Phone Numbers for Accessing CompuServe in Canada

British Columbia

Kamloops	604/374-5941
Kelowna	604/860-0331
Nelson	604/354-4411
Prince George	604/564-4060
Terrace	604/635-7221
Vancouver	604/683-8711
Victoria	604/388-9300

Alberta

Calgary	403/264-9340
Edmonton	403/420-0185
Fort McMurray	403/791-2884
Grande Prairie	403/539-5990
Lethbridge	403/329-8755
Medicine Hat	403/526-6587
Red Deer	403/343-7200

Saskatchewan

Moose Jaw	306/693-7611
Regina	305/565-0111
Saskatoon	306/665-6660

Manitoba

Brandon	204/725-0878
Winnipeg	204/475-2740

Ontario

Barrie	705/737-4100
Brampton	416/791-8900
Brantford	519/756-0000
Chatham	519/354-7710
Clarkson	416/823-6000
Cornwall	613/938-9700
Galt	519/622-1714
Guelph	519/836-7930
Hamilton	416/523-6800
Kingston	613/549-7720
Kitch.-Waterloo	519/579-0009
London	519/679-7500
Niagara Falls	416/357-2702
North Bay	705/476-3900
Oshawa	416-579-8920
Ottawa	613/567-9100
Peterborough	705/748-6940
St. Catherines	416/688-5620
Sarnia	519/336-9920
Sault Ste. Marie	705/942-4690
Sudbury	705/673-9602
Toronto	416/868-4000
Windsor	519/255-1000
Woodstock	519/485-5220

Quebec

Granby	514/375-1240
Jonquiere	418/545-2272
Montreal	514/878-0450
Quebec City	418/647-4690
St. Hyacinthe	514/774-9270
Sherbrooke	819/566-2770
Trois Rivier	819/566-2770

New Brunswick

Fredericton	506/454-9462
Moncton	506/854-7078
Saint John	506/693-7078

Nova Scotia

Halifax	902/425-6931 902/539-7010

Prince Edward Island

Charlottetown	902/569-3391

Newfoundland

St. John's	709/726-4920

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.

Racing Information Service

Los Angeles, CA—Racing Information Systems (RIS) is one of the many information providers on the CompuServe Information Service. RIS, based in Hollywood, provides motorsports information to the many CompuServe subscribers through easily accessed menu pages. Point standings in NASCAR, IMSA, SCCA, and Formula One auto racing, information on who to contact for tickets to various motorsports events, sources of free films for sports car clubs, and the Motor Racing News and Notes column written by Michael F. Hollander of RIS are among the information provided..

This column provides insight into the world of motorsports and with the immediacy of the CompuServe connection, subscribers can find out what is happening to auto, motorcycle, boat, and airplane racing days or even weeks before they can read about it in magazines or newspapers. Working with the staff of ON TRACK Magazine, RIS has access to some of the finest motorsports journalists in the country for its databases.

Motor racing is the first sport to have its own database on CompuServe. The database provides more information than the racing fan can get from his local radio or television station. By simply going to page RIS-1 on the information service, the racing fan is presented with a myriad of choices of information in the racing scene. For information on how to subscribe to CompuServe, see your local Radio Shack store or dealer.

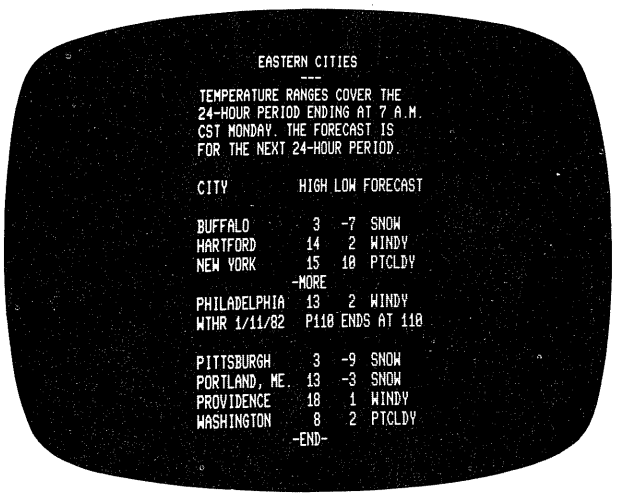
For more information on RIS, send mail to :

Racing Information Systems
2401 Micheltorena Street
Los Angeles, CA 90039
or call (213) 667-1589

Electronic mail may be sent to 70001,557 on CompuServe.

Weather Information on Dow Jones

"Pickles?" (Couple going over "laundry list")
"Check"
"Mustard?"
"Check"
"My down coat?"
"Oh Phil, you will not be needing that. I heard on the 11:00 news last night that game time temperatures will be in the low 50's."
"Madge, you know you turned the set off after Dallas. Besides, we need the weather report for Boston, not New Haven. Let us ask George."
"George? When did he become a weatherman?"
"Thanks to the DOW JONES NEWS/RETRIEVAL® Service, George can furnish us with updated weather information from United Press International."
"Really? Show me how, Phil."
Moments later . . .



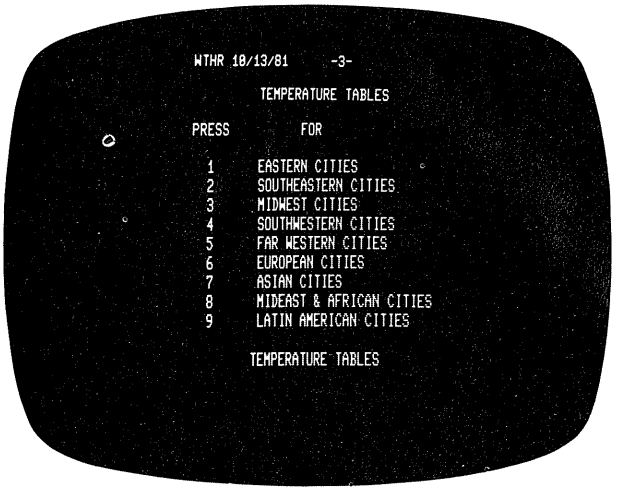
Wondering what to pack for that sales meeting in Seattle? Or the one-day whirlwind tour of the Midwest? An umbrella? A sweater? An overcoat? Dressing for the occasion often has more to do with the weather than it does with fashion. Find out weather conditions fast by checking the new Weather Report data base from the Dow Jones News/Retrieval Service. It is quick! It is easy!

In an effort to bring Radio Shack customers an ever-expanding package of news and information, Dow Jones is happy to announce an exciting new feature — weather conditions and forecasts for at least 50 major cities. The data base includes:

1. Temperature Tables
2. National Weather Forecast
3. National Weather Wrap-Up

WEATHER TABLES FOR THE MOST RECENT 24-HOUR PERIOD

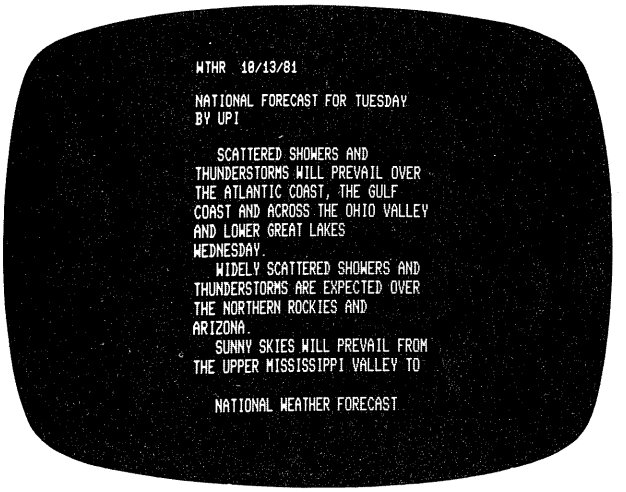
These tables provide an accurate weather picture for the current day for dozens of major U.S. and foreign cities, broken out by geographic region.



Included in the tables are forecasts by city showing the city name, the high and low temperatures from the previous day and the predicted current day weather conditions.

NATIONAL WEATHER FORECAST

This regional summary provides weather predictions for the current day for the United States.

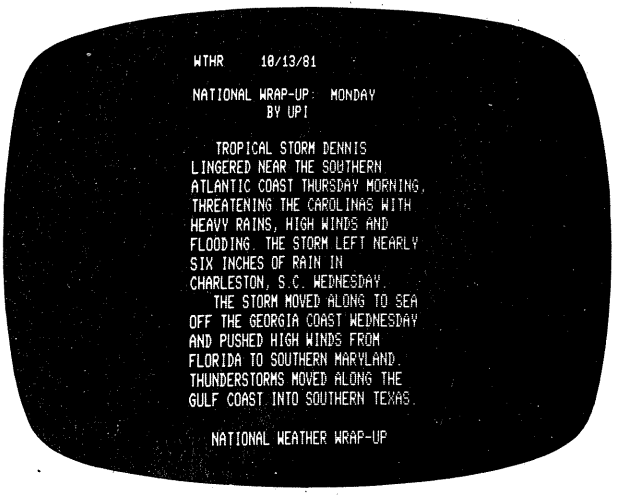


NATIONAL WEATHER WRAP-UP

A summary of the previous day's major weather developments across the United States by geographic region.

All of this valuable weather information, which is furnished by United Press International, is updated twice daily Monday mornings through Friday afternoons.

Be prepared . . . get a fast weather report for your travel destination on the Dow Jones News/Retrieval Service!



THE COST

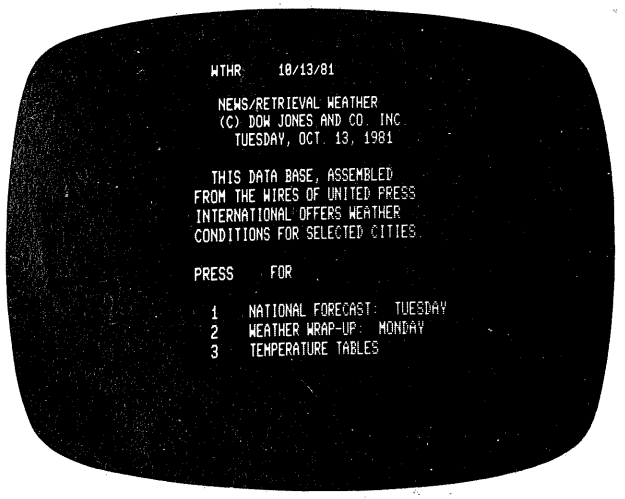
The usage rate for the Weather Report data base is the same as our regular news data. (See your Dow Jones Radio Shack price list for details.)

HOW TO GET THE WEATHER REPORT

To enter this data base on the Dow Jones News/Retrieval Service, type two slashes, WTHR and press the ⟨**ENTER**⟩ key.

/ / WTHR
⟨**ENTER**⟩

Here is what you will see:



Select the information you want by pressing the CODE NUMBER followed by the ⟨**ENTER**⟩ key.

Notes

The following commands will assist you in moving about the data base:

1. To go forward to the next page within a section, simply press the ⟨**ENTER**⟩ key.
2. To go back to the previous page within a section, press R (for Reverse) followed by the ⟨**ENTER**⟩ key.
3. Menu pages are numbered from 0-99. Text pages are numbered beginning with 100. You will find the page number at the top of each page, showing the page you are on and the last page of the section.

   For example, "P103 of 108" means you are on page 103 of text and that page 108 is the end of the section.

   To proceed to any page of the data base from any other page (forward or back), press P (for Page) followed by the page number and the ⟨**ENTER**⟩ key.

   Example: To view page 105, type:

   P 1 0 5
   ⟨**ENTER**⟩
4. To return to a previous sub-menu within the data base, press M (for Menu) and the ⟨**ENTER**⟩ key until the desired menu page is reached.
5. For a shortcut back to the Top Menu, press T (forTop) followed by the ⟨**ENTER**⟩ key.

   Be prepared . . . get a fast weather report for your travel destination on the Dow Jones News/Retrieval Service.

Dow Jones News/Retrieval Service
P.O. Box 300
Princeton, New Jersey 08540
1-800-257-5114 (in NJ, 609-452-1511)

Blinking Prompt and INKEY$

Sal S. Vescera
Melbourne, FL

At times, INKEY$ is good for selecting an option. However, it can provide some operator confusion. I have found that the characters can be made to blink to attract the operator's attention.

Example:

```
100 CLS
110 A$=INKEY$
120 PRINT@ 960,"< >ES OR < >O";
130 FOR I=1 TO 20
    : NEXT I
140 PRINT @ 960,"<Y>ES OR <N>O";
150 IF A$="Y" GOTO 180
160 IF A$<>"N" GOTO 110
170 GOTO 200
180 PRINT
    : PRINT "YOU PICKED <Y>ES"
190 END
200 PRINT
    : PRINT "YOU PICKED <N>O"
```

NOTE: Lines 120 and 160 use "PRINT @ 960, . . . ," which also helps. I use it to direct the operator's eyes to the same line on the screen for input requirements.

New Commands and Functions for VisiCalc®

Make the Program Speedier to Set Up and More Useful.

The VisiCalc program is fast approaching the status of a programming language—for people who want to use a computer, not program it. The enhanced version of the VisiCalc program for the TRS-80 Model III includes logical operators and a number of new commands which make the program far more powerful in handling complex business applications than its earlier version.

At the end of this column, we will have some sample work sheets set up which may spark your imagination on how to use the new commands and functions in your own VisiCalc work sheets.

BREATHING EASY

Ever want to get a printed record of the exact formulas you use at the row and column cells of your VisiCalc model? Now you can, with /SS:P command.

A printed listing of your formulas might look like this:

```
)B3:@SUM(B1 . . . B2)
)B2:/F$832.25
)A2:"ITEM2
)B1:/F$356.95
)A1:"ITEM1
```

As you see, the VisiCalc listing runs from what would be the bottom right hand corner of your work sheet to the upper left hand corner, and supplies you with all the format, immobile title or window information you would need to reenter an entire model from scratch. (Of course, it is a lot easier to reenter it from diskette . . .)

Another reason you can breath easier is because of the new /E (EDIT) command. What /E does is bring the contents of a VisiCalc cell back to the edit line. No longer do you have to retype the contents of an individual cell simply to change one character. If you wish to perform an @SUM from B2 instead of from B3, /E allows changing only the number 3 to a 2.

/E Edit - Allows editing of entry position contents. Places the contents of the highlighted entry position on the edit line. The left and right arrow keys (→ and ←) move the edit cue over the characters without changing them. The ↑ arrow key moves the edit cue to the beginning of the edit line; the ↓ arrow key moves the edit cue to the end of the edit line. Enter or delete to the left of the edit cue. SHIFT-ENTER invokes the Edit Command while entering values and labels.

WHAT IS THE DIF™?

DIF is a program-independent way of storing data on diskette. What this means to the TRS-80 user is that other programs which use the DIF storage format may be able to read their data from the same storage diskette the VisiCalc program uses.

More immediately, it means that you have a way of "consolidating" or combining VisiCalc work sheets—such as summing up the profit & loss statements of several branch offices.

For example, store the "bottom line" (net profit) from each of five branch offices in DIF format. The data from these bottom lines may be brought back into another VisiCalc work sheet—anywhere on that work sheet. If you so choose, you may bring back that row of bottom lines as a column. After you have brought back your set of bottom lines, you may sum them, average them or do any other VisiCalc modeling with the data.

DIF can store rectangular areas of VisiCalc work sheets, too. You may reorient them just as easily.

One thing to remember is that the DIF stores VisiCalc data, not VisiCalc formulas. The numbers — and labels — stored in DIF are stored as they appear on the VisiCalc sheet, not as the formulas which appear at the top of the screen.

VisiCalc is now supplied with the "Programmer's Guide to the DIF," for those technically inclined users who wish to become involved with the fundamentals of programming with DIF.

NEW OPERATORS

How often have you wanted to set up a VisiCalc model and said to yourself, "If A4 is less than 6000, then I would like B4 to be 10000 . . . "? Now you can.

The VisiCalc program now has logical functions (Booleans) and arithmetic comparison operators: TRUE, FALSE, AND, OR, IF and the symbols for greater than ")", less than "(", equal to " = ", not equal to "( )", and combinations such as "greater than or equal to." The new operators make it easy to set up such applications as bonuses for attaining sales figures, breaks in interest rates or to help in figuring buy and sell points for portfolios.

The new logical functions can result in extremely flexible VisiCalc models and the logical functions are an important addition to the growing list of VisiCalc capabilities.

LOGICAL FUNCTIONS

@AND(*list*)	TRUE if all values in *list* are TRUE, otherwise FALSE.
@IF(*l,v1,v2*)	*v1* if *l* is TRUE; *v2* if *l* is FALSE.
@ISERROR(*v*)	TRUE if *v* is ERROR; otherwise FALSE.
@ISNA(*v*)	TRUE if *v* is NA; otherwise FALSE.
@NOT(*l*)	TRUE if *l* is FALSE; FALSE if *l* is TRUE.
@OR(*list*)	TRUE if any value in *list* is TRUE, otherwise FALSE.

@CHOOSE AND @LOOKUP

Another powerful new VisiCalc function is @CHOOSE—definitely nothing to sneeze at. The @CHOOSE function complements the @LOOKUP function, already present in the program. @LOOKUP conducts a search for the closest match to the values in a list, then selects and returns the value found to the right or below its range. @CHOOSE, however, is an index to the elements in a list.

SEARCH FUNCTIONS

@CHOOSE (*v,list*)	Returns the *v*th element of *list*. If *v* is greater than the number of elements in *list*, NA is returned.
@LOOKUP (*v,range*)	Compares *v* to the successive values in *range* and returns the corresponding value from the column or row immediately to the right or below the entries in *range*.

In the argument @CHOOSE(n, list) for example, @CHOOSE simply selects the nth element of the list (if n is a fraction, @CHOOSE truncates downward: 2.99 becomes 2). This means that depending on the value placed in one VisiCalc cell, the program can choose from among a lengthy list of values elsewhere without having to approximate or match a given value.

CONVENIENCE TOO

Two "convenience" features have been included in the program. When you type a /V, the program's version number appears on the edit line, along with the copyright notice.

The other convenience feature is the "soft boot." No longer must you turn off the computer in order to run another program, but now you can type /SQ (Storage Quit) and leave the program to return to TRSDOS.

The new VisiCalc commands and functions dramatically enhance the capabilities of the program.

* * *

Examples of how to use the new VisiCalc features in your work sheets:

A salesman's commission is 10% of sales; but if one month he sells over $10,000, he gets a bonus of $200.

	A	B
1)	Sales	(Your Input)
2)	Commission	(B1 * .1)
3)	Bonus	@IF (B1>10000,200,0)
4)	Income	+B2+B3

.....

Has each department met its production quota?

	A	B	C	D
1)	Section	Quota	Produced	Quota Met
2)				
3)	Nuts	13000	135000	(C3>B3)
4)	Bolts	11550	11500	(C4>B4)
5)	Pins	530	5600	(C5>B5)
6)	Snaps	6600	7600	(C6>B6)

A quota previously set at B3 through B6 and unit production values entered in C3 through C6 would result in a TRUE or FALSE answer in D3 through D6, depending on the success of each section.

.....

A salesman gets a bonus if his sales are over $10,000 or if the number of units he sells is over 35 and his total sales are over $6,000.

	A	B
1)	Bonus	(Your Input)
2)	Units	(Your Input)
3)	Bonus	@IF(@OR(B1>10000, @AND(B2>35,B1>6000)), 200,0)

.....

Given the day of the month, number of the month and year, calculate the number of the day of the year.

	A	B	C
1)	Day =	(Your Input)	0
2)	Month =	(Your Input)	31
3)	Year =	(Your Input)	59
4)			90
5)	Day of the		120
6)	Year	(formula)	151
7)			181
8)			212
9)			243
10)			273
11)			304
12)			334

The formula placed in B6 = @CHOOSE(B2,C1 . . . C12) + B1 + @IF(@AND(B2,@INT (B3/4)*4 = B3), 1,0)

The formula also handles leap years other than centennial leap years.

Printer Codes Revisited

You may remember our marathon article last May which set forth a concept of printer code structure designed to allow for the tremendous growth in printer performance. There was a triple mode standard established to support the various applications to which modern printers are directed. These were the data processing, word processing and graphic modes. Last summer we introduced the first machine designed with these ideas in mind. Line Printer VIII (26-1168) has a data processing (D.P.) mode with code response matching the expectations of D.P. programs. Its word processing (W.P.) mode allows use of the printer with programs such as SCRIPSIT word processors. (Note: SCRIPSIT 2.0 for Model II 26-4531 includes certain code features which allow tailoring to the L.P. VIII). It also responds to bit image graphic commands.

The primary purpose of the article was to establish a code response standard which will be uniform from machine to machine. This is becoming more and more important these days as printers get smarter and programs more complex. One thing we don't want to happen is to have programs married to one printer. Our goal this month is to refine our galaxy of printer codes into a structure which allows programmers to make the best use of our machines without fear of "next year's wonder" causing instant obsolescence!

As much as I hate the proliferation of Roman numerals throughout our catalog, (I personally guarantee that there will be NO Line Printer IX!) the codes are divided into two basic groups: Level I and Level II. Level I codes are those which are common to all printers in each class of Data Processing, Word Processing, and Graphic machines. Level II codes are in some way unique to certain printers and will require some care on the part of the programmer if pitfalls are to be avoided.

There are two points which need to be clarified before we proceed much farther. The code standards are not to be confused with performance specifications. Not every printer need have the capability to respond to every code in the standard. Some machines are intended for less demanding applications and have a limited capability. A printer designed for simple data processing does not have to do the things that a printer meant for word processing has to do. In addition, a multi-mode printer will have responses required by one mode and not needed by another. In addition, a low-cost printer might meet only certain minimum responses, leaving the total set to more expensive machines.

