Radio Shack

TRS-80™ Microcomputer NEWS

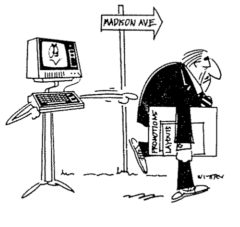
June, 1981

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

THE MICROCOMPUTER NEWSLETTER PUBLISHED FOR TRS-80 OWNERS

Volume 3, Issue 6

Newsletter Index In This Issue . . .



RADIO SHACK ANNOUNCES AGREEMENT WITH PHILADELPHIA CITY SCHOOLS FOR MICROCOMPUTER READING PROGRAM

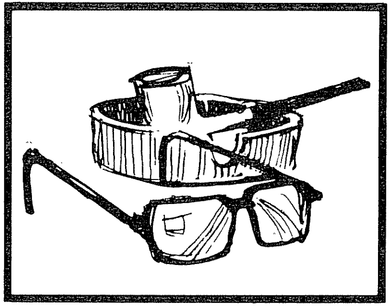
Radio Shack recently announced an agreement with the Philadelphia City School District to convert two minicomputer-based reading programs developed by the Philadelphia Schools to the TRS-80 microcomputer. Radio Shack will be licensed to distribute the reading programs, known as Computer Assisted Reading Development (CARD), and Systems Approach to Basic Reading Education (SABRE).

Computer Assisted Instruction (CAI) is an ideal medium for instruction in reading skills. The Philadelphia reading programs are designed to provide individualized, self-paced instruction in reading to students and are based on the use of established CAI techniques for using computers to supplement and reinforce regular classroom instruction.

These programs are designed to encourage positive attitudes toward learning, language and reading in students. The curriculum material is relevant and provides non-threatening, positive reinforcement in private sessions at the computer.

William Gattis, Director of Radio Shack's Education Division, said that the agreement will allow Radio Shack to offer a high quality reading program to schools for classroom use on the popular TRS-80. He added that Radio Shack's Education Division will be responsible for conversion of the reading programs to the TRS-80 from the timesharing minicomputer system they are currently implemented on. The conversion will be done using Radio Shack's new courseware authoring system, and will be the first major educational software product offered by Radio Shack to use that authoring system.

Developed originally by the Philadelphia Schools for use with selected students who were reading below grade level, the system has been in use for ten years and has had many revisions and improvements during that time. Research available as a result of the extensive use in the Philadelphia Schools documents consistent gains in student achievement in reading as a result of the reading programs. (Continued on Page 24)



View From the 7th Floor

by Jon Shirley, Vice President Computer Division

In a recent issue of a magazine which shall go un-named, there was a comparison of our BASIC compiler (RSBASIC) and the Microsoft BASIC compiler. The writer was very upset that we did not choose to sell the Microsoft compiler and implied that money might be the reason for our choice. Since our compiler is, as he pointed out, not compatible with our BASIC interpreter, I thought it might be interesting to tell just why we did choose to bring out RSBASIC. Before I go on, you color computer and pocket computer readers beware, this is about Model I, II and III only and it's so long it's continued next month. Actually due to NCC and a trip to Japan I had to write two issues at one time anyway.

You Model I, II and III owners please read on as I am talking about a language that will allow you to write a program that will run on all three machines without modification (except for screen size). First let's get down to basics (very sorry about that). The fundamental difference between the interpreter that comes with our computers and a compiler is that the interpreter changes each line of BASIC into machine language the TRS-80 understands as it gets to that line while the compiler changes the entire program to machine language all at one time. So what? Well if you have a loop that repeats 100 times the interpreter must decode the BASIC statements 100 times while the compiler does it only once. Result? Speed!

But that's just the beginning. When the compiler has finished compiling your program, which in compiler language is called source code, it produces the compiled machine code, called an object file, which is then stored. You do not need the original source code to run the program, only the object code and it is unreadable! Get the message — really protected programs. Compilers are great for business programs because they offer the speed needed by sorts and the protection of that valuable code. Most other languages like COBOL, Fortran, and PASCAL are only sold as compilers.