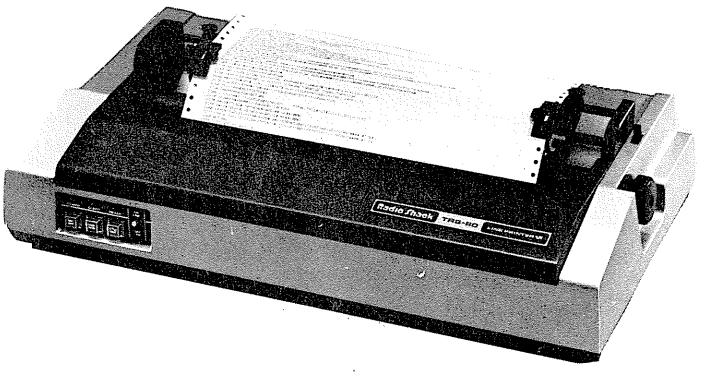
The word processing and data processing modes have two areas of difference. Only one aspect of response is physically different in the two modes; this is the matter of action taken upon receipt of line feed codes. In the data processing mode, the only code which produces a line feed action is the standard LF code (Decimal 10) or the print command CR (Decimal 13), either of which causes a carriage return and line feed. All other line feed control codes set latches in the printer to produce line feeds of various spacings whenever a line feed action is taken by the printer. (Please note that for now questions about the character set are not included in the discussion. We will save that for later. For now we will be talking about the control code responses of printers.)

On the other hand, in the word processing mode each line feed code produces an immediate response in the printer according to the code received. Except for this fact, the only difference between the two modes is a difference in the codes required to be present for each application.

Now, let's get down to the business at hand. Remember Level I codes are common to all printers in each class. If a programmer writes using only these codes he can be assured that his program will work on all Radio Shack printers so designated. Until recently, Dot Matrix printers were intended entirely for data processing functions. All Radio Shack printers of this type include Level I code response. (NOTE: Refer to the May article for a detailed description of discrepancies in some early machines which affect some aspects of line feed response. If the user of these machines runs the utility LPC with his applications, these differences will be transparent to this discussion.)

The following chart outlines Level I Data processing codes:

HEX	DEC	SYMBOL	DESCRIPTION
ØA	1Ø	LF	Execute CR/LF
ØD	13	CR	Execute CR/LF (Print Command)
1B 36	27 54	ESC 6	Select 6 LPI (Power up default)
1B 38	27 56	ESC 8	Select 8 LPI



This list looks rather limiting at first glance, but in fact most D.P. programs require little of the printer but to lay out line after line of "data"—i.e., facts and numbers.

The list of codes for Word Processing is longer because W.P. requires more of the printer. Level I W.P. codes are as follows:

HEX	DEC	SYMBOL	DESCRIPTION
0A	10	LF	Execute 1/6" FWD LF
1B 0A	27 10	ESC LF	Execute 1/6" REV LF
1B 38	27 56	ESC 8	Execute 1/8" FWD LF
1B 1C	27 28	ESC FS	Execute 1/12" FWD LF
1B 1E	27 30	ESC RS	Execute 1/12" REV LF
0D	13	CR	Execute CR/LF
08 n	08 n	BS n	*Backspace (value)
1B n	27 n	ESC n	*Incremental space (1-9)

This list includes the responses needed to allow backspaces, sub- and super-scripts, and simple right justification of texts. All Radio Shack word processing printers (LP IV, VIII, W.P.50, and D.W.II) execute these codes correctly. Only LP IV and VIII actually implement the BS n syntax. For the other printers the value "0" can be used as this null code is not transmitted by printer drivers. (*Please note the asterisks here. This symbol will mark any code which is in some way printer dependent). Properly constructed programs will allow some kind of action to initialize the program to fit the printer in use. This could be in the form of user definable defaults, built-in look-up tables, etc.

Many of the Level II codes require associated values which can depend entirely upon the printer's mechanical features. The number of dot columns in each character sometimes changes from pitch to pitch in the same printer. Values for proportionally spaced characters can also vary from machine to machine.

Level II codes then are those codes which can give associated value differences. Many of them are optional and are not to be found in all machines.

Other machines also bear burdens of the past: There are some conflicts in response in early Radio Shack printers. Codes for these actions are fixed for the future, but unless special care is taken programs written using these codes will not function with some printers sold in the past by Radio Shack. We will always try to identify these areas of conflict and one of the goals of the May Newsletter article was to do just that. Much of that information will bear repeating in this context.

Remember some Level II codes have printer unique associated values. Others are optional because of printer design, type, or application. Others, while uniform for current and future printers are in conflict with past machines. Future printers will follow these standards. If they are built to perform described functions, the code required is established. New functions will use new codes which are currently unassigned. We will do our best to structure printer enhancements so they do not interfere with programs currently being written. Proper and careful use of Level II codes will enable you to build programs which get the most out of the wide array of printers offered by the Shack.

For now, I will discuss the codes grouped according to function. Later I can regroup them all together in number order. Printers which are multi-mode will have to have commands which select this mode of operation.

HEX	DEC	SYMBOL	DESCRIPTION
12	18	DC2	Select graphics mode
13	19	DC3	Select D.P. mode
14	20	DC4	Select W.P. mode

Note that if, for instance you are already in DP mode, the repetition of DC3 is ignored.

The question of pitch selection is problematical. Of course, dot matrix printers can change pitch at will—all under software control. On the other hand a Daisy Wheel printer can completely change pitches only with manual operator intervention (to change the wheel!). Pitch changes can also get you in lots of trouble. A change in letter spacing changes the number of characters in a line (or the line length). This can play havoc with justification schemes or fancy report formats. Unfortunately it is this area which has the greatest number of code conflicts. Used wisely, however, changes in print densities can produce greater impact and reader understanding in a printout.

Daisy Wheel II has three codes to select the pitches which correspond to the wheels first available: 10 CPI, 12 CPI, and PS (proportional space). The printer also has a manual switch to accomplish this which is ignored once the pitch is set by software. Since manual intervention is required to complete any pitch change, most programs will leave pitch changes to the operator.

The pitch change codes for Daisy Wheel printers:

HEX	DEC	DESCRIPTION
1B 0E	27 14	1/16 space set (12 cpi)
1B 0F	27 15	1/10 space set (10 cpi)
1B 11	27 17	PS space set

NOTE: The # sign will be used to signify a code conflict lurking in the past. Refer again to the May printer article.

The pitch change codes for dot matrix printers are:

HEX	DEC	SYMBOL		DESCRIPTION
1B 11	27 17	ESC DC1		Select proportional characters
1B 13	27 19	ESC DC3		Select 10 CPI (Normal matrix) characters
1B 14	27 20	ESC DC4	#	Select condensed characters (normally used to produce 132 col in 8" line)

Additional codes will be added as additional fonts are added in Dot Matrix printers.

One of the great advantages of a dot matrix printer is the great flexibility it offers. One can do all kinds of pitch and font changes so easily! Certain fonts (sets of characters) can easily be manipulated in several ways—elongation, underline, bold, etc. These codes are listed below:

HEX	DEC	SYMBOL		DESCRIPTION
1B 0E	27 14	ESC SO	#	Start elongation
1B 0F	27 15	ESC SI	#	End elongation
1B 1F	27 31	ESC US		Start BOLD
1B 20	27 32	ESC SP		End BOLD
0E	14	SO		End underline
0F	15	SI		Start Underline

Current standards support simple bit image control of dot matrix printers. When in graphic mode, the seven LSB (Least Significant Bits) of 8-bit data words are used to control the firing of seven pins of the head. There are a number of associated commands. (Note: the line feed spacing is automatically set in the printer to accomplish continuous vertical lines.)

HEX	DEC	SYMBOL	DESCRIPTION
0A	10	LF	Forward LF
0D	13	CR	CR/LF
1E	30	RS	End graphics and return to previously established char. status
1C n m	28 n m	FS n m	Repeat data pattern
1B 0E	27 14	ESC SO #	Start elongation
1B 0F	27 15	ESC SI #	End elongation
1B 10 n m	27 16 n m	ECS POS n m	Dot column addressing

There we have it. Lots of codes, lots of printers. In case you got lost, here (in one place) are all the codes which we now support in our printers.

CHART

HEX	DEC	SYMBOL		DESCRIPTION
00	00	NUL		Ignored
01	01	SOH		Ignored
08 n	08 n	BS n	*	Back Space
0A	10	LF		Line Feed
0D	13	CR		Carriage Return (Print Command)
0E	14	SO		End Underline
0F	15	SI		Start Underline
12	18	DC2		Bit Image Set
13	19	DC3		DP Mode Set
14	20	DC4		WP Mode Set
1B 01-09	27 01-09	ESC n	*	Incremental Space
1B 0A	27 10	ESC LF		Full Reverse LF (6 lpi)
1B 0E	27 14	ESC SO	#	Start Elongation
1B 0E	27 14			Set D.W. to 10 cpi
1B 0F	27 15	ESC SI	#	End Elongation
1B 0F	27 15			Set D.W. to 12 cpi
1B 10 n1 n2	27 16 n1 n2	ESC POS n1 n2		Dot Positioning
1B 11	27 17	ESC DC1		Proportional Set (D.M and D.W)
1B 12	27 18	ESC DC2		Reserved for future use
1B 13	27 19	ESC DC3		Standard Set
1B 14	27 20	ESC DC4	#	Condensed Set
1B 15	27 21			Enable C/R only (DWII)
1B 16	27 22			Enable C/R LF (DWII)
1B 18	27 24			Enter EPM (DWII)
1B 19	27 25			Exit EPM (DWII)
1B 1C	27 28	ESC FS		Half Forward LF (12 lpi)
1B 1E	27 30	ESC RS		Half Reverse LF (12 lpi)
1B 1F	27 31	ESC US		Start Bold
1B 20	27 32	ESC SP		End Bold
1B 33	27 51	ESC 3		1/36 Forward LF
1B 36	27 54	ESC 6		Full Forward LF (6 lpi)
1B 38	27 56	ESC 8		3/4 Forward LF (8 lpi)
1C n1 n2	28 n1 n2	FS n1 n2		Repeat Data
1E	30	RS		End Bit Image
7F	127	DEL		Ignore
FF	255	DEL		Ignore
Other 02 - 1F	Other 02 - 31			Invalid Code
Other 80 - 9F	Other 128 - 159			Invalid Code

Line Printer VIII and all future Dot Matrix printers will print a "blob" (▮) upon receipt of an invalid code. Refer to our printer manuals for additional information.

We will do our best to keep Murphy out of our code file. As our printers get smarter, we will just have to run a little harder to stay ahead of them. Good luck . . . and lots of good printouts.

Computer Camp for Diabetic Children

In addition to their regularly scheduled sessions*, the National Computer Camp™ recently announced a one week computer camp designed for children and adolescents with diabetes mellitus to be held in August, 1982, in Simsbury, Connecticut.

CAMP OBJECTIVES

First, the camp is designed to give campers the opportunity to learn computer programming in a setting where the health care needs of the diabetic are assured. Secondly, the learning experiences are designed to evaluate and develop computer programs that illustrate the interaction of exercise, diet composition, and insulin treatment in the diabetic.

MEDICAL SUPERVISION

While diabetic control of the children will be monitored by the methods used by the individual at home, great emphasis will be placed on home blood glucose monitoring. All physical activities at the camp will be supervised by physical education instructors and monitored by the medical staff.

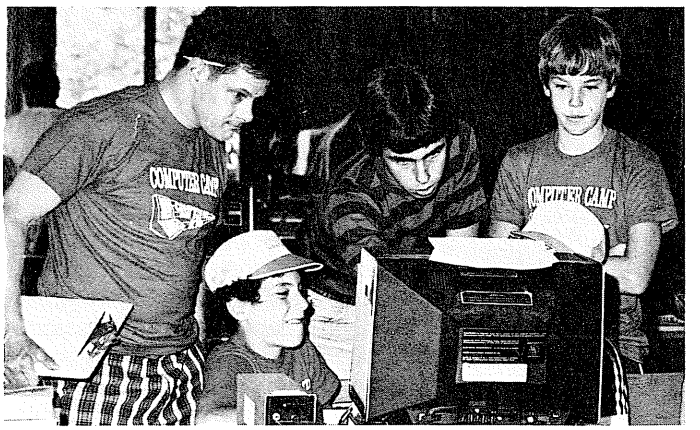
Medical support for the camp will be provided by the members of the division of Pediatric Endocrinology and Diabetes at the University of Connecticut. At least one physician and one nurse will be present at the camp at all times.

Any patient requiring hospitalization or emergency care will be treated at the John Dempsey Hospital which is within ten miles of the camp.

FOR MORE INFORMATION

For educational information contact Dr. Michael Flaks (1-203-795-3049). For Medical Information contact the Division of Pediatric Endocrinology at the University of Connecticut Health Center (1-203-674-2221). In addition, you may also write to National Computer Camp, P.O. Box 624, Orange, CT 06477.

*The regularly scheduled camps in Simsbury, Connecticut and Atlanta, Georgia, will be held July 11 through August 6.



Desktop/Plan™-80

Sophisticated Business Forecasting for TRS-80 Model III

Like its sister software, the VisiCalc program, the recently introduced Desktop/Plan-80 (26-1594) performs business forecasting tasks by allowing the user to set up and execute a financial model.

While the two programs seem to invite surface comparisons, they differ greatly in their approach to the computer and the art of forecasting. Where VisiCalc software is quick and flexible like a word processor, able to perform engineering calculations or statistics in many formats—as well as financial analysis—Desktop/Plan-80 software is deliberate, highly structured and specifically designed for financial modeling.

Desktop/Plan-80 is a "menu-driven" program which performs its operations on files of information, much in the manner of the time-shared financial analysis programs on which it was modeled. As such, it provides for consolidation of financial models, transferring individual line items from model to model and the ability to incorporate in its models calculations programmed by the user. The program may receive (receive only) data from a /PD VisiCalc file for printout in Desktop/Plan-80 format.

A TOUR OF THE PROGRAM

Financial modeling software takes a minimum of known information—costs, rates of change and the like—applies calculations to that information in an orderly way, and arrives at conclusions about the results of those calculations which can then be printed out in a report format.

Because Desktop/Plan-80 software makes extensive use of the system's printer, the first step after booting the program is to configure the system parameters—how many characters across printer, length of the paper you are using, the name you have given the financial model you are working with and the footer message you wish printed at the bottom of each page of your model.

The program's menu presents you with a choice of tasks: build the description file of your model, enter your initial planning values, enter the calculations and where you wish them performed, "execute" the model you have constructed or print the values which have been calculated by that execution.

Each of those tasks (except printing) creates a file on your data disk. Remember that the Desktop/Plan-80 program performs its calculations on files of information. Because of this, you can create a number of different planning value files, or different calculation files and "mix-and-match" them at execution time.

MODEL DESCRIPTION

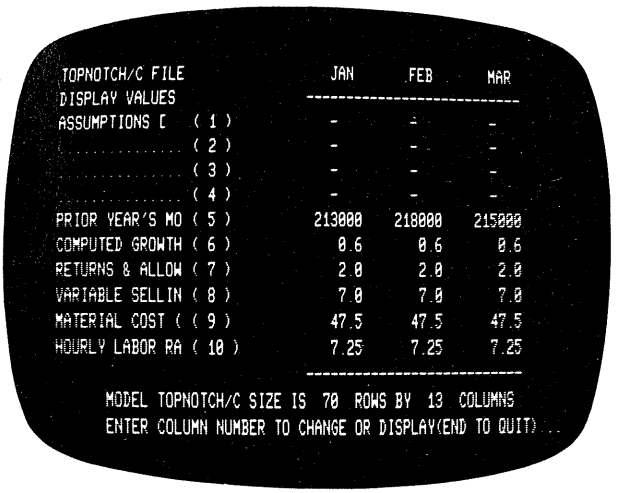
The Desktop/Plan-80 program produces what is known as a "vertical model"—a maximum of 100 line items (100 rows) by a maximum of 18 columns of figures. The 18 columns allow the representation of twelve months, four quarter summations, a percent of total column (or semi-annual total) and an annual total.

You build your model description by telling the program how many rows and columns you intend to use. This is important information for two reasons: only identically-sized models may be later consolidated, and because when finally printed, the printing routine "ignores" those rows of the model which do not have a row description. By allowing gaps between the rows you leave yourself room to perform involved step-by-step calculations which can later be traced step-by-step.

Also during the description process, you will write up to two lines of column headings, decide on the number of decimal places (0, 1, or 2) and title your report (Jones Company, Confidential Sales Projection).

ENTERING PLANNING VALUES

Once the model description is complete, you may enter



your initial planning values—assumptions you make for your model. For example, in a sales projection, you may assume that sales are 100,000 units for January, that draw against commission is 20% of total sales or that rent for facilities is a fixed cost no matter what the month.

To make planning value entry easier, ten rows and three columns of your model are displayed at a time on the screen of the Model III, along with the left-hand titles of each row.

When finished, your initial planning values are also filed away on disk.

THE CALCULATION RULES

Once your model description and planning values are filed away, you may wish to print them out as a reminder of which items are supposed to appear on which rows. Your next step is to assign calculation rules.

In the Desktop/Plan-80 program, calculations are first written to a calculation rules file and later performed on entire rows or entire columns at a time—rather than instantaneously and on screen, as in the VisiCalc program. While you may reference an individual row and column intersection, this is done only by writing a "custom calculation rule" as provided for in the program. We will get to that in a moment.

You select your calculations from among 20 calculation rules supplied by the program. Such calculations as summation of a column, subtracting one row from another, multiplying one column by another and replacing the result in a third column; these are the types of rules provided by the program.

As stated above, "blank" areas of your model will not print in the final report. However, because the calculation rules may be printed by themselves, you can fill those blank areas with involved calculations and then retrieve them for later study.

One of the more powerful features of the Desktop/Plan-80 program is the "custom calculation rules." These custom rules are for the sophisticated user of the program. They are actually BASIC language subroutines which are written by the user and later called by the Desktop/Plan-80 program. Even machine language routines may be written and called by a "CALL USR" custom rule.

When finished, the calculation rules file will contain the formulas by which the initial values of your model are transformed into the computed values of the final report. More than one calculation rules file may be constructed, allowing for later "mix-and-match".

EXECUTE AND PRINT YOUR MODEL

When model description, calculation rules and initial planning values are determined to your satisfaction, it is time to execute your model.

Execution uses the three files which you have previously created and produces a fourth: the Computed Values File. If you have prepared a series of rules files or planning values files, you may mix-and-match to create a number of computed values files.

If you have a series of models which are identical—for example, the Profit & Loss statements with identical line items for a number of branch offices—you may consolidate them into a master company-wide model. If you have the departmental budgets for various departments in your corporation— no two of which are precisely the same, as is often the case—you may select individual line items from those models and combine them into a company budget. Consolidation and summarization are powerful features of Desktop/Plan-80.

Printing the computed values file of your model produces a professional-quality report which automatically configures itself to the size of your printer. Options for printing include use of single page or fanfold paper, and number of lines per page (usually 66 on 8½" by 11" paper).

Each page is automatically numbered, displays up to 80 characters of title, can display a page footer and will print with or without row numbers.

GRAPHICS

Report printing is not all of the Desktop/Plan-80 program's printing capabilities. On a Radio Shack printer, it will also print out an X-Y graph from any two line items. The printing is performed using ASCII characters, so no special graphics are required.

SUMMARY

Desktop/Plan-80 software is a self-contained financial forecasting package which offers a structured approach to financial modeling, and will particularly appeal to those who have had experience with modeling on a time-sharing computer and who wish to continue with that same type of time-shared format.

The Desktop/Plan-80 program does not have the immediate flexibility of the VisiCalc program. However, it is especially useful in situations where whole sub-models need to be consolidated into master models, or where the same model may be executed over and over. The program's graphics facility and its ability to accept a computed value file from a VisiCalc model make it useful when used in conjunction with that program. Desktop/Plan-80 is available for $199.00 for any 48K Model III with at least one disk and a printer.

Model I/III Bugs, Errors and Fixes

GENERAL LEDGER (26-1552)

In version 1.1 of General Ledger for both the Model I and III, the program hangs up when attempting to post transactions.

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 "GLTXPOST".
3. Make the following corrections:

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

Old Line:
```
285 CLOSE2:OPEN"R",3,"DETAIL:1"
    :OPEN"R",2,"GLJOUR:Ø":CLOSE2
```

New Line:
```
285 CLOSE2:OPEN"R",3,"DETAIL:1"
    :OPEN"R",2,"GLJOUR:Ø"
    :ME=LOF(2)*6
```

```
Old Line:  3Ø1 ME=1:ONERRORGOTO3Ø4
New Line:  3Ø1 IFME>2ØØTHEN3Ø5ELSENE=1
               :ME=Ø:ONERRORGOTO3Ø4

Old Line:  3Ø2 PUT2,ME+1Ø:GOTO3Ø2
New Line:  3Ø2 PUT2,NE:NE=NE+1:ME=ME+6
               :IFNE<1ØØTHEN3Ø2ELSE3Ø5

Old Line:  3Ø4 IFERR<>122THENRESUME:ELSE
               ME=(ME-3Ø)*6:RESUME3Ø5
New Line:  3Ø4 IFERR<>122THEN99ØØELSERESUME3Ø5

Old Line:  99ØØ RESUME
New Line:  99ØØ ER=ER+1:IFER<5THENRESUMEELSEPRINT@965,
               "ERROR"ERR/2+1"IN LINE"ERL"-PRESS <ENTER>
               TO RESTART";:FL=1:GOSUB1ØØ:PRINT@96Ø,
               CHR$(31);:GOTO1Ø

DELETE LINE 29Ø
```

4. Type SAVE "GLTXPOST" to save program changes.
5. At TRSDOS READY make a backup of the corrected diskette.

ACCOUNTS RECEIVABLE (26-1555)

Below are four corrections to the Accounts Receivable program. The first is to recover a crashed index. The second eliminates Error Code 5 in line 2500. The third eliminates an Error code 2 in line 1590 (version 3.0) or in 1950 (version 1.0). The fourth correction fixes a problem with correction entries.

1 - RECOVERING A CRASHED INDEX (ALL VERSIONS)

If any of the following symptoms occur, the index has been crashed and the following Recovery program should be run. (These corrections replace those published in the September 1980 Newsletter.)

Unexpected Error Code 64, Line 2850 ("ARS")
Unexpected Error Code 64, Line 1430 ("ARS")
Bad Record Number on program line with "GET" or "PUT" statement

When sorting, the statement "duplicate account # 0 exists" appears or transactions appear under the wrong account—i.e. some transactions for account 3 are on account 20's bill, etc.

In order to correct the above errors, please run the following program written to reconstruct "CUSINDEX" and "TRANSACT" files of Model I Accounts Receivable program package:

Quicker routines can be written, but this was written with the customer in mind. This routine will not recapture report numbers nor session numbers. It is merely intended to aid in recapturing data files rendered inaccessible because the Customer Index or Transaction files have been damaged or scrambled.

*****This is a last resort. All transactions must be checked afterwards and verified. This program will pick up **all** transactions on the disk, including deleted transactions and previous period transactions not yet over-written in the current period.

1. In BASIC, type the following program with each line followed by an ⟨**ENTER**⟩

```
1 ' *** RECOVER *** (C) 1981 TANDY CORP
2 ' *** RECOVERS POINTERS/INDEX FOR ACCTS. RECEIVABLE
3 ' PROGRAM DISK SHOULD BE IN DRIVE Ø, DATA DISK IN DRIVE 1
    --- ON 3 DRIVE SYSTEMS, SECOND DATA DISK IN DRIVE 2.
4 ' RECOVER WILL RUN SORT AND RETURN TO ARS ON COMPLETION.
5 ' **** FOR VERSION PRIOR TO 3.Ø, LINE 21Ø MUST BE
    MODIFIED: CHANGE "JD*128ASD$" TO "JD*127ASD$" AND
    "5ASDE$" TO "4ASDE$"
1Ø CLEAR5ØØ:DEFINTA-Z:CLS:INPUT"PASSWORD";P$:INPUT"2 OR 3
    DRIVE SYSTEM";S
2Ø GOSUB22Ø:OPEN"R",1,PT$:NT=LOF(1)*8:OPEN"R",2,PD$
    :NA=LOF(2)*2
3Ø DIMTR(NT,2),AC(NA,2):CLS:PRINT@154,"FIRST
    PASS":PRINT@4Ø2,"WORKING ON TRANSACTION #"
4Ø KK=NT:FORK=1TONT:GOSUB18Ø:PRINT@426,K:TR(K,Ø)=CVI(V1$)
    :IFVØ$<"A" ORVØ$>"Z"THENKK=K:K=NT
5Ø NEXTK:PRINT@154,"SECOND PASS":PRINT@426,
    "      ":UN=KK:UT=KK-1:U=UT:FORK=1TOUT
    :IFTR(K,Ø)<ØTHENTR(K,Ø)=-TR(K,Ø):U=U-1
6Ø PRINT@426,K:NEXT:CLS:PRINT@154,"FIRST
    PASS":PRINT@4Ø4,"WORKING ON ACCOUNT #"
7Ø FORJ=1TONA:GOSUB2ØØ:PRINT@424,J:AC(J,Ø)=CVI(DD$)
    :IFAC(J,Ø)<1THENF=F+1 ELSECV#=CV#+CVD(DB$)
8Ø NEXT:TN=J-1:TI=TN-F:JJ=J-1:PRINT@154,"SECOND
    PASS":PRINT@424,"    ": UX=UT+(TR(UT,Ø)=1)
9Ø FORJ=1TOJJ:PRINT@424,J:PT=Ø:FT=Ø:AC=AC(J,Ø):FORK=1TOUX
    :IFTR(K,Ø)=ACTHENTR(PT,2)=K:TR(K,1)=PT:PT=K:IFFT=ØTHEN
    FT=K
1ØØ NEXTK:AC(J,1)=FT:AC(J,2)=PT:NEXTJ:CLS:PRINT@4Ø2,
    "RESETTING TRANSACTION POINTERS"
11Ø FORK=1TOUT:GOSUB18Ø:LSETV8$=MKI$(TR(K,1))
    :LSETV9$=MKI$(TR(K,2)): PUT1,KR:NEXT:CLOSE1
12Ø CLS:PRINT@4Ø4,"RESETTING ACCOUNT POINTERS"
13Ø FORJ=1TOTN:GOSUB2ØØ:LSETDI$=MKI$(AC(J,1))
    :LSETDJ$=MKI$(AC(J,2)): PUT2,JR:NEXT:CLOSE2
14Ø CLS:PRINT@41Ø,"WRITING INDEX"
    :OPEN"O",1,"CUSINDEX."+P$+":1":PRINT#1,TI;
    TN;F;U;UT;UN;Ø;1;1;1;1;1;1;1;1;STR$(CV#
    )+"DØ"
15Ø FORN=1TOTN-1:CN=N:IFAC(N,Ø)<1THENCN=-CN
16Ø PRINT#1,CN;CN;CN:NEXT:CLOSE1:OPEN"O",1,"TRANSFER:1"
    :PRINT#1,PI$:PRINT#1,PD$:PRINT#1,PS$:PRINT#1,PT$
    :PRINT#1,PG$:CLOSE
17Ø CLS:PRINT@41Ø,"SORTING DATA":CLEAR5Ø:RUN"ARSORT"
18Ø KR=INT((K-1)/8)+1:KD=K-8*INT((K-1)/8)-1
19Ø FIELD1,KD*31ASVV$,1ASVØ$,23ASV1$,2ASV8$,2ASV9$,3ASVX$
    :GET1,KR:RETURN
2ØØ JR=INT((J-1)/2)+1:JD=J-2*INT((J-1)/2)-1
21Ø FIELD2,JD*128ASD$,1Ø5ASDØ$,8ASDB$,4ASDC$,2ASDD$,5ASDE$,
    2ASDI$,2ASDJ$: GET2,JR:RETURN
22Ø PT$="TRANSACT."+P$+":"+CHR$(47+S):P$=P$+":1":PI$=
    "CUSINDEX."+P$:PD$="CUSDATA."+P$:PS$="CUSSETUP."+P$
    :PG$="GLFILE."+P$:RETURN
```

2. Type SAVE "RECOVERY" to save the program for later use.
3. Type RUN and press ⟨**ENTER**⟩ to start the recovery process.
4. If the program is ever needed again load it from the disk with LOAD "RECOVERY".

2 - ERROR CODE 5 IN LINE 2500 (MODEL I VERSIONS PRIOR TO 3.0)

If an Error Code 5 in line #2500 occurs there is a customer with more than 256 transactions posted.

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 "ARS".
3. Make the following corrections:

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

```
Old Line:  254Ø LSETDF$=ST$:LSETDG$=FC$:LSET
                DH$=TC$:LSETDI$=MKI$(DI):LSET
                DJ$=MKI$(DJ):PUT2,JR:RETURN

New Line:  254Ø LSETDF$=ST$:LSETDG$=FC$:LSET
                DH$=CHR$(TC-(255-TC)*(TC>255))
                :LSETDI$=MKI$(DI):LSETDJ$=
                MKI$(DJ):PUT2,JR:RETURN
```

4. Type SAVE "ARS" to save the changes in the program.

5. At TRSDOS Ready, make a backup copy of the corrected diskette.

3 - ERROR CODE 2 ON MODEL I/III

An Error code 2 may be encountered in line 1590 on version 3.0 or in line 1950 in version 1.0

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 "ARS".
3. Make the following corrections:

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

```
Old Line: 159Ø GOSUB146Ø:IFKS=ØANDVA$<>"D"THEN
          RM$="INVALID ID#-NOT POSTED""

New Line: 159Ø GOSUB146Ø:IFKS=ØANDVA$<>"D"THEN
          RM$="INVALID ID#-NOT POSTED"
```

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

NOTE

This is line number 1950 in version 1.0

```
New Line: 195Ø GOSUB178Ø:IFKS=ØANDVA$<>"D"THEN
          RM$="INVALID ID#-NOT POSTED"
```

4 - CORRECTION ENTRY TO A PAYMENT ERROR

When a correction entry is made to a payment, it replaces the previous payment on the statement (all versions of Model I/III A/R).

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 "ARS"
3. Make the following corrections:

****NOTE: On versions prior to 3.0, this is line 1650.

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

```
Old Line: 2Ø2Ø PR#=PR#-VE#:CB#=CB#+VE#:CV#=CV#
          +VE#:G#(VJ)=G#(VJ)-VE#:G#(2)=
          G#(2)-VE#

New Line: 2Ø2Ø PR#=PR#-VE#:CB#=CB#-VE#:CV#=
          CV#-VE#:G#(VJ)=G#(VJ)-VE#
          :G#(2)=G#(2)-VE#
```

4. Type SAVE "ARS" to save the changes in the program.
5. At TRSDOS READY make a backup copy of the corrected program.

DISK PAYROLL (26-1556)

There are two corrections for Disk Payroll for the Model I. The first corrects a problem which occurs in versions prior to 2.0 and the second corrects a problem which occurs in version 2.0.

VERSIONS PRIOR TO 2.0

The following corrections replace those published in the September 1980 Newsletter.

A customer cannot change weeks worked or workmen's comp in the personnel information section for employees.

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"PR4ADD"
3. Make the following corrections:

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

```
Old Line: 39ØØ FORI=Z1TOZ2:MID$(G$,I*8-7,8)=
          MKD$(E#(I-Z3)):NEXTI:LSETGG$=G$
          :PUT4,1+2*(N-1)+INT((Z-2)/2):GOTO81Ø

New Line: 39ØØ FORI=Z1TOZ2:MID$(G$,I*8-7,8)=
          MKD$(E#(I-Z3)):NEXTI:LSETGG$=G$
          :PUT4,1+2*(N-1)+INT((Z-2)/2):LSETFF$=N$
          :PUT#3,N:GOTO81Ø
```

4. Type SAVE"PR4ADD" to save the changes in the program.
5. Make a backup copy of the corrected diskette.

VERSION 2.0 CORRECTIONS

In Model I Disk Payroll version 2.0, if the city withholding limit is set to a percentage of gross earnings, city withholding continues to be calculated even after the year-to-date earnings for a particular employee have passed the limit.

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"PR4INPUT"
3. Type DELETE 6200-6290 ⟨**ENTER**⟩
4. Type 6290 REM ⟨**ENTER**⟩
5. Now enter the following lines.

ADDITIONS (Type the line followed by ⟨**ENTER**⟩)

```
144 IFL2<ØTHENL2=999999:G2#=L2ELSEG2#=L2*W5/1ØØ#
62ØØ IFZ=1THEN65ØØELSEE#(I)=Ø:IFMID$(N$,25Ø,1)<>"Y"ORK#<1
     THEN65ØØ
621Ø G$=MID$(E$,188,7):GOSUB69ØØ:G#=(G#*K#-CVS(MID$(E1$,241,
     4)))/K#
622Ø IFG#<.Ø1THEN65ØØELSEGOSUB627Ø:GG$=MID$(E$,187,1)
623Ø IFGG$="D"THENE#(I)=G#*W5/1ØØ#:GOTO625Ø
624Ø IFGG$<>"P"THEN65ØØELSEE#(I)=E#(9)*W5/1ØØ#
625Ø IFE#(I)+E4#(I)>G2#THENE#(I)=G2#-E4#(I)
626Ø IFE#(I)<ØTHENE#(I)=Ø:GOTO65ØØELSE65ØØ
627Ø G1#=Ø:FORJ=1TO7:IFMID$(G$,J,1)="1"THENG1#=G1#+E4#(J)
628Ø NEXTJ:RETURN
```

6. Type SAVE"PR4INPUT" to save the changes to the program.
7. At TRSDOS READY make a backup copy of the corrected program.
8. In BASIC, LOAD"PR4JNL".
9. Type LIST 200-205.

Somewhere in lines 200 to 205 (depending on the version), there is a line which contains CW% = CVI(MID$(S$ . . . .etc.

This should be CT%, not CW%.

MICROFILES (26-1565)

As you start to fill up your Microfiles database, it can sometimes be inconvenient and distracting to wait for the program to decide when to create a new extension. This procedure will allow you to create as many new extents as you wish, at your convenience.

The secret to this procedure is in how Microfiles knows how many extents are in the database. Microfiles stores the number of the highest extent that exists (plus 128) in the fourth byte of the second sector of the first extent (FILE/V0).

As you already know from using the program, Microfiles will create an extension of the database if it does not find FILE/V0.

Given this information, an additional extent can be created as follows:

1. Under DOS, RENAME FILE/V0 to FILE/X so Microfiles cannot find it.
2. Run Microfiles. It will ask you to specify a drive number where you want to create the extent.
3. When Microfiles finishes and says, "OK," type DONE.
4. Under DOS RENAME the newly created FILE/V0 to make it the next higher extent that you require. (If you already had FILE/V0 and FILE/V1, RENAME it FILE/V2.)
5. You can repeat steps 2-4 and create as many new extents as you need (up to FILE/V9).
6. Once you have created the new extents, RENAME FILE/X (your original FILE/V0) back to FILE/V0.
7. Finally, you must let Microfiles know that the new extents exist. In BASIC, key in the following program and run it.

```
10 OPEN"R",1,"FILE/V0"
20 FIELD#1,3 AS A$, 1 AS B$
30 GET 1,2
40 LSET B$=CHR$(128+?)
50 PUT 1,2
60 CLOSE#1
```

The question mark in line 40 must be replaced with the number of the highest extent that you have created (If FILE/V2 is the highest, use 2).

After following this procedure, run Microfiles and check your available space by typing FREE.

MODEL III DISK COURSE (26-2014)

In the Lesson 1 section which explains the additional editing features in Model III Disk BASIC, there is an error. SHIFT DOWN ARROW does not list the last line of the program as stated. SHIFT DOWN ARROW Z will list the last program line.

MODEL III DISK SYSTEM OWNER'S MANUAL (26-2111)

On page 124 under E for extending sequential output starting at the end of file, the manual incorrectly states "If the file is not found, it will be created." It should say instead, "The file must already exist."

Notes on Previous Newsletters

November 1981

COLOR DISASSEMBLER

Thomas Rokicki
Box 258
College Station, TX 77841

There is a small bug in Line 250 of the Color Dissassembler. It presently reads:

```
250 IF 16 AND (PEEK(PI+TE)) THEN OP$="(" + OP$ + ")"
    : RETURN ELSE RETURN
```

It must be changed to:

```
250 IF 16 AND PEEK(PI+TE+1) THEN OP$="[" +OP$ + "]"
    : RETURN ELSE RETURN
```

I apologize for this small bug, but I wrote this program in an hour on a bus and did not test it as extensively as I should have.

August 1981

CHECKER BOARD

Larry Smith
34825 Lakeview Dr.
Solon, OH 44139

With the correction to the August 81 Color Computer Checker Board program, I finally got the game playing. During play we could not read the "T" in the upper left box. To get the program to show the "T," we had to change the program as follows:

Delete the original line 825 and enter the following 825:

```
825 DRAW"C2; BM16,22; U20; L10; R20; C4"
```

Looking forward to more programs.

We also received the following additional change to the CHECKER BOARD program.

Matt Petrich, Age 12
1629 E.L.K. Sammamish Rd. North
Redmond, WA 98052

We have made some changes in the Checker Board program by William Cotton, Volume 3, Issue 8 of the Microcomputer News. We made changes to allow the program to run because the listing was incorrect. We also changed the colors of the checkers so people with black & white T.V.s could tell the checkers apart. We changed the 'D' to the space bar to make it easier to play.

The changes in the listing are as follows:

```
137 IF Y<84 THEN PAINT(X+1,Y+1),2,1
138 IF Y>108 THEN PAINT(X+1,Y+6),3,1
```

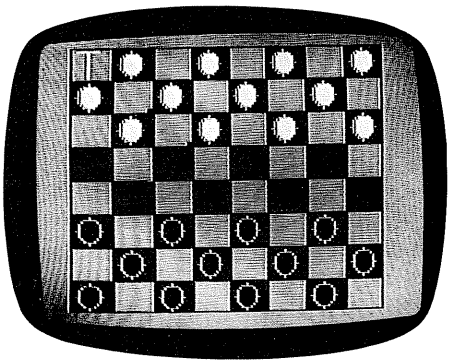
The above corrections allow the program to run as intended by Mr. Cotton.

The change below increases the contrast between the "red" and "black" checkers so you can play the game on a black & white T.V.:

```
137 IF Y<84 THEN PAINT(X+1,Y+1),1,1
```

The next change allows the use of the space bar instead of 'D' to begin the game.

```
180 IF A$=" " THEN 190
```

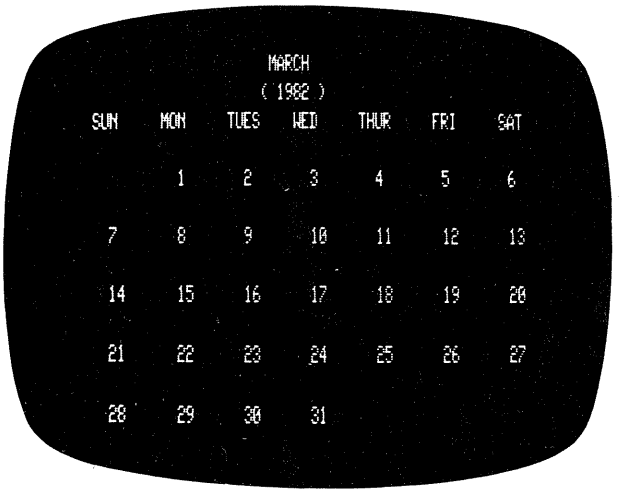


Level I Calendar

Dave Goldman
Fort Worth, Texas

Calendar will display a calendar for a given month and year, or by inputting a date and year, it will display the day of the week on which the input date occurs.

```
1 A(1)=31
      : A(2)=28
      : A(3)=31
      : A(4)=30
      : A(5)=31
      : A(6)=30
2 A(7)=31
      : A(8)=31
      : A(9)=30
      : A(10)=31
      : A(11)=30
      : A(12)=31
3 REM ***FOR COMMENTS, SEE LINE 1000***
4 D=1
      : C=2
5 CLS
      : INPUT "'D'AY OF WEEK OR 'C'ALENDAR";Q
6 IF Q=2 THEN INPUT "MM,YY";M,Y
      : D=1
      : GOSUB 300
      : GOTO 11
7 IF Q=1 THEN INPUT "MM,DD,YY";M,D,Y
      : GOSUB 300
      : GOTO 600
8 GOTO 4
11 CLS
12 L=A(M)
      : IF (M=2)*((INT(Y/4))*4=Y) THEN L=29
13 P=T
      : K=257
14 RESTORE
      : FOR I=1 TO M
      : READ A$
      : NEXT I
15 PRINT AT 28;A$
      : PRINT
16 PRINT "          SUN     MON     TUES    WED
     THUR    FRI     SAT"
17 PRINT AT 92;"(";1900+Y;")"
20 FOR I=1 TO L
24 IF T>7 THEN T=1
      : K=K+128
25 J=K+T*8
30 PRINT AT J;I;
35 T=T+1
40 NEXT I
50 GOTO 5000
70 DATA " JANUARY"," FEBRUARY"," MARCH","
     APRIL"," MAY"
71 DATA " JUNE"," JULY"," AUGUST"," SEPTEMBER","
     OCTOBER"
80 DATA " NOVEMBER"," DECEMBER"
90 DATA SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,
     FRIDAY,SATURDAY
300 IF(D>A(M))+(D<1) THEN 5
305 A=INT(Y/12)
310 B=Y-A*12
320 C=INT(B/4)
330 T=A+B+C+D
340 ON M GOTO 400, 370, 370, 410, 390, 360, 410,
     380, 350, 400, 370, 350
350 T=T+1
360 T=T+1
370 T=T+1
380 T=T+1
390 T=T+1
400 T=T+1
410 IF T>7 THEN T=T-7
420 IF T>7 THEN 410
430 RETURN
600 RESTORE
      : FOR I=1 TO M
      : READ A$
      : NEXT I
610 FOR I=M+1 TO 12+T
      : READ B$
      : NEXT I
620 CLS
630 PRINT AT 210, A$ ;D ;1900+Y ;" IS A"B$
999 REM *****
1000 REM *** CALENDAR-BY D.A. GOLDMAN VER.3.0 ***
1002 REM *** VALID FOR ALL DATES 1909-1999 ***
1003 REM *** ACCURACY OF 1900 UNSURE. CHANGE ***
1004 REM *** TO ACCOMMODATE 1800-2099 SHOULD ***
1005 REM *** ONLY INVOLVE A CONSTANT(+-1 OR 2)
     ***
1006 REM *** NOTE: TO CONTINUE PROGRAM AFTER ***
1007 REM *** EACH DAY OR CALENDAR, JUST TOUCH ***
1008 REM *** ANY KEY. PROGRAM WILL LOOP TO TOP
     ***
1009 REM *** DATA: MONTH- M 0<MM<12 ***
1010 REM *** DAY - DD 0<DD<A(M) ***
1011 REM *** YEAR- YY 0<YY<99
1012 REM *** DAY IS NOT INPUT TO CALENDAR ***
1013 REM *** ROUTINE. YEAR IS INPUT AS Y-1900 ***
1014 REM *****
5000 PRINT AT 1022;" ";
5005 SET(127,47)
5010 IF POINT(127,47) THEN 5010
5015 GOTO 4
```



Secondary Problem-Solving Programs

If you are a regular reader of the Education section, you've recently seen discussions of elementary and secondary math programs that build skill in mathematical foundations by leading students through sequences of drill-and-practice lessons of progressive difficulty. This month's Education article will introduce you to Radio Shack computer assisted instruction programs of a different type for the secondary physics or upper-level math student. Like the drill-and-practice programs, these programs are designed to complement the school's standard curriculum, but they differ from the drill programs in structure and purpose. These are problem-solving programs, or programs whose main purpose is to provide an opportunity for exploration within the framework of certain mathematical or scientific concepts.

Four of Radio Shack's math and physics programs are: Advanced Graphics, Interpreting Graphs in Physics, Vector Addition, and Graphical Analysis of Experimental Data. They were designed and programmed by Richard Born, a professional educator with fourteen years of experience in teaching high school physics, mathematics, and computer science. Mr. Born is now a consultant for Radio Shack. This month we'll take a look at the programs Advanced Graphics and Vector Addition. In the April issue, we'll examine the programs
Interpreting Graphs in Physics and Graphical Analysis of Experimental Data.