Since a BASIC compiler was an obviously needed program we set out to find a really good one that would offer not just the compiler advantages but, if possible, a lot more features than our interpreters AND be compatible between all our Z-80 computers. We did look at Microsoft and some others and our choice was not due to money but to a tough comparison between the various available products. RSBASIC was our eventual choice and is now available for Model I, II and III. It was written by the same group that did our COBOL and like the COBOL it derived from a mini-computer version of the language. Let's take a detailed look at what it can do for you.

One point I want to make clear is that RSBASIC is not compatible with our interpreter BASIC. That is, a program that you wrote using your TRS-80's BASIC will have to be modified to run on RSBASIC. It will also have to be modified to run on Microsoft's compiler, but there will be far fewer modifications. If your goal is to write some new software that can take advantage of features not available in our interpreter BASIC, this package is for you. If you want to convert your old software to compiler, buy the Microsoft package.

Feature difference number one is accuracy. RSBASIC works in 14 digit accuracy and does not have rounding errors. If you have ever tried to write an accounting package you can appreciate this. This 14 digit accuracy is used by ALL functions. Another feature is program size. A BASIC program cannot be directly converted to machine code without a lot of subroutines being available. In RSBASIC those are provided by a program called the run-time program that must be on the disk with your compiled program. It provides the link between your compiled code and machine language. On a Model II disk the runtime module takes up 75 sectors. We wrote a short accuracy/time testing program, that took up 7 sectors as source program. We compiled it in RSBASIC and it took up 5 sectors. But in Microsoft compiler it took 43 sectors as their compiler adds the information needed for each program onto the program instead of using a runtime module. So if you had a lot of programs on your disk you would use much more space with the Microsoft compiler.

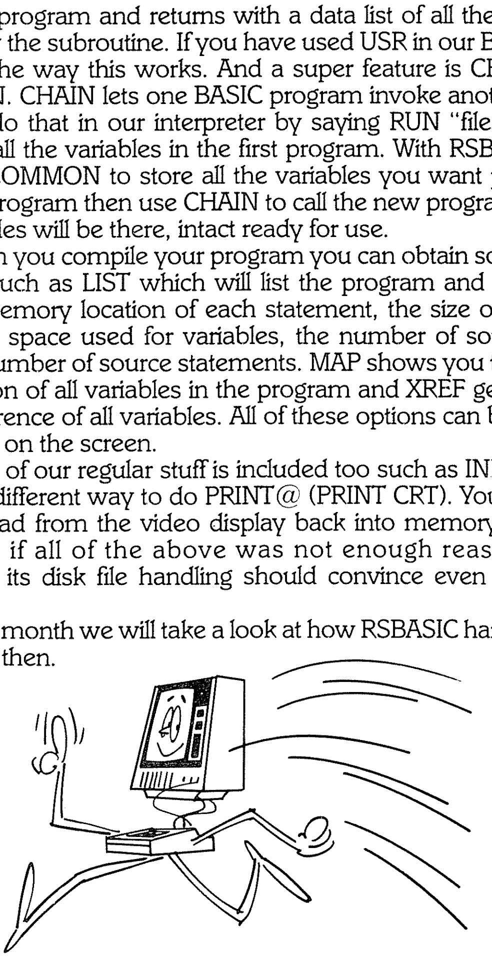
RSBASIC is also a very fast compiler when it compiles, in fact what takes a few seconds in RSBASIC takes minutes in Microsoft since you have to invoke a linker before you can get a finished program. RSBASIC also has a neat feature that allows you to run your program directly from the compiler after you have done all or part of it and want to check for errors. There are a bunch of debug tools available that let you set breakpoints to stop the program at a given line, a global replace command, a duplicate command that lets you duplicate lines from one part of the program to another part, a step command that allows you to tell the program to execute a given number of lines then stop, renumber and more.