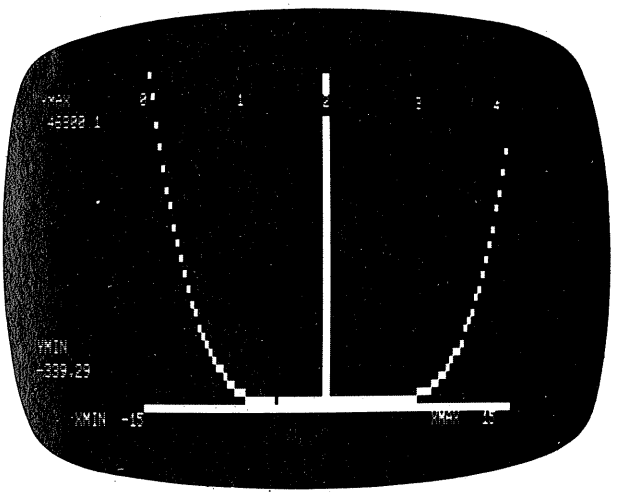
When the TRS-80 microcomputer was introduced in Fall, 1977, Born was teaching a senior-level math class that required students to do much plotting of equations. Born saw the microcomputer as a tool that could be used to plot equations quickly and thus act as a source of motivation and enrichment for his math and physics class, and so he ordered a TRS-80 in February 1978. Advanced Graphics was the first courseware program Born wrote for the TRS-80.

ADVANCED GRAPHICS

Advanced Graphics actually contains two programs: Plotting Functions, and Plotting Polar and Parametric Equations. Plotting Functions comes with an equation built into a line of the program:

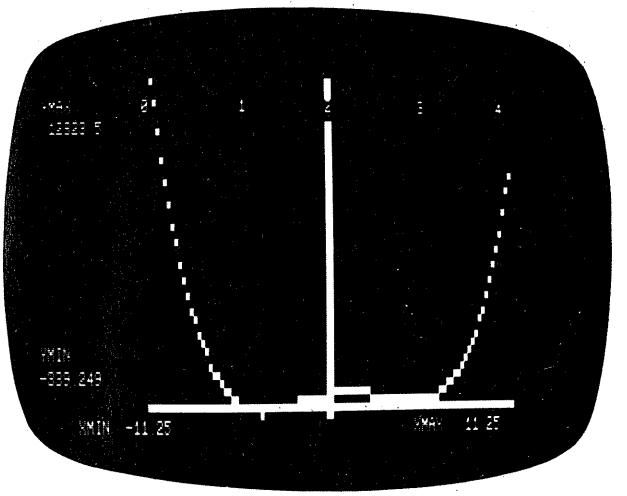
$$(y = x^4\text{-}2x^3\text{-}43x^2 + 80x + 300)$$

but by following a few quick steps the student or teacher can change the equation as many times as desired. Once the equation to be used has been stored in the program, the student simply enters the left and right endpoints of the domain, and the computer prepares and displays the graph. For example, a graph like the following would be displayed if the original equation were used and the endpoints -15 and 15 were selected:



Once the computer has displayed the graph, three options are presented. The student can change the domain, explode the graph, or redraw the graph:

To explode the graph, the student would simply type ⟨E⟩ to select the "explode" option and would then enter the new endpoints. After several seconds, the exploded region of the graph would appear on the video screen. For example, the graph of the original equation exploded between .5 and 3.5 would be displayed as follows:



The second Advanced Graphics program, Plotting Polar and Parametric Equations, works in a similar way to allow the student to quickly plot the graph of equations in the polar form $r = f(\theta)$, where r and θ are the polar coordinates, or to graph

sets of parametric equations of the form $x = f(t)$ and $y = f(t)$, where x and y are the Cartesian coordinates and t is the parameter that is taken as the independent variable.

Born designed Advanced Graphics not as a crutch that does the students' work for them, but a supplementary tool for motivation and enrichment. He found that his students were motivated by having the opportunity to check their answers on the computer after using traditional methods to arrive at the answer. Born has also used the Advanced Graphics program to introduce some of the complex curves, spirals, and figures that students do not usually see until they take college math courses. Born says that Advanced Graphics provides a way to illustrate "the beauty of the mathematics involved" without allowing that beauty to be "lost in all of the calculations".

Born's philosophy has been that a teacher can best use a program like Advanced Graphics when the program is accompanied by related exercises and investigations. Consequently, the Advanced Graphics manual contains a Selected Investigations section for students who wish to use the computer to explore special aspects of functions.

The Selected Investigations section for Plotting Functions includes exercises in such areas as determining the real zeroes of functions, finding relative maxima and minima of functions, graphing functions with discontinuities, building square waves, and determining the horizontal range of projectiles fired at different angles.

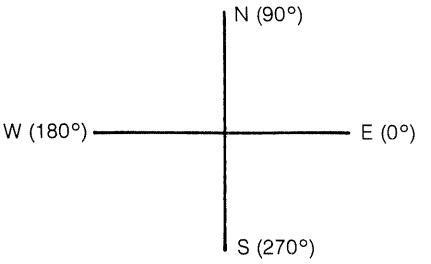
Some of the Selected Investigations included for Plotting Polar and Parametric Equations involve graphing Lissajous figures, rose curves, and several kinds of spirals. The Advanced Graphics manual also includes a Programming Guide which documents the structure of both programs, and an Answer Key which provides answers to the problems in both of the Selected Investigations sections.

VECTOR ADDITION

Like Advanced Graphics, the Vector Addition program uses the TRS-80 as a visual tool. It helps students understand mathematics concepts, without overemphasizing the calculations involved. Using vector values entered by the student, the computer can display the vectors with their tails at a common origin, can draw the vectors tip-to-tail, and can display a list of vector values including values for the resultant vector.

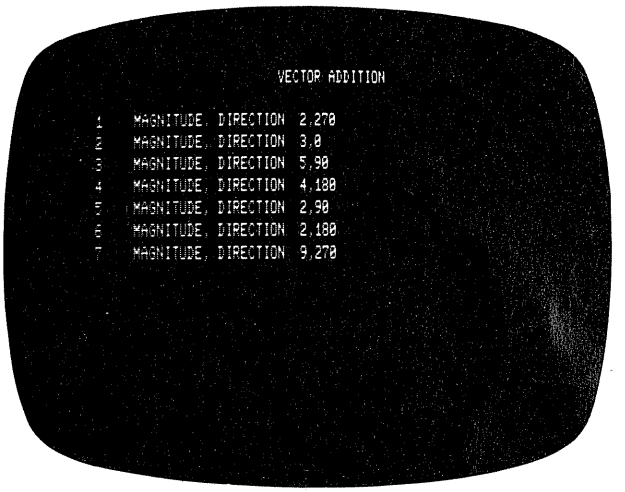
Let's take a look at how the program works. The student will probably begin by reading a stated problem and determining the magnitude and direction of each vector. For example, let's suppose that the problem states that you are walking in a city with square blocks. You walk two blocks south, three blocks east, five blocks north, four blocks west, two blocks north, two blocks west, and nine blocks south. The problem asks, if a crow were to fly from your starting point directly to your finishing point, how far would it need to fly and in what direction?

To solve the problem, you first need to assign values to the vectors. Begin with your first vector, which is "two blocks south." Superimposing the directions north, south, east and west over a Cartesian coordinate graph, we find that south is represented as 270°. Your first vector would therefore have the magnitude 2 (for the two blocks you walked) and would have the direction 270 (because south has been defined as

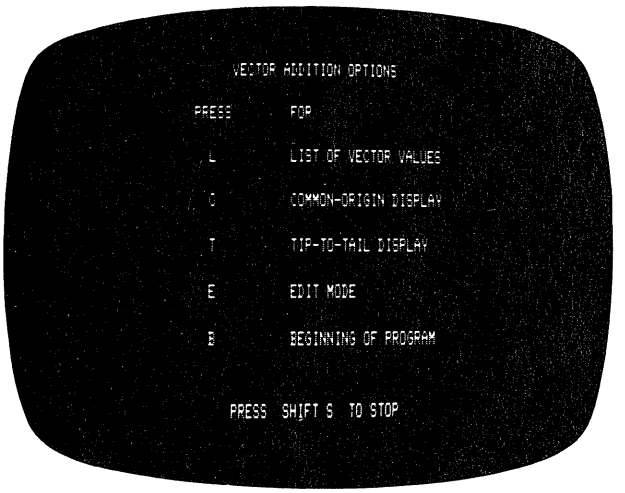


270°. You'd begin by typing the vector values **⟨2⟩⟨,⟩⟨2⟩⟨7⟩⟨0⟩** and pressing **⟨ENTER⟩**.

Once you had entered all vector values for the problem, your screen would look like this:

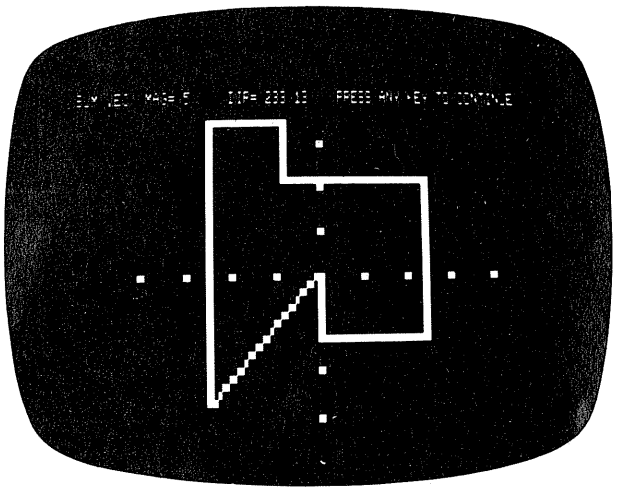


To continue, you'd press **⟨ENTER⟩**. The next screen would appear:



If you then typed **⟨T⟩** for a "tip-to-tail display" and pressed the space bar until all of your vectors were displayed, you'd see the following on your video screen.

The Vector Addition program is especially suited for use in high school physics classes as a supplement to introductory units covering vectors. To support a continuing focus on vectors throughout the physics course, the Selected Investigations section of the Vector Addition manual presents exer-



cises in areas ranging from fundamentals of vectors to electrical forces and fields. Intermediate vector concepts for which problems are included are displacement, velocity and acceleration, force, gravitation, and conservation of momentum. A Programming Guide outlines the structure of the Vector Addition program, while an Answer Key provides answers to the problems in the Selected Investigations section.

Advanced Graphics (26-1714) and Vector Addition (26-1720) can be run on the TRS-80 Model I or Model III 16K tape system, or 32K or 48K disk system. The suggested retail price of each is $29.95, though prices may vary at individual stores and dealers. For more information, contact your local Radio Shack Store or Computer Center.

String Packing for the Color Computer

Gordon Duff
500 Pittsfield-Lenox Road
Lenox, MA 01240

When I received the May issue of the TRS-80 Microcomputer News, I was disappointed. When are you going to start publishing information of a more technical nature regarding the Color Computer? In particular, I was very interested in the article on using VARPTR to compress graphic strings, and was very upset when I discovered that you did not include the information necessary to adapt the program to run on my computer. After a lot of experimenting, I managed to figure out the required modifications on my own, but this is a very frustrating procedure, with which I have become all too familiar.

I have included a listing of my version of the string packing program in case you would like to print it for other owners of the Color Computer to experiment with. I am sure there are other people like myself who would appreciate having this type of information made available. I bought a Color COMPUTER, not a color TV game.

```
6ØØØ CLS
    : C$="    "
6Ø1Ø K=VARPTR(C$)
6Ø2Ø PRINT"VARPTR=";K
6Ø3Ø PRINT"LENGTH OF STRING=";PEEK(K)
6Ø4Ø LS=PEEK(K+3)
    : MS=PEEK(K+2)
6Ø45 PRINT"LSB OF STARTING ADDRESS=";LS
6Ø5Ø PRINT"MSB OF STARTING ADDRESS=";MS
6Ø6Ø S=LS+MS*256
6Ø7Ø PRINT"STARTING ADDRESS=";S
6Ø8Ø PRINT"CHARACTER CODE OF STRING NAME=
    ".PEEK(K-2);PEEK(K-1)-128
61ØØ PRINT"ASCII CODES=";
    : FOR J=ØTO PEEK(K)-1
    : M=S+J
611Ø PRINT PEEK(M);
    : NEXT J
    : PRINT
612Ø PRINT"C$=";C$
613Ø FOR I=1 TO PEEK(K)
614Ø PRINT"VALUE FOR BYTE";I;
    : INPUT V
615Ø M=S+I-1
616Ø POKE M V
617Ø NEXT I
618Ø GOTO 6ØØØ
```

Using Data Tapes

Harold Morrissette
P.O. Box 1064
Presque-Isle, ME 04769

I was thumbing through my TRS-80 users manual and noticed the example program for using data tapes. I loaded the program in my Model I Level I computer and ran it for eight days. Then I noticed that you could never find out what the previous day's temperature and humidity was. It would not print the information on the tape but just averaged it out. So one night after school I started to play around trying to make a program to print the previous data. This is what I came up with:

```
10 CLS
20 INPUT "ENTER TODAY'S DATE";T
30 D=1
40 PRINT "REWIND DATA CASSETTE TO DATA."
50 INPUT "PRESS ENTER WHEN READY";A$
60 X=D
70 INPUT #Y,Z
80 PRINT "DATE";X;"TEMP.";Y;"HUMID.";Z;"SPACE BAR UNTIL YOU
   GET TO THE END OF THE SCREEN THEN END IT WITH"  THIS
   WILL MAKE IT SO THE NEXT LINE WILL GO RIGHT UNDER THE
   TOP ONE.
90 D=D+1
100 IF T=1 THEN 200
110 T=T-1
115 GOTO 60
200 GOTO 200
```

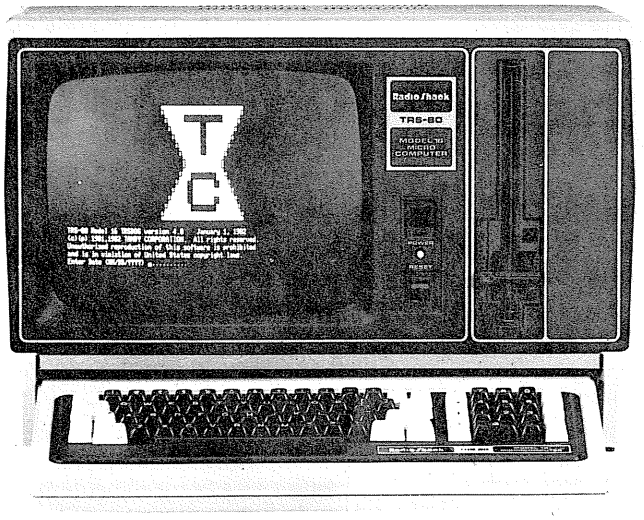
This is what the program looks like when you run it:

```
Enter today's date?
Rewind data cassette to data.
Press enter when ready?
Date 1    Temp. 70    Humid. 30
Date 2    Temp. 62    Humid. 30
Date 3    Temp. 75    Humid. 50
Date 4    Temp. 63    Humid. 45
Date 5    Temp. 70    Humid. 28
Date 6    Temp. 64    Humid. 20
Date 7    Temp. 63    Humid. 30
Date 8    Temp. 60    Humid. 34
```

The New TRS-80 Model 16

Radio Shack Introduces the Most Powerful Self-Contained Microcomputer Ever

Radio Shack is happy to announce the new generation of desktop computers. The TRS-80 Model 16 puts minicomputer performance in a complete, compact desktop system. Maximum performance from a microcomputer—it took Radio Shack to design it.



SIXTEEN-BIT TECHNOLOGY

The new TRS-80 Model 16 is designed around a state-of-the-art MC68000 microprocessor—a true 16-bit central processing unit (CPU) that has a 16-bit data bus, but processes information internally as 32-bit "words." This allows the Model 16 to utilize much more memory than systems using the more familiar 8-bit microprocessors.

Among the many advantages of this design is the ability to perform tasks with increased speed allowing more complex tasks to be accomplished with ease. This means that many jobs too complex for ordinary microcomputers can now be done on a completely self-contained desktop computer: Model 16, the maxi-micro!

DUAL-PROCESSOR ARCHITECTURE

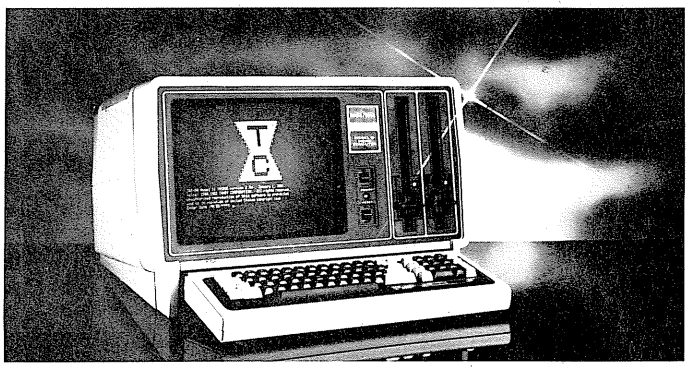
In addition to the 16-bit 68000 main processor, Model 16 incorporates a second processor—the Z-80A. The Z-80A handles all of the input/output functions of the main 16-bit CPU. In effect, it relieves the 68000 of time-consuming "housekeeping chores." As a result, the main CPU's raw processing power is utilized for number crunching and memory management.

128K—512K INTERNAL MEMORY

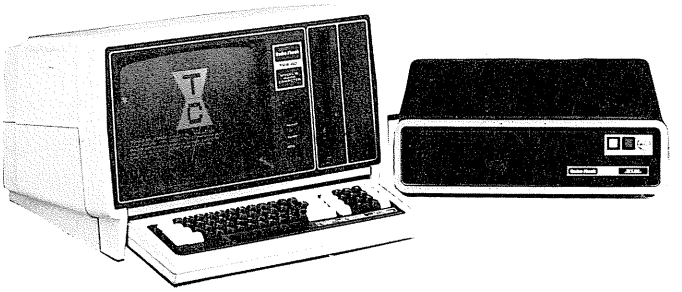
Model 16 includes a 128K byte Random Access Memory (RAM) board that can accept an additional 128K RAM at any time. Another 128K memory board may then be added, complete with provisions to add still another 128K, for a total RAM memory of 512K! The memory IC's in the Model 16 are 64K RAMS.

DELUXE FEATURES

Model 16 comes with one or two built-in thin-line double-sided, double-density disk drives. Each drive is capable of



storing 1.25 million characters on 8″ floppy diskettes, giving you the potential of 2.5 million characters of disk storage with no external equipment needed. The two slim-line drives occupy the same physical space as one of the current Model II drives. For additional storage, up to four of Radio Shack's 8.4 Megabyte Hard Disk Drives can be added.



A high-resolution 12 inch video display with 24 lines of 80 characters each is standard. A green screen is used to enhance readability and reduce glare. The detachable typewriter-style keyboard features a numeric keypad and eight special-function keys.

The Model 16 design includes two serial (RS-232C) ports. One of these ports is also designed for both BiSynch and SNA/SDLC communications to mainframes without the need for additional hardware. Also included is a parallel printer port. A hard disk port can be added for hard disk operation.

SOFTWARE COMPATIBILITY

Dual processor design permits the Model 16 to operate as a Model II and use existing TRS-80 Model II software! The

Z-80A is the same 8-bit microprocessor used in our highly respected TRS-80 Model II Business computer. So we've given the Model 16 the ability to function as a Model II and run existing TRS-80 Model II business software! This provides a large base of ready-to-run software, compatible with your Model 16 right now! This hardware/software compatibility is a rare feature in a new generation of computers. But you get it with Radio Shack's TRS-80 Model 16!

Software to take full advantage of the Model 16's 16-bit processing power is being developed by Radio Shack. Model 16's operating system is virtually "Library Command" compatible with the popular TRS-80 Model II. An Editor/Assembler is provided for machine language program development, and includes an Editor, assembler, linking loader, cross-reference and debugger.

Model 16 COBOL, the same one that is currently available for the Model II and III computers, will be available at the same time as the Model 16. This will allow us to move many of our application programs to the Model 16, as well as allowing many minicomputer applications packages to be moved to the 16 with little effort. During the coming year Radio Shack will add FORTRAN and both Compiler and Interpreter BASICs.

By mid-year Radio Shack will offer an integrated, fully interactive small business accounting system which will run on either floppy disk or hard disk. This system consists of six modules; General Ledger, Accounts Payable, Payroll, Accounts Receivable, Order Entry/Inventory Control, and Sales Analysis, all of which share the same common data base. These are very high quality COBOL programs which were all originally implemented on 16-bit minicomputers. All of these programs are currently available for the Model II, except for Payroll, as floppy disk programs. The programs will be recompiled to operate on the Model 16 with slight modifications for multitasking.

MULTI-USER CAPABILITY

One of the advantages of a 16-bit microprocessor is multitasking. The Radio Shack TRS-80 Model 16's advanced design allows a stand-alone computer to become the hub of a multi-user system. In a typical multi-user application, terminals would be connected to a Model 16 by the serial ports. Each user could then run a totally independent application package simultaneously with the other users. The Model 16 is hardware equipped to support three simultaneous users, one at its keyboard and two at terminals. This feature gives you a powerful expansion option that you can implement whenever you want. In this manner the Model 16 can grow to the equivalent of three separate computers. The multi-user operating system will be available about mid-year 1982.

TRS-80 MODEL II OWNERS

You can upgrade your present Model II computer to 16-bit computing power and up to 256K memory.

You can equip your Model II with our new TRS-80 Model 16 Enhancement Option! The Model 16 Enhancement Option provides you with the 16-bit, dual-processor, multi-user power of the Model 16.

The upgrade kit consists of the Model 16 software, documentation, and a Model 16 processor board and a 128K memory board. The processor and memory boards are installed in the existing card slots of any Model II. The only limitation to the upgrade is that the maximum memory is limited to 256K. Your present disk drive and monitor will be retained, and you'll still be able to use all of your existing Model II software. The 64K bytes that you have been using with the Z-80A will still be included. In the Model 16 mode, this memory is used as an Input/Output buffer, leaving the Model 16 memory available for your operating system and application software.

ARCNET COMPATIBILITY

Model 16s may be added to an ARCNET system (see the November, 1981 Microcomputer News for information on ARCNET) as applications processors in a mix with Model II's. In addition, the Model 16 may be used in a multitasking mode with two terminals attached and still run on ARCNET.

AND WE HAVE THE TERMINALS!

To use the TRS-80 Model 16 as a multi-user office system, just add an optional video terminal, such as Radio Shack's new DT-1, at each location.

Radio Shack's new TRS-80 DT-1 is a no-compromise video terminal. It is ideally suited for communications with Model 16. With the addition of a low-cost Radio Shack Modem, DT-1 can access other host computers and time-sharing information networks.

STATE-OF-THE-ART DESIGN

Unlike many terminals, you can easily set up and change the configuration of the DT-1 from the keyboard. There are no dip switches to bother with. The secret? A revolutionary non-volatile memory (EEPROM) that retains your terminal configuration even after power off, and without the need for battery backup!

MULTIPLE EMULATION MODES

The DT-1 is completely code compatible with four standard terminal protocols. It can emulate a Televideo 910, a Lear Siegler ADM-5, an ADDS 25 or a Hazeltine 1410 terminal.

OTHER FEATURES

The 12" diagonal black and white CRT displays 24 lines of 80 upper and lower case characters per line. Special single-character symbols are used to display control characters. The DT-1 is capable of displaying 6 video attributes including normal, reverse, invisible, blink, underline and half intensity. The user can select from 10 different baud rates, four different cursor types, and can program the carriage return, control key codes and other operating parameters. The DT-1 also includes both serial and parallel printer interfaces. The heart of the DT-1 is a custom programmed high speed 8-bit microprocessor which gives the DT-1 its high performance and low price.

SUGGESTED RETAIL PRICES

Model 16

Single 8" disk drive, 128K - 26-6001 $4999
Two 8" disk drives, 128K - 26-6002 $5798
Memory Board with 128K- 26-6011 $699
128K Memory Kit - 26-6012 $499
Model 16 Enhancement option
for the Model II - 26-6010 $1499

DT-1 Data Terminal - 26-6050 $699

Model II Bugs, Errors and Fixes

ACCOUNTS PAYABLE (26-4505)

We've had several inquiries about the option of posting invoices without having to return to the Main Menu on Version 1.0. The only purpose behind this would be to save time and not have to reload those large programs. If these changes are made, then a COMPLETE posting must be done since the selection of invoices is done from the Main Menu and is bypassed.

GOING DIRECTLY TO INVOICE POSTING FROM INVOICE MAINTENANCE:

To add the option of running the POST INVOICES function from the INVOICE MAINTENANCE function without returning to the MAIN MENU first, the following changes should be made. Note: You must request a Complete posting since the selection process is done from the main menu and will be bypassed.

The problem is corrected by following the steps listed below.

1. Backup the diskette(s) and make the changes on the Backup copy of the program.
2. In BASIC, load the program by typing

```
LOAD"APINVCE/BAS"
```

3. Make the following corrections:

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

```
Old Line: 502 PRINT@(23,20),EL$"> ENTER A SELECTION
          :";:FL=1:GOSUB10:IFCF=1THEN850
          ELSEIFCF<>0THEN502

New Line: 502 PRINT@(23,20),EL$"> ENTER A SELECTION
          :";:FL=1:GOSUB10:IFCF=1ORCF=2THEN850
          ELSEIFCF<>0THEN502

Old Line: 850 PRINT@(23,20),EL$;R$"RETURNING TO MAIN
          MENU"N$;:CLOSE:OPEN"O",1,VP$:PRINT#1,I;IT;IN;ID
          ;IS;IH;VT;VN;VD;EP;TD;R0;R1;R2;R3;R4;R5;R6;R7;R
          8;FC;LC;STR$(CP#)+"D0"

New Line: 850 CLOSE:OPEN"O",1,VP$:PRINT#1,I;IT;IN;ID;IS;IH
          ;VT;VN;VD;EP;TD;R0;R1;R2;R3;R4;R5;R6;R7;R8;FC;
          LC;STR$(CP#)+"D0"

Old Line: 860 FORW9=1TOVN-1:PRINT#1,P(W9,0);P(W9,1):NEXT
          :FORW9=1TOIN-1:PRINT#1,I(W9):NEXT:CLOSE1
          :RUN"APS/BAS

New Line: 860 FORW9=1TOVN-1:PRINT#1,P(W9,0);P(W9,1):NEXT
          :FORW9=1TOIN-1: PRINT#1,I(W9):NEXT:CLOSE

Old Line: 1024 PRINT:PRINTTAB(10);FNRV$("D");"ELETE";
          TAB(25);FNRV$("E");"DIT";TAB(40);FFNRV$("N")
          ;"EXT"

New Line: 1024 PRINT:PRINTTAB(10);FNRV$("D");"ELETE";
          TAB(25);FNRV$("E");"DIT";TAB(40);FNRV$("N");
          "EXT";TAB(55);FNRV$("F2")" POST INVOICES"
```

ADDITIONS (Type the line followed by an ⟨ENTER⟩)

```
870 IFCF=2THENRUN"APPOST/BAS"ELSERUN"APS/BAS"
```

4. Type SAVE"APINVCE/BAS" and press ⟨ENTER⟩.
5. At TRSDOS Ready, make a backup copy of the diskette.

RSBASIC COMPILER (26-4705)

Following are corrections for four problems that may occur when using this package.

1. ERROR 25 IN ISAM WITH TRSDOS 4.0

When RUNBASIC is chaining a series of programs under the 4.0 Hard Disk operating system, it will give an error 25 (PASSWORD PROTECTION) when it attempts to open the ISAM module a second time. This PATCH will correct that problem:

```
PATCH RUNBASIC A=2AF2 F=11612B C=CD9D2B
PATCH RUNBASIC A=2B9D F=00000000000000000000
                C=11612B2323233652C9
```

2. TRANSFERRING RSBASIC TO TRSDOS 4.0 HARD DISK SYSTEM

The following PATCH is required whenever RSBASIC is transferred to a TRSDOS 4.0 Hard Disk system. This PATCH will allow RSBASIC to recognize drive numbers 4 through 7.

```
PATCH RSBASIC/OLF R=27 B=53 F=12DB C=CE00
```

3. DATA NOT FOUND IN ISAM FILES

Sometimes with version 2.4, when calculating the record length and the number of records in a file, it is possible not to find a record that is actually in the file. The problem has occurred when OPENing two Binary files and two ISAM files at the same time. An OVERFLOW error persists regardless of the stack size or record length. Another manifestation of the problem is OPENING, WRITING, and CLOSING alternately between two ISAM files. In this case, another form of the error will occur after several records have been written.

To correct the problem, apply the following PATCHes:

```
PATCH RSBASIC/LIO A=3666 F=2024 C=0003
PATCH RSBASIC/LIO A=369B F=0037 C=4256
PATCH RSBASIC/OLF R=92 B=120 F=00000000000 C=DDE1C30956
PATCH RSBASIC/OLF R=155 B=67 F=8B35 C=7756
PATCH RSBASIC/OLF R=155 B=103 F=2024 C=0003
PATCH RSBASIC/OLF R=155 B=156 F=0037 C=4256
```

4. PATCHES FOR BOTH 26-4705 AND 26-4706

Any program compiled under RSBASIC Compiler and operating under RUNBASIC or RUNTIME BASIC can develop a problem when processing ISAM files. This problem will be manifested as an unexpected termination of the program and a return to TRSDOS READY status without displaying any error messages. The problem is caused when RUNBASIC (or RUNTIME BASIC) calls the module, RSBASIC/LIO, which handles the deletion of ISAM records). When deleting a particular group of ISAM records, the module fails to complete the deletion process, which causes RUNBASIC (or RUNTIME BASIC) to fail with a subsequent return to TRSDOS. These PATCHes will correct the problem:

```
PATCH RSBASIC/LIO A=45C2 F=44 C=4A
PATCH RSBASIC/LIO A=45EA F=1C C=22
PATCH RSBASIC/LIO A=45FE F=3654183C C=A26118AD
PATCH RSBASIC/LIO A=4602 F=00C3040A C=1A013654
PATCH RSBASIC/LIO A=4606 F=7624186 C=0E3C00C3
PATCH RSBASIC/LIO A=460A F=1A58A259 C=060A7024
```

BISYNC 3270 (26-4715)

The following PATCH allows the Model II, when operating in a 3275 polled environment, to respond to EOTs. This will allow multiple Model IIs which are emulating 3275s to operate on the same line with specific or general polling, as long as the Control Unit Addresses are unique.

```
PATCH BIS3270 A=7D40 F=CC607D C=000000
```

BISYNC 3780 (26-4716)

The following PATCH corrects problems with form feeds when using the 26-4716 BiSync Package. The problem is

characterized by invalid 'line skips' and 'page ejects'. The following PATCH corrects the problem and should result in properly printed documents:

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

TRSDOS (26-4910)

IMPORTANT NOTE: Page 40 of the January, 1982 Microcomputer News contains a one-byte PATCH to BASIC on TRSDOS 2.0a. **DO NOT** use this PATCH. This information was released inadvertently and SYSTEM problems may result if you install the PATCH. Please unPATCH using the following:

PATCH BASIC A=65B6 F=23 C=21

Following are several PATCHes for Model II TRSDOS.

DISK I/O PATCHES

We are now using several suppliers of the FDC controller for the TRS-80 Model II. Some of these chips have different timing constraints from the original chip. This has caused some machines to demonstrate problems when using the new chips.

If your machine is demonstrating I/O problems with disk (lockups requiring system reset) then these PATCHes should be applied. We hesitate to recommend applying these PATCHes to a system that is working properly. However, there should be no effect at all in applying these PATCHes to any system with any of the chips in current use.

For TRSDOS 1.2a make the following PATCHes:

```
PATCH IODVRS/SYS A=ØB9C F=3ED8D3E4CDB1ØBDBE7
                       C=F33ED8D3E43EDØD3E4
PATCH IODVRS/SYS A=ØBA5 F=DBE43EDOD3E4CDB1ØB
                       C=FBØØØØØØCDB1ØBDBE7
PATCH FORMAT A=2D69 F=3EDØD3E4CD7F2DDBE7
                    C=F33ED8D3E43EDØD3E4
PATCH FORMAT A=2D72 F=DBE43EDØD3E4CD7F2D
                    C=FBØØØØØØCD7F2DDBE7
```

For TRSDOS 2.0a make the following PATCHes:

```
PATCH SYSRES/SYS A=ØE54 F=3ED8D3E4CD6CØEDBE7
                        C=F33ED8D3E43EDØD3E4
PATCH SYSRES/SYS A=OE5D F=DBE43EDØD3E4CD6CØE
                        C=FBØØØØØØCD6CØEDBE7
PATCH FORMAT A=2D19 F=3ED8D3E4CD2F2DDBE7
                    C=F33ED8D3E43EDØD3E4
PATCH FORMAT A=2D22 F=DBE43EDØD3E4CD2F2D
                    C=FBØØØØØØCD2F2DDBE7
```

2.0a TERMINAL PROGRAM—OPTIONAL PATCHES FOR CHANNEL B USE

These PATCHes are to change the operation of the terminal program from serial channel A to serial channel B on a permanent basis. Do not apply these PATCHes unless you have a special situation which requires the use of the terminal program on channel B only. Please note that you should not try to use a Serial Printer (which also uses Channel B) once these PATCHes have been applied. If you attempt to do so, bytes will be intermixed on Channel B.

```
PATCH TERMINAL A=343C F=3E61 C=3E63
PATCH TERMINAL A=3488 F=3E6Ø C=3E62
PATCH TERMINAL A=352C F=3E64 C=3E65
PATCH TERMINAL A=3535 F=3E64 C=3E65
PATCH TERMINAL A=353F F=3E64 C=3E65
PATCH TERMINAL A=3549 F=3E61 C=3E63
PATCH TERMINAL A=3583 F=3E6Ø C=3E62
PATCH TERMINAL A=36A9 F=3E6Ø C=3E62
PATCH TERMINAL A=36D6 F=3E6Ø C=3E62
```

2.0a HOST FUNCTION—OPTIONAL PATCHES FOR CHANNEL B USE

These PATCHes are to change the operation of the HOST function from serial channel A to serial channel B on a permanent basis. Do not apply these PATCHes unless you have a special situation which requires the use of the HOST function on channel B only. Please note that you should not try to use a serial printer (which also uses serial channel B) once these PATCHes have been applied. If you attempt to do so, bytes will be intermixed on channel B.

NOTE: If you have a 32K Model II replace SYSTEM64 with SYSTEM32 in the first three PATCHes.

```
PATCH SYSTEM64 R=24 B=53 F=3E64 C=3E65
PATCH SYSTEM64 R=24 B=98 F=3E61 C=3E63
PATCH SYSTEM64 R=25 B=65 F=1115ØØ C=1117ØØ
PATCH SYSTEM/SYS R=1Ø7 B=2Ø5 F=41 C=42
PATCH SYSTEM/SYS R=1Ø7 B=214 F=CB5E28E1CB9EØ66Ø
                              C=CB5628E1CB96Ø662
PATCH SYSTEM/SYS R=1Ø7 B=231 F=Ø664 C=Ø665
PATCH SYSTEM/SYS R=1Ø7 B=239 F=42 C=41
PATCH SYSTEM/SYS R=1Ø7 B=242 F=CB5628C5CB96Ø662
                             C=CB5E28C5CB9EØ66Ø
PATCH SYSTEM/SYS R=1Ø8 B=3 F=Ø665 C=Ø664
PATCH SYSTEM/SYS R=61 B=141 F=CB5E C=CB56
PATCH SYSTEM/SYS R=62 B=3 F=41 C=42
```

OPTIONAL PATCH FOR 2.0 AND 2.0A TRSDOS

135 BAUD Option in SETCOM:

The following PATCHes are designed to allow the option of selecting 135 BAUD under the SETCOM utility. This requires the deletion of 4800 BAUD as a selection since space would not allow an additional option. If circumstances require the use of both BAUD rates at some time, the user simply needs to reverse the Find and Change values.

```
PATCH SYSTEM64   R=14  B=118 F=1A47 C=74Ø7
PATCH SYSTEM/SYS R=1Ø4 B=125 F=CØ12 C=87ØØ
```

NOTE: 135 BAUD typed in SETCOM command indicates an actual BAUD rate of 134.5.

Model II BACKIT

Chuck Rizzio
RSCC 7430 Orange, CT

BACKIT is a utility created on the Model II using a DO file. How many times have you started a BACKUP and left to get a cup of coffee only to find on your return:

Destination Disk Ready?

At TRSDOS Ready enter the following:

BUILD BACKIT:0 ⟨**ENTER**⟩

PAUSE Model II BACKIT Utility - Blank Disk in Drive 1??? ⟨**ENTER**⟩

FORMAT :1 {ABS} ⟨**ENTER**⟩⟨**ENTER**⟩

BACKUP :0 :1 {ABS} ⟨**ENTER**⟩⟨**ENTER**⟩

PAUSE BACKUP Complete 0 = = = ⟩ 1 ⟨**ENTER**⟩⟨**ENTER**⟩⟨**ENTER**⟩

To execute BACKIT type DO BACKIT ⟨**ENTER**⟩ at TRSDOS Ready. If you want to change the destination drive, simply change the second number in the BACKUP :0 :1. You can also do consecutive backups, i.e.:

BACKUP :0 :1 {ABS}

BACKUP :0 :2 {ABS}

BACKUP :0 :3 {ABS}

Pocket Computer Programs

Fred Nachbaur
1932 Clinton Street
Los Angeles, CA 90026

Since I acquired my TRS-80 Pocket Computer in December 1980, the little jewel has kept me fairly busy finding ever more applications for it. I recently saw my first Microcomputer News (July 1981) when my employer started receiving it after purchasing a Color Computer, and was delighted to find that the little Pocket Computer also had a column devoted to it.

Before I get any further, please let me compliment TANDY CORPORATION for taking on the distribution of this excellent product. I find it perfectly suited for my needs as a technician and engineering student; when I began to discover the sophistication viewpoint, this little powerhouse obsoletes the scientific calculator much as the calculator obsoleted the slide rule for most applications.

I am very much interested in seeing the Pocket Computer (and its "descendants") widely promoted and used as the powerful engineering tool it is. It is in this spirit that I am submitting a number of programs written at various phases of experience with the P.C. Improvements could be made for most, but all work and some serve to illustrate some of the "fine points" of this computer.

Before I get into the programs per se, I would like to share a few operational "tricks" I have come across, as well as a few other observations.

When trying to cram a lot of program into the 1424 program steps available (or less if flexible memories are needed) it is helpful to know that steps can be saved by omitting the ending quote mark after a string without affecting operation:

1. After a PRINT or PAUSE statement (also PRINT# and INPUT#) in a program if the end of the string is at the end of a line. e.g.:

```
1Ø: PRINT "HI, JOE
2Ø: PAUSE "HOW ARE YOU
3Ø: INPUT "LOAD DATE?";B$
    : IF B$="YES" INPUT# "DATA
```

will run, whereas

```
1Ø: PRINT "HI,JOE
    : PAUSE "HOW ARE YOU
```

will not (HI,JOE : PAUSE will be PRINTed and the computer will not know what to do with HOW ARE YOU). Another possibility is when specifying the contents of a string variable:

```
5Ø: B=Ø
    : F$="WIN
```

will work. Use this only when the $ variable will be displayed as the last item on a line, since in the above example F$ is actually loaded with WIN_____ (W, I, N, ⟨SPC⟩, ⟨SPC⟩, ⟨SPC⟩, ⟨SPC⟩). In other words, 'trailing spaces" are not suppressed.

2. In manual calculation mode, string variables may be loaded as above, e.g. typing ⟨A⟩ ⟨SHFT⟩ ⟨$⟩ ⟨=⟩ ⟨SHFT⟩ ⟨"⟩ ⟨W⟩ ⟨I⟩ ⟨N⟩ ⟨ENTER⟩ (A$="WIN) will load memory A$ with "WIN_____." More useful is the fact that the trailing quote mark need not be used with CLOAD or CSAVE statements. This allows us to reserve ⟨SHFT⟩⟨L⟩ to CLOAD" and ⟨SHFT⟩⟨S⟩ to CSAVE," for example. Thus, as long as you are in the RUN or PRO mode, tape loading and saving operations can be carried out by hitting ⟨SHFT⟩⟨L⟩ (or ⟨S⟩), typing the file name, and hitting ⟨ENTER⟩, thus saving two keystrokes. Interestingly, trailing spaces do not seem to matter in this case; evidently they are not recorded in the ID message.

When using the printer and writing a program containing long strings in PRINT statements, memory can be saved by phrasing the string such that a single PRINT statement actually prints two or more lines on the printer: e.g.

```
1Ø: PRINT " OPERATING INST-"
    : PRINT " RUCTIONS : SHFT"
2Ø: PRINT "A TO INPUT DATA"
    : PRINT "B TO REVIEW DATA"
    : PRINT "C TO EVALUATE"
3Ø: PRINT " "
    : END
```

takes 108 program steps. The exact same result is obtained using this modification:

```
1Ø: PRINT " OPERATING INST- RUCTIONS: SHFT"
2Ø: PRINT "A TO INPUT DATA B TO REVIEW DATAC TO
    EVALUATE"
3Ø: PRINT " "
    : END
```

but this one requires only 97 steps—saving 11 steps, which can make an appreciable difference in a busy program.

Another note on printer use. When operating the computer without the printer, programs such as the following are useful for inputting data (such as values on a graph, coefficients, etc.) to sequential variable locations:

```
1Ø: D=Ø
2Ø: PRINT "X("; USING "###"; D; ")="
3Ø: " " AREAD A(D+27)
4Ø: IF D<5Ø LET D=D+1
    : GOTO 2Ø
```

This will not work, however, if the printer is on; all you get is a list of PRINT statements. An easy way to modify the program is to simply change the PRINT statement in line 20 to PAUSE, change the AREAD statement in 30 to INPUT, and move the definable program label (" ") to line 20 instead of 30. (Actually, the " " in line 30 need not be removed, since the computer always goes to the first line containing the definable tag ⟨SHFT⟩⟨SPC⟩). This method is very easy to use;

the computer flashes which division number is being input and asks for input with a ?. If you forget which division number is being input, type ⟨**SHFT**⟩⟨**SPC**⟩ and the computer will again flash the X(n)= prompt, followed by ?.

PROGRAMS

Now on to the programs. First, a sequence of programs illustrating how the PRINT#, INPUT# and CHAIN commands may be used to the advantage of extending the computer's effective memory far beyond the 1424 steps available in RAM. The program sequence was designed while writing a paper on Fourier Analysis. These 17 programs evaluate the first 101 sine and cosine Fourier coefficients of an arbitrary repetitive (subject to certain restrictions) wave. The effective resolution of the program is 300 divisions, though only 150 memory locations are used (one of the restrictions is that the second half of the wave is assumed to be an exact negative of the first half—i.e. odd-harmonic only symmetry). Please note, these programs are set up for automatic use with the printer.