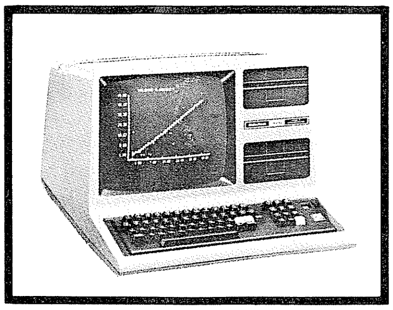
Machine language subroutines can be executed through a very simple CALL command that transfers control to a machine language program and returns with a data list of all the data calculated by the subroutine. If you have used USR in our BASIC you will love the way this works. And a super feature is CHAIN and COMMON. CHAIN lets one BASIC program invoke another. Now you can do that in our interpreter by saying RUN "filespec" but you lose all the variables in the first program. With RSBASIC you first use COMMON to store all the variables you want passed to the next program then use CHAIN to call the new program and all the variables will be there, intact ready for use.

When you compile your program you can obtain some helpful tools such as LIST which will list the program and show the relative memory location of each statement, the size of the program, the space used for variables, the number of source lines and the number of source statements. MAP shows you the memory location of all variables in the program and XREF generates a cross reference of all variables. All of these options can be printed or viewed on the screen.

Most of our regular stuff is included too such as INKEY$ and a slightly different way to do PRINT@ (PRINT CRT). You can also directly read from the video display back into memory (CRTI$). But even if all of the above was not enough reason to try RSBASIC, its disk file handling should convince even the most skeptical.

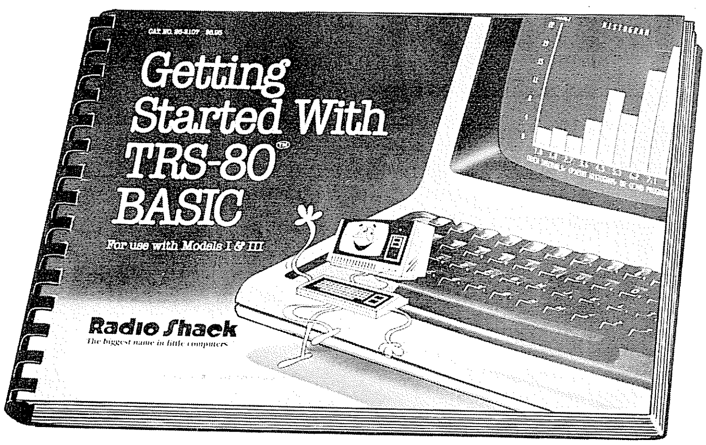
Next month we will take a look at how RSBASIC handles disk files. Until then.





Model I/III

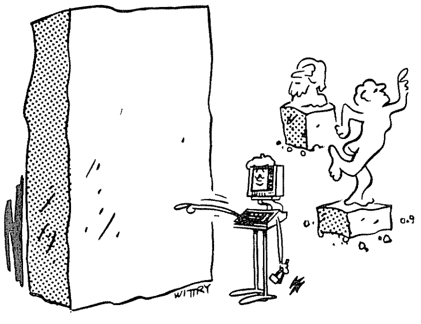
Product Line Manager's News


Getting Started With TRS-80 BASIC
For use with Models I & III
Radio Shack
The biggest name in little computers

Whether you are interested in: **BASIC Training**

or you are ready for:

Great Works of Art



you will enjoy "Getting Started with TRS-80 BASIC." Here is a short test which we took from that book:

Question: Choose the best method for inserting a new line in an existing program:

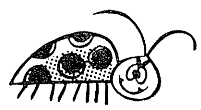
a) type in NEW and start all over.
b) hire an out-of-work programmer.
c) type in a new line number between two existing lines; then type the addition, then press ENTER.

The correct answer is c. If you got this question wrong, then this is the book you have been looking for. The book is written in three sections: Novice, Beginner, and Intermediate programmer. The writing style is light, with just enough humor to get you through the rough parts. If you have tried "textbooks" and did not like them, try this book, we think you will like it.