FOURSEQ

PROGRAM A

```
1: "A" PRINT "FOURIER ANALYSIS"
   : PRINT "3ØØ DIVS"
   : PRINT "ASSUME A(Ø)=Ø"
   : PRINT "ODD HARM ONLY"
2: C=3ØØ, B= pi/15Ø
3: INPUT "DATA ON TAPE? (YES/NO)"; I$
4: IF I$="YES" PAUSE "LOAD DATA TAPE"
   : INPUT #"DATA";A(27)
   : PAUSE "LOAD PGM TAPE"
   : CHAIN "C"
5: IF I$="NO" PAUSE "STAND BY"
   : CHAIN "B"
6: GOTO 3
```

PROGRAM B

```
1: "B" A=Ø
   : PAUSE "T/R OFF"
2: " " PAUSE "Y("; USING "####"; A; " )"
3: INPUT A(A+27)
   : GOTO 5
4: GOTO 3
5: A=A+1
   : IF A<151 GOTO 2
6: PRINT "ASSUME Y(X+15Ø)=-Y(X)"
7: INPUT "FILE DATA? "; I$
   : IF I$="NO" GOTO 1Ø
8: IF I$="YES"PAUSE "SET LEVEL"
   : PRINT # "DATA"; A(27)
   : GOTO 9
9: GOTO 7
1Ø: PAUSE "LOAD PGM"
11: CHAIN "C"
```

PROGRAM C

```
1: "C" PRINT " "
   : PRINT "SINE TERMS (A)"
   : RADIAN
   : FOR H=1 TO 33 STEP 2
2: A=Ø, F=Ø, D=Ø, G=1
   : GOSUB 1Ø
3: A=E+A, F=F+B, D=D+1
   : GOSUB 1Ø
4: A=4E+A, F=F+B, D=D+1
   : GOSUB 1Ø
5: A=E+A
6: IF D<C/2 GOTO 3
7: A((H+19)/2)=2AB/3pi
8: NEXT H
9: BEEP 5
   : PRINT " "
   : CHAIN "D"
1Ø: E=A(D+27)*SIN HF
   : RETURN
```

PROGRAM D

```
1: "D" PRINT "N=   A(N)="
2: FOR H=1 TO 33 STEP 2
3: E=A((H+19)/2)
4: PRINT USING "###"; H; USING "######.######"; E
5: NEXT H
8: CHAIN "E"
```

PROGRAM E—Same as "C" except:

```
1: "E" PAUSE "SIN #35 TO 67"
   : FOR H=35 TO 67 STEP 2
.
.
.
7: A((H-15)/2)=2AB/3pi
9: BEEP 5
   : PRINT "T/R ON"
   : CHAIN "F"
```

PROGRAM F—Same as "D" except:

```
1: "F"PRINT"PGM - SIN 35 TO 67"
2: FOR H=35 TO 67 STEP 2
3: E=A((H-15)/2)
.
.
7: PRINT "DATA TAPE, SET LEVEL"
   : PRINT# "S35-67"; A(1Ø)
8: PRINT "REPEAT - SHFT F"
   : PRINT "T/R ON"
   : CHAIN "G"
```

PROGRAM G—Same as "C" except:

```
1: "G" PAUSE "SIN #69-1Ø1"
   : FOR H=69 TO 1Ø1 STEP 2
.
.
7: A((H-49)/2)=2AB/3pi
.
9: BEEP 5
   : CHAIN "H"
```

PROGRAM H—Same as "D" except:

```
1: "H" PRINT "PRINT PGM - SIN 69-1Ø1"
2: FOR H=69 TO 1Ø1
3: E=A((H-49)/2)
.
.
.
.
8: CHAIN "I"
```

PROGRAM I

```
2: "I" INPUT "FIND COS TERMS? ";I$
3: IF I$="YES" PAUSE "T/R ON"
   : C=1ØØ
   : B=pi/15Ø
   : RADIAN
   : CHAIN "J"
4: IF I$="NO" GOTO 6
5: GOTO 2
6: INPUT "RE-CONSTRUCT? ";I$
7: IF I$="YES" PRINT "T/R ON"
   : CHAIN "FOURSN"
8: IF I$="NO" PRINT "END OF PROGRAM"
   : PRINT "BY F., NACHBAUR"
   : END
9: GOTO 5
```

FOURCOS

```
1: "FOURCOS" C=3ØØ
   : B=pi/15Ø
   : RADIAN
   : CHAIN "J"
```

PROGRAM J—Same as "C" except "SIN" is replaced by "COS" in lines 1 and 10. Chains "K".

PROGRAM K—Same as "D" except "SIN" replaced by "COS" in line 1. Chains "L"

PROGRAM L—Same as "E" except "SIN" replaced by "COS" in lines 1 and 10. Chains "M".
PROGRAM M—Same as "F" except "SIN" replaced by "COS" in line 1. Chains "O"
PROGRAM O—Analogous to "G." Chains "N".
PROGRAM N—Analogous to "H." Chains "P".
PROGRAM P

```
1: "P" INPUT "RECONSTRUCT?"; I$
2: IF I$="NO" PRINT "END OF PROGRAM"
   : PAUSE "BY F. NACHBAUR"
   : END
3: IF I$="YES" PAUSE "T/R ON"
   : CHAIN "FOURSN"
4: GOTO 1
```

FOURIER SYNTHESIS—CONSTRUCTS A WAVEFORM FROM KNOWN FOURIER COEFFICIENTS.

FOURSN

```
1: "A" PAUSE "FOURIER SYSNTHESIS"
   : PAUSE "3ØØ DIVS"
   : PAUSE "ASSUME A(0)=Ø"
2: PAUSE "ODD HARMONICS ONLY"
   : PAUSE "STAND BY"
   : DEGREE
   : C=3ØØ
3: FOR D=27 TO 177
4: A(D)=Ø
5: NEXT D
6: PAUSE "STAND BY"
   : CHAIN "D"
```

PROGRAM B
RUNS SIN OR COS 69-101 FIRST IF ALREADY IN MEMORY

```
1: INPUT "COEFF. 69 - 1Ø1 IN MEM? "; I$
   : IF I$="YES" GOTO 4
2: IF I$="NO" LET G$=" "
   : CHAIN "C"
3: GOTO 1
4: INPUT "SIN OR COS IN MEM? "; G$
   : IF G$="SIN" CHAIN "H"
5: IF G$="COS" CHAIN "O"
6: GOTO 4
```

PROGRAM H

```
1: "H" PAUSE "S69-1Ø1"
   : FOR D=Ø TO 15Ø
2: B=1.2D
3: FOR H=69 TO 1Ø1 STEP 2
4: A(D+27)=A(D+27)+A((H-49)/2)*SIN HB
5: NEXT H
6: NEXT D
7: BEEP 5
   : PRINT "T/R ON"
   : CHAIN "C"
```

PROGRAM O

```
1: "O" PAUSE "C69-1Ø1"
   : FOR D=Ø TO 15Ø
2: B=1.2D
3: FOR H=69 TO 1Ø1 STEP 2
4: A(D+27)=A(D+27)+A((H-49)/2)*COS HB
5: NEXT H
6: NEXT D
7: BEEP 5
   : PRINT "T/R ON"
   : CHAIN "C"
```

PROGRAM C

```
1: "C" PRINT "SIN COEFF 1 TO 33"
   : H=1
2: PRINT "A("; USING "###"; H; ")= SHFT SPC"
3: " " AREAD E
   : A((H+19)/2)=E
4: H=H+2
5: IF H<34 GOTO 2
6: PRINT "T/R ON"
   : CHAIN "E"
```

PROGRAM E—Similar to "H" above. Chains "D".
PROGRAM D—Similar to "C" above. Chains "F".
PROGRAM F—Similar to "H" above. Chains "G".
PROGRAM G—Similar to "C," except skips programs G & H, if H done earlier.
PROGRAM H—Same as above except line 7 changed and line 8 added:

```
.
.
7: BEEP 5: IF G$="COS" CHAIN "J"
8: CHAIN "I"
```

PROGRAM I

```
1: "I" INPUT "ANY COS TERMS?"; I$
   : IF I$="YES" CHAIN "J"
2: IF I$="NO" CHAIN "K"
```

PROGRAM K

```
1: "K" PRINT "READY TO PRINT ANSWERS"
   : PRINT "T/R OFF"
2: FOR D=Ø TO 15Ø
3: A=A(D+27)
4: PRINT USING "####"; "Y("; D; USING
   "####.####"; A
5: NEXT D
6: PRINT "REPEAT - SHFT K"
   : PRINT "END OF PROGRAM"
   : PAUSE "F. NACHBAUR"
   : END
```

FOURSC

```
1: "A" PAUSE "FOURIER SYSNTHESIS"
   : PAUSE "3ØØ DIVS"
   : PAUSE "ASSUME A(Ø)=Ø"
2: PAUSE "ODD COS HARM ONLY"
   : PAUSE "STAND BY"
   : DEGREE
   : C=3ØØ
3: FOR D=27 TO 177
4: A(D)=Ø
5: NEXT D
6: PAUSE "STAND BY"
7: IF G$="COS" CHAIN "J"
8: CHAIN "O"
```

PROGRAM O—Same as "O" above except chains "J".
PROGRAM J—Same as "C" except cosine version chains "L."
PROGRAM L - Same as "E" except cosine version chains "M."
PROGRAM M—Same as "D" except cosine version chains "P".
PROGRAM P—Same as "F" except cosine version chains "N".
PROGRAM N—Same as "G" except cosine version chains "O" if not done before.
PROGRAM O—Same as "O" above except chains "S".
PROGRAM S—Same as "K".

FOURIER ANALYSIS

An earlier version of this program sequence, and a somewhat more general one, is FOURIER and its inverse FOURSYN. Both of these programs were designed for use without the printer, but could be modified easily.

Program : Fourier Analysis
Description : Up to 100 Y values of a repetitive function over one complete cycle are entered, and the Fourier coefficients (sine and cosine terms) of each harmonic up

to the tenth are evaluated. Data location of Fourier coefficients is compatible with the Fourier Synthesis program of re-construction and further analysis of the original wave. "Number of divisions" MUST BE AN EVEN NUMBER.

Operation : In DEF mode, ⟨**SHFT**⟩ ⟨**A**⟩ starts program. Enter # of divisions desired on prompt, then enter the sequential values of the function being evaluated using ⟨**SHFT**⟩ ⟨**SPC**⟩. Enter number of harmonics needed. Computer will commence evaluating the coefficients up to the desired harmonic and will beep when ready. Repeatedly hitting ⟨**ENTER**⟩ will then display first A° and the cosine terms, then the sine terms. ⟨**SHFT**⟩ ⟨**B**⟩ will repeat or restart the display sequence.

```
1Ø: "A" RADIAN
    : PAUSE "FOURIER ANALYSIS"
2Ø: INPUT "# DIVS (MAX 1ØØ)"; C
    : D=Ø
3Ø: PRINT "Y("; USING "####"; D; ")="
4Ø: " " AREAD A(D+32)
5Ø: IF D<C LET D=D+1
    : GOTO 3Ø
55: INPUT "# HARMONICS-MAX 1Ø="; A(31)
6Ø: PAUSE "WORKING"
    : B=2*pi/C
7Ø: FOR G= Ø TO 1
8Ø: I=3ØØ+1ØØ*G
9Ø: FOR H=Ø TO A(31)
1ØØ: IF G+H=Ø GOTO 21Ø
13Ø: A=Ø
    : F=Ø
    : D=Ø
    : GOSUB I
14Ø: A=E+A
    : F=F+B
    : D=D+1
    : GOSUB I
15Ø: A=E*4+A
    : F=F+B
    : D=D+1
    : GOSUB I
16Ø: A=E+A
17Ø: IF D<C GOTO 14Ø
175: E=A*B/(3*pi)
19Ø: IF G=Ø LET A(2*H+9)=E
2ØØ: IF G=1 LET A(2*H+1Ø)=E
21Ø: NEXT H
22Ø: NEXT G
23Ø: BEEP 5
    : PRINT "READY"
24Ø: "B" FOR H=Ø TO A(31)
245: A=2*H+1Ø
    : PRINT "A("; USING "###"; H; ")="; USING;
    A(A)
25Ø: NEXT H
255: IF A(31)=Ø GOTO 28Ø
26Ø: FOR H=1 TO A(31)
265: A=2*H+9
    : PRINT "B("; USING "###"; H; ")="; USING;
    A(A)
27Ø: NEXT H
28Ø: PRINT "REPEAT-SHFT B"
29Ø: END
3ØØ: E=A(D+32)*SIN(H*F)
31Ø: RETURN
4ØØ: E=A(D+32)*COS (H*F)
41Ø: RETURN
```

FOURIER SYNTHESIS

Program : Fourier Synthesis

Description : Known coefficients of sine and cosine terms of a complex wave (Fourier coefficients), up to 10 harmonics plus DC term, are input and the computer evaluates up to 100 equally spaced points for plotting purposes. It can be used in conjunction with Fourier Analysis program "FOURIER" to reconstruct a wave with up to ten of its component harmonics.

Operation : In DEF mode, ⟨**SHFT**⟩ ⟨**A**⟩ starts program. Enter # of divisions desired and # of harmonic terms to be taken into account on prompt. Inputting of A (cosine) and B (sine) terms use ⟨**SHFT**⟩ ⟨**SPC**⟩. Average (DC) level is defined as $A_o/2$, so for A(O) be sure the value is twice the average value of the function over a complete cycle. Computer will beep when all points have been evaluated; repeatedly hitting ⟨**ENTER**⟩ will sequentially display the function value and the corresponding division number for ease in plotting. For a repeat of the output sequence use ⟨**SHFT**⟩ ⟨**B**⟩ (in DEF mode). Hitting ⟨**ENTER**⟩ twice after the last data point has been displayed ends program.

```
1Ø: "A" DEGREE
    : PAUSE "FOURIER SYNTHESIS"
    : D=Ø
2Ø: INPUT "# HARMONICS (MAX 1Ø)="; F
3Ø: INPUT "#DIVS="; C
    : A$= "B(SIN)#"
    : B$= "A(COS)#"
4Ø: H=INT(D/2)
    : E=D+1-2*H
5Ø: IF D+H=Ø GOTO 8Ø
6Ø: PRINT A$(E); USING "###"; H
7Ø: " " AREAD A(D+9)
8Ø: D=D+1
    : IF D<2*F+2 GOTO 4Ø
85: PAUSE "WORKING"
9Ø: FOR G=Ø TO C
1ØØ: A=J/2
    : B=G*36Ø/C
11Ø: FOR H=1 TO F
12Ø: A=A+A(2*H+1Ø)*COS(H*B)
13Ø: NEXT H
14Ø: FOR H=1 TO F
15Ø: A=A+A(2*H+9)*SIN(H*B)
16Ø: NEXT H
17Ø: A(G+32)=A
18Ø: NEXT G
19Ø: BEEP 5
    : PRINT "READY"
2ØØ: "B" FOR G=Ø TO C
2Ø5: A=A(32+G)
21Ø: PRINT "Y("; USING "####"; G; ")"; USING; A
22Ø: NEXT G
23Ø: PRINT "REPEAT-SHFT B"
24Ø: END
```

Memory:
A - Sums Fourier terms; answer
A$ - "B(COS)#" (Input Program only)
B - Increment of X
B$ - "A(SIN)#" (Input program only)
C - # of divisions
D - Input index
E - String selector
F - # Harmonics
G - Index
H - Index
J-A(30) - Fourier coefficients
A(32)-A(132) - Y values
Data memory locations (A(10) to A(30) and A(32) to A(132)) compatible with Fourier Analysis program.

TECH-3

The next program, TECH-3, is an example of the applicability of the Pocket Computer to solution of commonly used

equations. Certainly other approaches could be used to solve for any of three or more variables given the rest, but the approach used for the "A" and "B" to accomplish this is simple and works well. The "C" section may be of interest, as it is a simple way to get an equivalent resistance of a number of parallel resistances (or parallel inductances without mutual coupling) or series capacitances. Note that it simply sums the conductances and then inverts the total conductance if no further values are input. The "F" program, it should be noted, does not give the correct sign of the angle (i.e. 180° off) in quadrants 3 and 4—it would not be difficult to correct this, but it may be just as well to use common sense along with the answers given by the computer.

Program: "TECH-3" - Resonance, F(3db), Parallel Resistors, Coil Q, Complex Number Conversion

OPERATION:

⟨SHFT⟩ ⟨A⟩ - Resonance

Solve for? - enter F, C or L

Inputs other two variables and displays answer, including units (kHz, pF, mFD, etc.)

Assumes $B^2 \rangle\rangle 2d$

Repeats "Solve for ?"

⟨SHFT⟩ ⟨B⟩ - RC network 3db (halfpower) frequency

Solve for ? - enter F, C, or R

Inputs other two variables and displays answer, including units.

Repeats "Solve for ?"

⟨SHFT⟩ ⟨C⟩ - Parallel Resistors.

Input "R=" - first R value. Repeat input cycle until all values entered.

Push ⟨ENTER⟩ without entering R value for answer, expressed in ohms, kilohms, or megohms as applicable.

⟨SHFT⟩ ⟨D⟩ - Coil Q.

Enter F, L, R.

Successively prints X = abs. value of reactance, coil Q, and F(BW) bandwidth assuming coil Q contributes only damping in circuit.

Repeats

⟨SHFT⟩ ⟨F⟩ - Complex Number Conversion

Input: "MAGNITUDE="
"ANGLE="
"REAL="
"IMAG=J"

Hitting ⟨ENTER⟩ without inputting a variable causes next input statement to appear, cycling through until data is entered. E.g. to find polar equiv. of 3+J4 hit ⟨ENTER⟩ twice, input Real part = 3, input Imag. part = J4

Answer (polar) displayed as MAGNITUDE/ANGLE, e.g. 5./53.13010235

Answer (rectangular) displayed as REAL+J IMAG, e.g. 3.+J4.

Returns to same type of problem on input cycle; if you want to work the other type of problem (going the other direction) just hit ⟨ENTER⟩ until either MAGNITUDE or REAL appears

NOTE: All programs above require inputs in basic units (i.e. farads, ohms, hertz, henries) (except for "F"), even if they contain conversion steps to reduce the quantity to a 'nicer' number. So be sure to take this into account when going from one program to another with one or more of the variables remaining the same in the new program. E.g. if F=1.5 and B$="MHZ" it would not be correct to use the value of F (i.e. 1.5) for 1.5 mHz in the new problem. You could enter "F=F*Exp6" before running the second program, or you could re-enter the value itself if it is simple.

```
 2:  "A"PAUSE "RESONANCE"
     : D=6
 4:  INPUT "SOLVE FOR? ";A$
 5:  GOTO D
 6:  IF A$="F" GOTO 1Ø
 7:  IF A$="L" GOTO 2Ø
 8:  IF A$="C" GOTO 3Ø
 9:  GOTO 4
 1Ø: INPUT "L=";L
 11: INPUT "C=";C
 12: F=1/92piSQRLC))
     : B$=" HZ"
 13: IF F>Exp3 LET F=F/Exp3
     : B$=" KHZ"
 14: IF F>Exp3 LET F=R/Exp3
     : B$=" MHZ"
 15: V=F
     : GOTO 2ØØ
 2Ø: INPUT "F=";F
 21: INPUT "C=";C
 22: L=1/((2piF)^2*)
     : B$=" H"
 23: IF L<1 LET L=L*Exp3
     : B$=" MH"
 24: IF L<1 LET L=L*Exp3
     : B$=" UH"
 25: V=L
     : GOTO 2ØØ
 3Ø: INPUT "F=
 31: INPUT "L=";L
 32: C=1/((2piF)^2*L)
     : B$=" FD"
 33: IF C<1 LET C=C*Exp6
     : B$=" MFD"
 34: IF C<1 LET C=C*Exp6
     : B$=" PF"
 35: V=C
     : GOTO 2ØØ
 42: "B"PAUSE "RC 3DB FREQUENCY"
     : D=46
     : GOTO 4
 46: IF A$="F"GOTO 5Ø
 47: IF A$="R"GOTO 6Ø
 48: IF A$="C"GOTO 7Ø
 49: GOTO 4
 5Ø: INPUT "R=";R
 51: INPUT "C=";C
 52: F=1/92piRC)
     : B$="HZ"
 53: IF F>Exp3 LET F=F/Exp3
     : B$="KHZ"
 54: IF F>Exp3 LET F=F/Exp3
     : B$="MHZ"
 55: V=F
     : GOTO 2ØØ
 6Ø: INPUT "F=";F
 61: INPUT "C=";C
 62: R=1/(2piFC)
     : B$="OHMS"
 63: IF R>Exp3 LET R=R/Exp3
     : B$=" K"
 64: IF R>Exp3 LET R=R/Exp3
     : B$=" MEG"
 65: V=R
     : GOTO 2ØØ
 7Ø: INPUT "F=";F
 71: INPUT "R=";R
 72: C=1/(2piFR)
     : B$=" FD"
 73: IF C<1 LET C=C*Exp6
     : B$="MFD"
 74: IF C<1 LET C=C*Exp6
     : B$=" PF"
 75: V=C
     : GOTO 2ØØ
2ØØ: PRINT USING "#######.####";" ";A$;"=";V;"
     ";B$
```

```
202: GOTO 4
300: "C"PAUSE "PARALLEL RESISTORS"
     : S=0
310: INPUT "R=";R
     : S=S+1/R
     : GOTO310
320: R=1/S
     : B$="OHMS"
330: IF R>Exp3 LET R=R/Exp3
     : B$="K"
340: IF R>Exp3 LET R=R/Exp3
     : B$="MEG"
350: PRINT USING "####.####";"R(T)=";R;" ";B$
360: GOTO 300
400: "D" PAUSE "COIL Q"
410: INPUT "F=";F
420: INPUT "L=";L
430: INPUT "R=";R
440: X=2piFL
     : Q=X/R
     : B$="OHMS"
442: IF X>Exp3 LET X=X/Exp3
     : B$="K"
444: PRINT USING "####.###";"X=";X;" ";B$
446: PRINT "Q=";Q
448: D=F/Q
     : B$="HZ"
450: IF D>Exp3 LET D=D/Exp3
     : B$="KHZ"
452: IF D>Exp3 LET D=D/Exp3
     :  B$="MHZ"
454: PRINT "F(BW)=";D;" ";B$
460: GOTO 410
500: "F" PAUSE "COMPLEX # CONV."
     : USING
510: INPUT "MAGNITUDE=";G
520: INPUT "ANGLE=";T
     : GOTO 600
530: INPUT "REAL=";H
540: INPUT "IMAG.=J";J
     : GOTO550
545: GOTO 510
550: G=SQR(HH+JJ)
     : T=ATN(J/H)
560: PRINT G;"/";T
570: GOTO 530
600: H=G*COS T
     : J=G*SIN T
610: P=SGN J
     : J=ABS J
     : B$="+"
620: IF P=-1 LET B$="-"
630: PRINT H;B$;"J";J
640: GOTO 510
```

TRF-1

In a similar vein, the TRF-1 program does the necessary manipulation of circuit parameters to design a single-tuned RF stage. "A" evaluates required inductor parameters for given bandwidth, center frequency, and capacitance, also evaluating resonant gain given transconductance and circuit parameters of an FET or pentode amplifier. "B" evaluates the bandwidth resulting when a specified inductor is used. "C" and "D" are analogous programs for bipolar transistor amplifiers with the tuned circuit at the input (as opposed to the output for "A" and "B") taking tuned circuit transformer ration into account. "F" plots amplitude (gain) and phase shift characteristics for a completed circuit. "G" is similar but finds only one value at a time whereas "F" automatically evaluates A^v and phase shift at predetermined intervals between predetermined end points. "H" evaluates the effective input admittance to a transistor stage with tapped L to aid in running programs "B" and "C." My original plan was to write an entire series of programs like this, so that double-tuned, stagger-tuned, and other amplifier configurations could be rapidly and easily designed. Amateur and other radio operators and designers would definitely find such a package useful.

Program: Single-staged single-tuned Narrow-band Amplifiers

OPERATION

⟨SHFT⟩ ⟨A⟩ - Pentode Amplifier: see Fig. 1.
Input G_m, R_p, R_L, C, f_{res}, required bandwidth BW. If resulting poles are real, pauses "Try larger C"; Input C.
Finds required L, gives A_o, $R_{(coil)}$ = equivalent series resistance of coil, $Q_{(coil)}$, circuit (loaded) Q.

⟨SHFT⟩ ⟨B⟩ - Pentode Amplifier Bandwidth
Input G_m, R_p, R_L, C, L, R_{coil}; if variable remains same as before, hit ⟨ENTER⟩.
Finds f_{res}, Bandwidth, A_o, circuit Q, Q_{coil}

⟨SHFT⟩ ⟨C⟩ - Bipolar Transistor Amplifier with Tapped L; Fig 2
Input transformer ratio a: $1 = N_s/N_p$ (less than one)
Input G_m, R_p (output resistance of previous stage, etc.), R_L (input resistance to transistor), C, f_{res}, req. BW.
Finds required L, gives A_o, finds R_{coil} allowable, Q_{coil}, circuit Q.

⟨SHFT⟩ ⟨D⟩ - Bipolar Transistor Bandwidth
Input Q_{coil} of proposed inductor, L, transformer ratio a: 1, G_m, R_p, R_L, C.
Finds f_{res}, BW, A_o, Q (circuit), Q_{coil}

⟨SHFT⟩ ⟨F⟩ - Amplifier Amplification and Phase Shift Plotter
Input F_{min} (lowest frequency in plot), F_{max}, and number of divisions desired between the end points. Prints Div # and corresponding Frequency, hitting ⟨ENTER⟩ causes it to display MAGNITUDE/PHASE SHIFT (for example 1.5556/45. = amplification 1.5556, phase 45°
Hit ⟨ENTER⟩ again and it goes to next point on plot. When done, goes to ⟨SHFT⟩ ⟨G⟩.

⟨SHFT⟩ ⟨G⟩ - Input F, Evaluates A_v and Phase Shift.
Repeat for as many different values of F desired.

⟨SHFT⟩ ⟨H⟩ - Bipolar Transistor Input Capacitance (including Miller effect) and Input Resistance (including biasing R's).
Input I_c (quiescent collector current), Beta, R_a and R_b, R(LOAD) = collector load resistance & input resistance to next stage, C_c (collector to base capacitance), F_T (current gain BW product).
Finds total input capacitance $C_e + (1 + A)C_c$, total input resistance ($= R_L$ in A-D above)
Enter "A", goes to C above, except also input "C(ext)" (external trimmers, variation swampers, circuit wiring, etc.) and sums C_{in} and C_{ext}.
Enter "B", goes to D above, except also inputs C_{ext}; See Fig. 3
NOTE: Input all data in basic units (e.g. hertz, farads, henries, ohms, amperes); e.g. to input 5.0mA. enter '.005' or 5Exp-3. All answers printed in basic units.

TRF-1 PROGRAM LINE CONTENT

```
10: "A"PAUSE "TRF-1 PENTODE
    : D=60
    : J=1
20: INPUT "G(M)=";G
30: INPUT "R(P)="O
    : O=OJJ
40: INPUT "R(L)=";N
50: INPUT "C=";C
55: GOTO D
60: INPUT "F(0)=" ;F
    : W=2piF
```

```
62: INPUT "BW=";B
64: GOTO 75
7Ø: INPUT "L=";L
72: INPUT "R(COIL)=";P
74: GOTO 16Ø
75: IF (1/O+1/N)=Ø LET S=9Exp99
    : GOTO 8Ø
77: S=1/(1/O+1/N)
8Ø: R=JJ/(2piBC)
    : IF R>=S PAUSE "TRY LGR C"
    : INPUT "C=";C
    : GOTO 8Ø
9Ø: L=1/(WWC)
95: IF S=9Exp99 LET P=L/(CR)
    : GOTO 1ØØ
97: P=RS/(S-R)
    : P=P/(JJ)
    : P=L/(CP)
1ØØ: A=GR/J
     : Q=WL/P
     : K=R/(JJWL)
11Ø: PRINT "L=";L
12Ø: PRINT "A(O)=";A
13Ø: PRINT "R(COIL)=";P
135: PRINT "Q(COIL)=";Q
14Ø: PRINT        "Q=";K
     : END
15Ø: "B" J=1
     : D=7Ø
     : PAUSE "PENTODE BW"GOTO 2Ø
16Ø: W=1/SQR(LC)
     : F=W/2pi
17Ø: IF 1/C+1/N=Ø LET S=9Exp99
     : GOTO 19Ø
18Ø: S=1/(1/O+1/N)
     : IF D=16Ø LET R=1/91/S+1/(JJQWL))
     : P=WL/Q
     : GOTO 195
19Ø: R=1/(1/S+1/L(CP)))
195: B=JJ(2piRC)
     : A=GR/J
     : K=R/(JJWL)
     : Q=WL/P
2ØØ: PRINT USING;"F(O)=";F
21Ø: PRINT "BW=";B
22Ø: PRINT "A(O)=";A
23Ø: PRINT "Q=";K
24Ø: PRINT "Q(COIL)=";Q
     : END
25Ø: "F" PAUSE "A(V) VS F"
26Ø: INPUT "F(MIN)=";D
27Ø: INPUT "F(MAX)=";E
28Ø: INPUT "# DIVS=";H
     : H=(E-D)/H
     : I=Ø
29Ø: W=2piD
     : DEGREE
3ØØ: X=1/(LC)-WW
     : Y=WJJ/(RC)
     : V=SQR(XX+YY)
     : IF X=O LET X=Exp-5Ø
3Ø5: U=ATN(Y/X)
     : U=9Ø-U
31Ø: V=(WJG/C)/V
     : IF X<O LET U=U-18Ø
312: IF I=-1 GOTO32Ø
315: PRINT USING "####";"DIV #";I;USING;" F=";D
32Ø: PRINT V;"/";U
     : IF I=-1 GOTO 35Ø
33Ø: D=D+H
     : I=I+1
     : IF D>E GOTO 35Ø
34Ø: GOTO 29Ø
35Ø: "G" INPUT "F=";D
355: I=-1
     : GOTO 29Ø
36Ø: "C"PAUSE "TRF-1 BIPOLAR"
     : D=6Ø
37Ø: INPUT "RATIO=";J
38Ø: GOTO 2Ø
39Ø: "D"PAUSE "BIPOLAR BW"
     : D=16Ø
     : INPUT"Q(COIL)=";Q
4ØØ: INPUT "L=";L
41Ø: GOTO 37Ø
42Ø: "H"PAUSE "INPUT C + R"
43Ø: INPUT "I(C)=";G
     : G=4ØG
44Ø: INPUT "BETA=";T
45Ø: INPUT "R(A)=";M
     : N=1/M
46Ø: INPUT "R(B)=";M
     : N=1/(N+1/M+G/T)
47Ø: INPUT "R(LOAD)=";Z
48Ø: INPUT "C(C)=";C
49Ø: INPUT "F(T)=";E
     : C=C*(1+GZ)+G/2piE
5ØØ: PRINT "C(IN)=";C
51Ø: PRINT "R(IN)=";N
52Ø: INPUT "FIND Q (A) OR BW (B)";M$
     : GOTO 54Ø
53Ø: END
54Ø: INPUT "RATIO=";J
     : C=CJJ
55Ø: INPUT "R(P)=";O
     : O=OJJ
552: PRINT "C(REFL)=";C
555: INPUT "C(EXT)=";Z
     : C=C+Z
56Ø: IF M$="A" LET D=6Ø
57Ø: IF M$="B" INPUT "Q(COIL)=";Q
     : D=16Ø
58Ø: GOTO D
```

FIXED MEMORY LOCATIONS

A - A(o) Amplification at resonance (A and B - A_v; C and D - A_c)
B - Bandwidth, Hz.
C - Capacitance C, Farads
D - Line tag (A,B,C,D,H), Frequency variable F (F and G)
E - F(MAX) (F only), F_T (H only)
F - F_o, resonant frequency, Hz.
G - G_m (H only - also temporarily I_c)
H - Increment in F (F only)
I - Index (F only), jump variable (G only)
J - Transformer turns ratio a : 1 (A and B - set equal to 1)
K - Circuit Q
L - Inductance, Hys.
M - R_a and R_b; used to evaluate $R_{(IN)}$ (H only), M$ = "A" or "B" (H only)
N - (A and B - R(L) = input resistance to next stage), (C,D,H - input resistance to transistor base incl. biasing nets)
O - Parallel external resistance (A and B R(P) = plate resistance, C&D R(P) = output resistance of previous stage). Reflected to secondary value.
P - Coil resistance (series) R_{coil}
Q - Coil Q
R - Total equivalent parallel resistance R'
S - R(P) paralleled with R(L) $(1/o + 1/N)^{-1}$
T - Beta (H only)
U - Phase ∠ of amplification (F and G only)
V - Amplitude of amplification (F and G only)
W - Angular resonant frequency
X - Re(A_v (F&G only)
Y - Im(A_v) (F&G only)
Z - R(LOAD), C(EXT) (H only)

VARIABLE MEMORIES - none used

SILINEQ

Finally, there is SILINEQ, which solves second, third or fourth order determinants in order to solve simultaneous linear equations such as would be obtained from mesh-current or node-voltage circuit analysis, or any such set of simultaneous linear equations. Though there are a number of approaches that could be used which could use less memory space and/or negotiate higher than fourth-order matrices, this one has the advantage that since it evaluates the determinants directly the entire process is very much faster than if they were evaluated using FOR-NEXT or GOSUB-RETURN loops; it takes only about 20 seconds from the time the last parameter is input until the answers appear for a fourth-order set of simultaneous equations. Seldom does the need arise for higher than fourth-order solutions, so I have not devoted much attention to this aspect.

```
1: "A" PRINT "SIMULTANEOUS"
     : PRINT "LINEAR"
     : PRINT "EQUATIONS"
     : PRINT "BY F. NACHBAUR"
     : CLEAR
2Ø: PRINT " "
     : INPUT "ORDER (2,3,OR 4)";Z
3Ø: IF Z=2GOTO 1Ø1
4Ø: IF Z=3GØTØ 1Ø1
5Ø: IF Z=4GOTO 1Ø1
6Ø: BEEP 1
     : PAUSE "ERROR"
     : GOTO 2Ø
1Ø1: INPUT "H(11)=";A
     : INPUT" H(12)=";B
1Ø3: IF Z=2GOTO 1Ø7
1Ø4: INPUT "H(13)=";C
1Ø5: IF Z=3GOTO 1Ø7
1Ø6: INPUT "H(14)=";D
1Ø7: INPUT "H(21)=";E
     : INPUT"H(22)=";F
1Ø9: IF Z=2GOTO 122
11Ø: INPUT "H(23)=";G
111: IF Z=3GOTO 113
112: INPUT "H(24)=";H
113: INPUT "H(31)=";I
     : INPUT"H(32)=";J
     : INPUT "H(33)=";K
116: IF Z=3GOTO 122
117: INPUT "H(34)=";L
     : INPUT"H(41)=";M
     : INPUT "H(42)=";N
     : INPUT "H(43)=";Q
     : INPUT "H(44)=";P
122: INPUT "Y(1)=";Q
     : INPUT "Y(2)=";R
124: IF Z=2GOTO 6ØØ
125: INPUT "Y(3)=";S
126: IF Z=3GOTO 7ØØ
127: INPUT "Y(4)=";T
13Ø: U=A*(FKP+LGN+JOH-HKN-LOF-JGP)-E*(BKP+JOD
     +NLC-DKN-LOB- CJP)
14Ø: U=U+I*(BGP+CHN+FOD-DGN-HOB-FCP)-M*(BGL+CHJ
     +FDK-JGD-KHB- FCL)
15Ø: V=Q*(FKP+LGN+JOH-HKN-FOL-JGP)-R*(BKP+LCN
     +JOD-DKN-LOB- CJP)
16Ø: V=V+S*(BGP+CHN+FOD-DGN-HOB-FCP)-T*(BGL
     +CHJ+DFK-DGJ-HKB- FCL)
     : V=V/U
17Ø: W=A*(RKP+GLT+SOH-HKT-ROL-GSP)-E*(QKP+CLT
     +SOD-DKT-QOL- CSP)
18Ø: W=W+I*(QGP+CHT+ROD-DGT-QOH-CRP)-M*(QGL+CHS
     +RKD-GDS-QKL-RCL)
     : W=W/U
19Ø: X=A*(FSP+LRN+JTH-HSN-FLT-JRP)-E*(BSP+LQN
     +JTD-DSN-BLT- QJP)
2ØØ: X=X+I*(BRP+QHN+FTD-DRN-BHT-QFP)-M*(BRL+QHJ
     +FSD-DRJ-BHS-QFL)
     : X=X/U
21Ø: Y=A*(FKT+GSN+JOR-RKN-FSO-GJT)-E*(BKT+CSN
     +JOQ-QKN-BSO-CJT)
22Ø: Y=Y+I*(BGT+CRN+FOQ-QGN-BRO-CFT)-M*(BGS+
     CRJ+QFK-QGJ-BRK- CFS)
     : Y=Y/U
     : GOTO 75Ø
6ØØ: U=A*F-B*E
     : V=Q*F-R*B
     : W=R*A-Q*E
     : V=V/U
     : W=W/U
     : GOTO 75Ø
7ØØ: U=A*(FK-GJ)-B*(EK-GI)+C*(EJ-FI)
     : V=Q*(FK-JG)-R*(BK-CJ)+ S*(BG-FC)
     : V=V/U
72Ø: W=-Q*(EK-GI)+R*(AK-CI)-S*(AG-CE)
     : W=W/U
     : X=Q*(EJ-FI)- R*(AJ-BI)+S*(AF-BE)
     : X=X/U
75Ø: BEEP Z
     : PRINT "X(1)="
     : PRINT V
76Ø: PRINT "X(2)="
     : PRINT W
     : IF Z=2GOTO 2Ø
77Ø: PRINT "X(3)="
     : PRINT X
     : IF Z=3GOTO 2Ø
78Ø: PRINT "X(4)="
     : PRINT Y
     : GOTO 2Ø
```

"pi" should be replaced in all of the above programs with the pi symbol on the pocket computer (SHFT UP ARROW) or the value 3.141592654.

Pocket Computer Bugs, Errors and Fixes

ELECTRICAL ENGINEERING I (26-3520)

Corrections to the Manual

On page 7 under the heading **You type:**

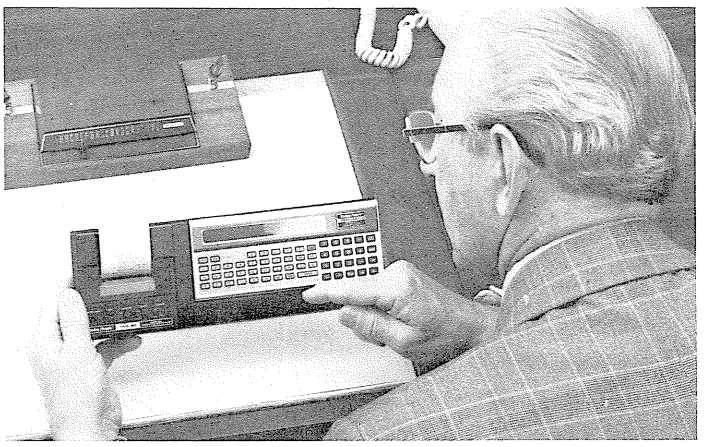
SHFT A (plus)

should be moved up one row so that the line reads:

66.4250693 -76.32851033 SHFT A (plus)

On the same page toward the bottom right hand side, the number 55.99999996 should be replaced by the number 56.0000556.

On page 12 the program name AMPEDS is incorrect. It should be AMPDES.



Convert Integer to Binary

Marvin E. Wilborne, III
Rt. 5, Box 314
Danville, VA 24540

Here is a pocket computer program that converts a positive integer from 0 to 1023 into binary, using successive division.

```
1Ø: "A"
     : PAUSE "DECIMAL TO BINARY."
2Ø: INPUT "ENTER DECIMAL NUMBER.", A
3Ø: IF A<Ø THEN 2Ø
4Ø: IF A>1Ø23 THEN 2Ø
5Ø: IF A<>INT(A) THEN 2Ø
6Ø: B=Ø
     : C=Ø
     : Z=A
7Ø: D=A/2
     : E=D-INT(D)
8Ø: F=E*2*(1Ø^B)
9Ø: C=C+F
1ØØ: IF D<1 THEN 13Ø
11Ø: B=B+1
12Ø: A=INT(D)
     : GOTO 7Ø
13Ø: PRINT Z; " = "; C
14Ø: INPUT "ENTER <Y> TO CONTINUE.", G$
15Ø: IF G$="Y" THEN 2Ø
16Ø: END
```

To use the program, press ⟨SHFT⟩ ⟨A⟩ in the DEF mode. The computer will display "Decimal to Binary," and it will then ask for a number (0-1023). For the number 1023 it takes approximately 11 seconds to get the binary equivalent.

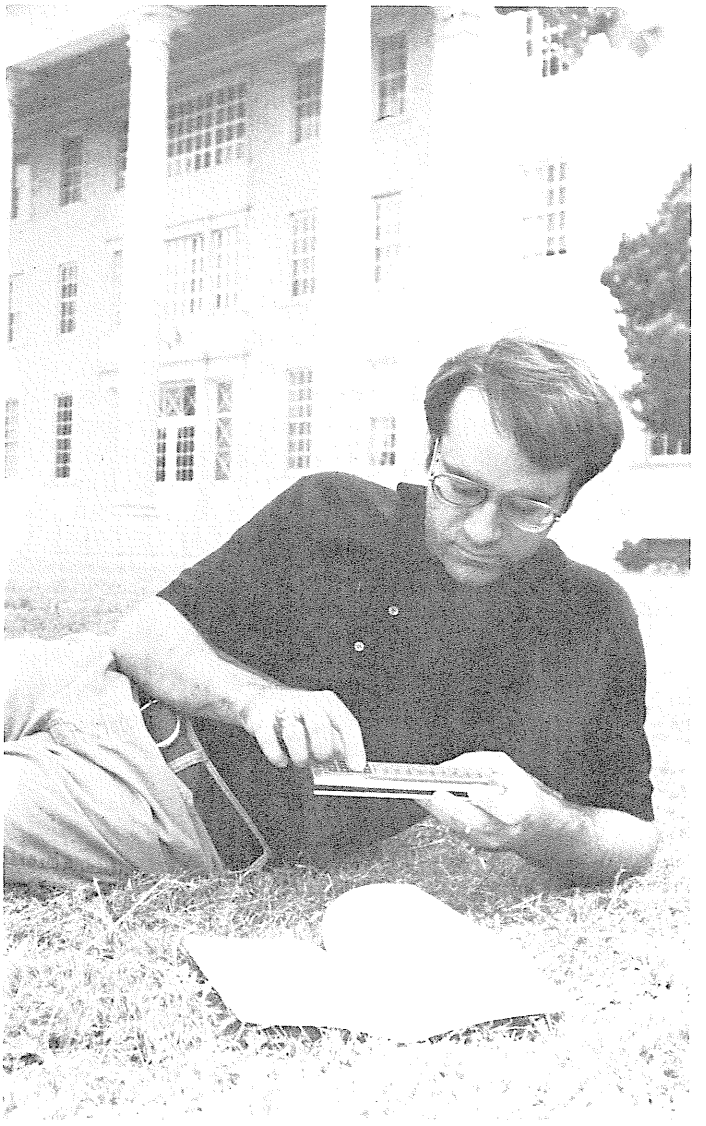
Here is an alternate method of conversion:

You enter a positive integer from 0 - 65535 and the binary equivalent is printed.

The program works fine on the printer as well as the LCD display.

```
1Ø: "A" INPUT "ENTER DECIMAL NUMBER", B
2Ø: IF B<Ø THEN 1Ø
3Ø: IF B>65535 THEN 1Ø
4Ø: IF B<>INT (B) THEN 1Ø
5Ø: C=INT (B/256)
6Ø: D=(B/256-C)*256
7Ø: E=INT (C/16)
8Ø: F=(C/16-E)*16
9Ø: G=INT (D/16)
1ØØ: H=(D/16-G)*16+1
     : E=E+1
     : F=F+1
     : G=G+1
11Ø: GOSUB E*1Ø+2ØØ
12Ø: I$=A$
13Ø: GOSUB F*1Ø+2ØØ
14Ø: J$=A$
15Ø: GOSUB G*1Ø+2ØØ
16Ø: K$=A$
17Ø: GOSUB H*1Ø+2ØØ
18Ø: PRINT B; " DECIMAL ="
19Ø: PRINT I$; J$; K$; A$
2ØØ: END
21Ø: A$="ØØØØ"
     : RETURN
22Ø: A$="ØØØ1"
     : RETURN
23Ø: A$="ØØ1Ø"
     : RETURN
24Ø: A$="ØØ11"
     : RETURN
25Ø: A$="Ø1ØØ"
     : RETURN
26Ø: A$="Ø1Ø1"
     : RETURN
27Ø: A$="Ø11Ø"
     : RETURN
28Ø: A$="Ø111"
     : RETURN
29Ø: A$="1ØØØ"
     : RETURN
3ØØ: A$="1ØØ1"
     : RETURN
31Ø: A$="1Ø1Ø"
     : RETURN
32Ø: A$="1Ø11"
     : RETURN
33Ø: A$="11ØØ"
     : RETURN
34Ø: A$="11Ø1"
     : RETURN
35Ø: A$="111Ø"
     : RETURN
36Ø: A$="1111"
     : RETURN
```

```
65535. DECIMAL = 1111111111111111
256. DECIMAL = ØØØØØØØ1ØØØØØØØØ
255. DECIMAL = ØØØØØØØØ11111111
65535. DECIMAL = 1111111111111111
```



Answers to Interesting Questions

With the new year well under way, it's probably time to do a little house cleaning. I have received some interesting questions from Color Computer users and over the next couple of months, I'm going to attempt to answer some of the more general ones.

"How do you save Assembly Language Programs?"

Toward the end of 1981, I read numerous articles about saving assembly language programs using Extended Color BASIC and Disk Extended BASIC. Most of what was said was that using Hexadecimal numbers would not work. To my knowledge, this is not true. The correct format for saving assembly language programs using Extended BASIC is as follows:

CSAVEM "filename," &Hstart, &Hend, &Hexecute

"filename" is an eight character maximum name for your program.

&Hstart is the hexadecimal starting address for your program.

&Hend is the hexadecimal ending address for your program.

&Hexecute is the transfer location or beginning of the actual program.

Notice that when using hexadecimal numbers, the &H is required. This tells the computer that the following values are in Hex.

Extended Color BASIC is required to save a program in this fashion.

With the Disk system, the format is virtually the same:

SAVEM "filename",&Hstart,&Hend,&Hexecute

Again, the &H is necessary to tell the computer that the numbers are in Hex.

When saving assembly language programs using the Disk system, there is one additional thing to keep in mind. If your program resides in low memory, with starting addresses around &H1000 and you save it to disk, when you try to LOADM the program from disk, you'll probably get an I/O error. The way to get around it is when loading the program from tape (assuming it was on tape in the first place), use the offset capability of the CC to move the program up in memory. I have found that about 2050 bytes (&H802)is usually enough. Then save the program using the new offset addresses to disk. (Don't forget to extend the ending address also.) The reason is this: when the machine powers up under Disk Extended BASIC, the ROM grabs about 2K of free memory, just beyond Video RAM (usually located at &H400—&H600). If you've PCLEARed more memory for graphics, the disk buffer will be placed above those graphics "pages." If your program starts in this area of memory, you could be saving the data buffer for the disk along with your program. When you try to load the program back into the machine, the information that was saved from the data buffer garbles up your program, giving you an I/O error. As reports go, I cannot say whether saving programs from low memory will work any better using decimal (base 10) numbers instead. I usually work with Hex when saving programs (on either tape or disk).

"Is it possible to put on a second set of joysticks?"

In reference to some inquiries about adding additional joysticks (more than two at a time), there are no plans for the near future to offer a modification and/or "black box" to enable more than two joysticks to be attached to the Color Computer. Such a modification would obsolete current software (in excess of 27 Program Paks, most of which offer joystick option) and would probably require internal changes in the Color Computer structure itself (ROMs, I/O routines, etc.). This is not to say that it cannot be done, simply that Radio Shack has no plans to do it in the foreseeable future.

"Is there any way through an outside device that both sound and graphics operate at the same time? If so what?"

Precluding any after-market ideas, this is not that feasible. The sound is controlled by one of the Input/Output (I/O) chips in the Color Computer. The sound can originate from four different sources and is processed through the Digital to Analog (D/A) multiplexer. To enable an external device to be attached would require an additional I/O chip. Since the Central Processing Unit (CPU) and the dynamic RAM controller (SAM) control what is going on inside the computer, to have simultaneous sound and action would require changes in both the CPU and SAM chips. Again, Radio Shack has no plans in the foreseeable future for this type of modification. As far as an external device is concerned, if the necessary logic were to be placed inside the "external box" and control from the machine was passed to that "box," it probably could be done. (It would probably carry a heavy price tag too!)

"How might a menu be set up using DSKI$ to list all the programs in the directory and have them run on request?"

On page 62 of the Color Computer Disk System Owners Manual and Programming Guide is a program listing which you can modify to list the "menu" from which you can select the program you wish to run. The program is as follows:

```
5 CLEAR 1ØØØ
1Ø FOR X=3 TO 11
2Ø DSKIS Ø,17,X,A$,B$
3Ø C$= A$ + LEFT$(B$,127)
4Ø NAM$(Ø) = LEFT$(C$,8)
5Ø EXT$(Ø) = MID$(C$,9,3)
6Ø FOR N=1 TO 7
7Ø NAM$(N) = MID$(C$,N*32+1,8)
8Ø EXT$(N) = MID$(C$,9+N*32,3)
9Ø NEXT N
1ØØ FOR N=Ø TO 7
11Ø IF EXT$(N) = "DAT" AND
    LEFT$(NAM$(N),1)<>CHR$(Ø) THEN PRINT
    NAM$(N)
12Ø NEXT N
13Ø NEXT X
```

The format for the DSKI$ command, which allows direct reading of disk sectors from Basic, is in line 20. Its syntax is as follows:

DSKI$ drive #, track #, sector #, string variable #1, string variable #2

The drive # can be 0, 1, 2, or 3 (remember you can have up to four drives on the Disk system). You can read any of the 35 tracks (numbered 0 to 34, with the directory track being in the middle at track 17). Each sector can contain 256 bytes of data. The string variable #1 reads the first 128 bytes while string variable #2 reads the second 128 bytes.

The sectors this program is looking at are specified in line 10. Since the operating system uses the first 3 sectors of track 17 for internal disk housekeeping, (0, 1, & 2), there won't be any programs listed in these three sectors, so there is no need to search them for programs.

Lines 30 through 90 combine string #1 & #2 and break it down into the various program NAMes and EXTensions.

Lines 100 through 120 check for any filenames with an EXTension "DAT".

For your menu program, you would not need to separate the filenames and extensions. You also would not need to search for a specific EXTension. As you read in the program names/extensions, you could dimension an array, telling the program to print out the listing on the screen next to the corresponding array element where the program is stored. At the end of your list, you could then use the INKEY$ to pick one of your programs to run. The instruction could be something like:

```
1ØØ RUN A$(X)
```

Where A$ is the name of the array and X is the cell in the array.

If you have an assembly language program in your directory, you would need to examine the EXTension for the BIN designation and use a GOTO statement like the following:

```
99 IF MID$(A$(X),9,3)="BIN" THEN GOTO
   1ØØØ
1ØØØ LOADM A$(X)
```

Author's Note: Since my CC is otherwise occupied at the moment, I cannot tell you whether you will need to insert a "/" between the program name and the extension or whether it is saved in that fashion. Be sure to check it out before running the program.

"How can passwords be put on disk programs?"

Since the operating system was designed to be "invisible" to the user (so you would not have to learn a whole new language), the provision for password protection was not included in the Disk instruction set. Following that idea, since there is not a password-protect for tape-based programs, one was not included for disk-based programs either. This is not to say that it is impossible to do, only that our disk system does not offer a command (or subcommand) which supports password protection.

As far as protecting your programs, there are a number of ways that can be tried. As with "burglar alarms," the best protection is that made up by the programmer (unique to that program or series of programs). You can try saving a short program on the master diskette which must be loaded from the program before it will continue. Or you can try PEEKing at a location in memory that you have POKEd a value into prior to loading the program before the program will continue. (Neither of these solutions will prevent you from loading the program though.) You might try setting up your own directory (using the DSKI$/DSKO$ commands to read/write directly to the diskette without updating the directory. Here, you would need to remember where you put what, how long it is, etc., not to mention the need to convert each of your program lines to strings. Like I said, it's not impossible and with a little ingenuity, you could develop your own security system for your disk files/programs.

High Resolution Graphics with 4K

Mark Ayen
5 Glenvale Avenue
Darien, CTR 06820

Not many people who own 4K Color Computers with Color BASIC realize that they can use high resolution graphics, too. The program I have enclosed makes use of these graphics and is interesting to fool around with. The program, Sine I, plots a sine wave using a routine readily adaptable to other purposes.

To change the frequency of the wave, change the number '20' in line 90 to anything from 5 to 25. To thicken and define the wave better, change line 160 to:

```
16Ø BYTE=3Ø88: BIT=128: GOTO 7Ø
```

Does that not look better?

```
1 '
2 '
3 ' Sine I
4 ' by MARK AYEN
5 ' 1981
6 '
7 '
1Ø CLEAR 1Ø,3Ø71
2Ø BYTE=3Ø72
     : BIT=128
3Ø FOR X=3Ø72 TO 4Ø95
     : POKE X,Ø
     : NEXT
4Ø FOR X=Ø TO 9
     : READ M
     : POKE 65472+X*2+M,Ø
     : NEXT
5Ø DATA 1,Ø,Ø,Ø,1,1,Ø,Ø,Ø,Ø
6Ø POKE 65314,14Ø
7Ø FOR X=Ø TO 6.2832 STEP .1
8Ø V1=SIN(X)
9Ø V=V1*2Ø+32
1ØØ B=BYTE+INT(V+.5)*16
11Ø POKE B,(PEEK(B) OR BIT)
12Ø BIT=BIT/4
13Ø IF BIT<2 THEN BIT=128
     : BYTE=BYTE+1
14Ø NEXT
15Ø FOR X=3568 TO 3583
     : POKE X,255
     : NEXT
16Ø GOTO 16Ø
```

Sub Destroyer

R. Bruce Nagel
100 Monteray Avenue
Dayton, OH 45419

The object of this game is to destroy ten submarines per round in the least amount of time. This is accomplished by dropping depth charges from a ship on the surface of the water.

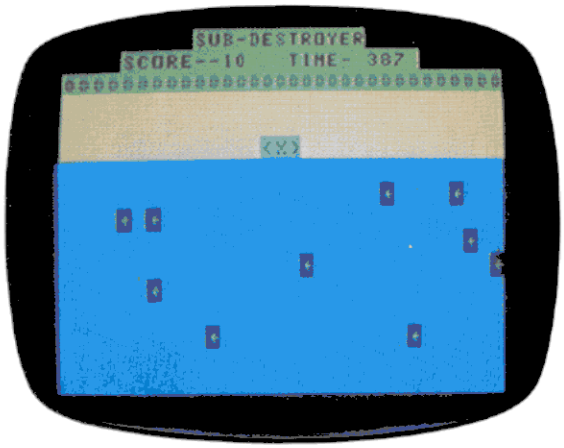
When all ten submarines are destroyed, a new round of ten submarines will begin. Each submarine destroyed is worth ten points, but if a submarine is allowed to surface (reach the top left corner of the ocean), five points are deducted, and the submarine returns to the ocean floor.

HOW TO PLAY

The player controls the movement of the ship and the dropping of the depth charges. To move the ship, the keys ⟨G⟩ (left) and ⟨H⟩ (right) are used. Pressing ⟨F⟩ drops a depth charge. There is no limit on the amount of shots which may be fired, yet the ship cannot move or fire until the charge hits the submarine or the ocean floor. If a charge hits a submarine, ten points are awarded and a beep is heard. There is no penalty for missing.

RULES AND STRATEGIES

Each game is five hundred time units in length. When a depth charge is underway, the clock stops. If a key is pressed while the clock is stopped, the computer will remember and carry it out when time resumes. If a submarine reaches the top left corner, it returns to the bottom of the screen causing five points and, more costly, time to be lost. Each submarine is placed randomly in the ocean each round, but the ship remains where it was at the end of the previous round.




```
5     REM SUB-DESTROYER
10    REM BY R.BRUCE NAGEL 7/81
20    PMODE 0,1
      : PCLEAR 1
      : CLEAR 100
30    K=176
      : QQ=501
40    DIM P(9)
50    CLS 3
      : PRINT STRING$(9,144)
60    PRINT @9,"SUB-DESTROYER";
70    PRINT STRING$(42,144);
75    PRINT STRING$(32,"#");
80    PRINT STRING$(96,159);
90    PRINT @160,STRING$(32,159);
      :  PRINT @K,"<X>";
100   FOR X=0 TO 9
110   P(X)=RND(288)+221
120   IF X=0 THEN 140
130   FOR Q=0TO X-1
      : IF P(X)=>P(Q)+2 OR P(X)=<P(Q)-2
        THEN NEXT Q ELSE 110
140 PRINT @ P(X),CHR$(127);
150 NEXT X
160 FOR X=0 TO 9
170 IF P(X)=0 THEN 220
180 P(X)=P(X)-1
190 IF P(X)<192 THEN 240
200 PRINT @ P(X)+1,CHR$(175);
210 PRINT @ P(X),CHR$(127);
220 NEXT X
230 IF A$<>"F" THEN 250 ELSE RETURN
240 PRINT @P(X)+1,CHR$(175);
      : XX=XX-5
      : P(X)=507
      : GOTO 200
250 A$=INKEY$
260 QQ=QQ-1
      : PRINT @36,"SCORE-"XX, "TIME-" QQ;
270 IF QQ=0 THEN 530
280 IF A$="G" THEN K=K-2
290 IF A$="H" THEN G=G+2
300 IF A$="F" THEN GOSUB 360
310 IF K<160 THEN K=160
320 IF K>189 THEN K=189
330 PRINT @ 160,STRING$(32,159);
      :  PRINT@K,"<X>";
340 GOSUB 160
350 GOTO 250
360 II=K
370 FOR I=0 TO 9
380 II=II+32
390 IF II>511 THEN 490
400 GOSUB 160
410 PRINT @ II,"I";
420 FOR T=0 TO 9
430 IF P(T)=II THEN 440 ELSE 460
440 P(T)=0
      : XX=XX+10
      : Z=Z+1:PRINT @ II,CHR$(175);
      : SOUND 10,1
450 IF Z=10 THEN 500 ELSE 490
460 NEXT T
470 PRINT @ II,CHR$(175);
480 NEXT I
490 A$=""
      : GOTO 160
500 Z=0
510 CLS 0
      : PRINT @ 231,"ALL SUBS DESTROYED!!"
515 PRINT @ 6,"SCORE-" XX "TIME-"QQ;
520 FOR TM=1 TO 15
      : PLAY"T30CDEFGAB"
      : NEXT TM
      : GOTO 50'
530 PRINT @ 72,"TIME HAS EXPIRED########";
540 PRINT @ 96,"RATING-";
550 IF XX<100 THEN PRINT "STICK TO RAFTS"
      : END
560 IF XX<200 THEN PRINT "SAFE IN A ROWBOAT"
      : END
570 IF XX<300 THEN PRINT "DINGHY PILOT"
      : END
580 IF XX<400 THEN PRINT "NAVY MATERIAL"
      : END
590 IF XX<500 THEN PRINT "P.T.CREWMAN"
      : END
600 IF XX<600 THEN PRINT "DESTROYER CAPTAIN"
      : END
610 IF XX<700 THEN PRINT "FLEET COMMANDER"
      : END
620 IF XX<800 THEN PRINT "ADMIRAL OF THE NAVY"
      : END
630 IF XX<900 THEN PRINT "SECRETARY OF THE NAVY"
      : END
640 PRINT "COMMANDER-IN-CHIEF"
      : END
```

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

ALABAMA

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

ARIZONA

PHOENIX 10233 Metro Pkwy. E. (602) 861-1124; 4301 N. 7th Ave. (602) 277-3031
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
DOWNEY 8031 Florence Ave. (213) 927-7882
ESCONDIDO 347 W. Mission Ave. (714) 741-6032
FRESNO Princeton S/C. 2721 N. Blackstone Ave. (209) 225-5551
GLENDALE 236 N. Brand Blvd. (213) 246-9310
HAYWARD 20942 Mission Blvd. (415) 278-2888
LAKEWOOD 5830 Lakewood Blvd. (213) 920-9671
LA MESA 5346 Jackson Dr. (714) 460-3610
LONG BEACH 2119 Bellflower Blvd. (213) 597-3377
MONTEREY 484 Washington St. (408) 375-8430
MOUNTAIN VIEW 1933 El Camino Real W. (415) 961-0542
PASADENA 575 S. Lake Ave. (213) 449-5424
RIVERSIDE 3844 La Sierra Ave. (714) 689-0340
SACRAMENTO 4749 J St. (916) 454-3287
SAN BERNARDINO 764 Inland Center Dr. (714) 884-6871
SAN DIEGO 3062 Clairemont Dr. (714) 276-6050
SAN FRANCISCO One Market Place. (415) 777-9810
SAN JOSE 1228 S. Bascom Ave. (408) 297-2603
SAN MATEO 3180 Campus Dr. (415) 573-8607
SANTA BARBARA 4141 State St. A-1. (805) 967-4538
SANTA FE SPRINGS 14138 East Firestone Blvd. (213) 921-0702
STOCKTON College Sq. S/C. 963 West March Lane. (209) 957-3676
TORRANCE 3840 Sepulveda at Hawthorne. (213) 373-0306
VENTURA 4005 E. Main St. (805) 654-0196
WEST COVINA 2516 E. Workman St. (213) 915-5791

COLORADO

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

CONNECTICUT

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

DELAWARE

WILMINGTON 3847 Kirkwood Hwy. (302) 999-0193

DISTRICT OF COLUMBIA

WASHINGTON 1800 M St. NW. (202) 822-3933

FLORIDA

ALTAMONTE SPRINGS 766 B. East Altamonte Dr. (305) 339-1313
CLEARWATER 2460-D US 19 North. (813) 797-3223
HOLLYWOOD 429 S. State Rd. #7. (305) 966-4382
JACKSONVILLE 8252 Arlington Expwy. (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
N. MIAMI BEACH The Promenade. 1777 N.E. 163rd St. (305) 940-6887
ORLANDO 1238 E. Colonial Dr. (305) 894-0570
ST. PETERSBURG 3451 66th St. N. (813) 381-2366
SARASOTA 5251 S. Tamiami Tr. (Hwy 41). (813) 923-4721
TALLAHASSEE 2529 S. Adams. (904) 222-4440
TAMPA 4555 W. Kennedy. (813) 879-7470; 1825 E. Fowler Ave. (813) 971-1130
W. PALM BEACH 2271-A Palm Beach Lakes Blvd. (305) 683-3100

GEORGIA

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

IDAHO

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

ILLINOIS

AURORA 890 North Lake St. (312) 844-2224
CHICAGO 4355 S. Archer Ave. (312) 376-7617; CNA Plaza. 309 S. Wabash. (312) 922-0536
ELMWOOD PARK 7212 W. Grand Ave. (312) 452-7500
FAIRVIEW HEIGHTS #4 Market Place. (618) 398-6410
HOMEWOOD/GLENWOOD 329 Glenwood Lansing. (312) 758-0440
LaGRANGE One S. LaGrange Rd. (312) 482-3484
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

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

IOWA

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

KANSAS

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

KENTUCKY

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

LOUISIANA

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

MAINE

BANGOR Maine Square. (207) 945-6491

MARYLAND

BALTIMORE 7942 Belair Rd., Putty Hill Plaza. (301) 882-9583
CATONSVILLE One Mile West S/C. 6600 B Balt Nat'l Pike. (301) 788-3277
FREDERICK Shoppers World, Rt. 40W. (301) 695-8440
ROCKVILLE Congressional Plaza. 1673 Rockville Pike. (301) 984-0424
SALISBURY Shoppers World S/C. Rt. 50. (301) 546-9223

MASSACHUSETTS

BOSTON 730 Commonwealth Ave. (617) 739-1704
BRAINTREE South Shore Plaza. 250 Granite St. (617) 848-9290
BROCKTON 675 Belmont. (617) 583-2270
CAMBRIDGE Harvard Square. 28 Boylston St. (617) 354-7594
CHESTNUT HILL 200 Boylston St. (617) 969-2031
NATICK 1400 Worcester Rd. (617) 875-8721
SAUGUS 343 Broadway. (617) 233-4985
SPRINGFIELD 1985 Main St., Northgate Plz. (413) 732-4745
WORCESTER Lincoln Plaza. (617) 852-8844

MICHIGAN

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

MINNESOTA

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

MISSISSIPPI

JACKSON 979 Ellis Ave. (601) 352-5001

MISSOURI

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

NEBRASKA

OMAHA 3006 Dodge St. (402) 346-4003

NEVADA

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

NEW HAMPSHIRE

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

NEW JERSEY

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; 139 E. 42nd St. (212) 953-6050
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
UTICA Riverside Mall. (315) 735-1933

NORTH CAROLINA

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

OHIO

AKRON Fairlawn Plaza. 2727 W. Market St. (216) 836-9303
CANTON 5248 Dressler Rd. NW. (216) 494-7230; Meliet 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
DAYTON Northwest Plaza. 3279 West Siebenthaler. (513) 277-6500
NORTH OLMSTED Great Northern S/C. (216) 734-2255
TOLEDO 5844 W. Central Ave. (419) 531-5797
YOUNGSTOWN Union Square Plaza. 2543 Belmont Ave. (216) 744-4541

OKLAHOMA

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

OREGON

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

PENNSYLVANIA

ALLENTOWN Crest Plaza S/C. Cedar Crest Blvd. US 22. (215) 395-7155
BALA CYNWYD 67 E. City Line Ave. (215) 668-9950
ERIE 5755 Peach St. (814) 868-5541
HARRISBURG Union Deposit Mall, Union Deposit Rd. #17. (717) 564-6753
LANCASTER Park City Plaza, US 30. (717) 393-5817
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; 1517 Main St. (214) 760-8601
EL PASO 9515 Gateway West. (915) 594-8211
FT. WORTH 15 One Tandy Center. (817) 335-7198; 2801 Alta Mere. (817) 738-0251
HOUSTON 211C-FM 1960. (713) 444-7006; 10543 Gulf Fwy. (713) 943-9310; 5900 North Fwy. (713) 699-1932; 6813 SW Fwy. (713) 777-7907; 809 Dallas St. (713) 651-3002
HURST Northeast Mall. (817) 284-1518
LUBBOCK 3625 34th St. (806) 793-1467
ODESSA 1613 "A" East 8th Street. (915) 334-8355
RICHARDSON Fleetwood Sq. S/C. 202 W. Campbell Rd. (214) 669-1494
SAN ANTONIO 6018 West Ave. (512) 344-8792; 4249 Centergate. (512) 657-3958

UTAH

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

VIRGINIA

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

WASHINGTON

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

WEST VIRGINIA

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

WISCONSIN

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

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