If you passed the quiz, you will still enjoy the book. But why should a sophisticated programmer like yourself be interested in a book whose title starts off with, of all things, "Getting Started . . ."? Well, maybe it came with your Model III and it's free. Even so, you will find that there are plenty of Useful and Factual Observations (That's UFO for the uninitiated) which make this book a real gold mine and a great reference book for programmers at all levels.

UFO

Or perhaps you have a friend or relative who hasn't gotten the programming bug yet and needs this entertaining approach to spark their interest in learning more about your TRS-80.



BUG

Whether you have a Model I or a Model III, you will find this book of interest. It is loaded with useful routines and programs like the graphics drawing board.

The graphics drawing board allows you to draw pictures and graphics on the video display using the four arrow keys for up/down or side to side movement, and the Q, W, A, and S keys for diagonal moves. Special keys are Shift-W which controls setting or resetting of graphics blocks, Shift-C which allows you to move the cursor without drawing or erasing, Shift-A which turns auto draw on or off and the Clear key which clears the screen.

```
100 ' DRAWING BOARD - REQUIRES 16K
110 '   CHANGES FOR MOD III SHOWN
        AS - REM III -
120 '*** INITIALIZATION - COLD START *************
130 CLEAR 2050                        'LOTS OF SPACE FOR DISPLAY
140 DEFINT A-Z
150 DIM V$(15)                        'ARRAY TO STORE DISPLAY
160 T= 1: F= -1: Z= 0                 'TRUE/FALSE/ZERO
170 VM= 15360                         'START OF VIDEO MEMORY
180 REM III POKE 16409,0              'SET MIII ULC MODE
190 ' KEY - DEFINITIONS
200 UP$= CHR$(91)                     'UP ARROW
210 DN$= CHR$(10)                     'DOWN ARROW
220 LF$= CHR$(8)                      'LEFT ARROW
230 RT$= CHR$(9)                      'RIGHT ARROW
240 NE$= "Q" :REM III NE$="q" NORTH-EAST
250 NW$= "W" :REM III NE$="w" NORTH-WEST
260 SE$= "A" :REM III SE$="a" SOUTH-EAST
270 SW$= "S" :REM III SW$="s" SOUTH-WEST
280 WR$= CHR$(ASC("W")+32)            :REM III WR$= "W" SHIFT - W
290 AD$= CHR$(ASC("A")+32)            :REM III AD$= "A" SHIFT - A
300 CP$= CHR$(ASC("C")+32)            :REM III CP$= "C" SHIFT -C
310 CL$= CHR$(31)                      'CLEAR
320 ST$= CHR$(ASC("S")+32)            :REM III ST$= "S" SHIFT - S
330 LT$= CHR$(ASC("L")+32)            :REM III LT$= "L" SHIFT - L
340 PR$= CHR$(ASC("P")+32)            :REM III PR$= "P" SHIFT - P
350 RD$= CHR$(ASC("R")+32)            :REM III RD$= "R" SHIFT - R
360 XH= 127: XL= 0                     'LIMITS OF X-COORDINATE
370 YH=  47: YL= 3                     'LIMITS OF Y-COORDINATE
380 ' READOUT FORMAT
390 FM$="WRITEMODE =%  % CURSORMODE = %           %"
    + CHR$(30)
395 WN$="ARRAY EMPTY - USE <SHIFT-S> FIRST"
397 QU$= CHR$(34)                      'DOUBLE QUOTE FOR TAPE OUT
400 D= 1                               'INCREMENT
410 CLS
420 '
500 '*** INITIALIZATION - WARM START **********
510 PX= (XH - XL) / 2 + 1             'START OVER AT CENTER
520 PY= (YH - YL) / 2 + 1
530 SB= POINT(PX,PY)                  'SAVE BLOCK STATUS
540 W= T                              'WRITE = ON
550 C= F                              'CURSOR-MOTION-ONLY = OFF
560 A= F                              'AUTO-DRAW = OFF
570 DX= 0: DY= 0                      'ZERO X AND Y VECTORS
580 '
```

(Continued on Page 4)

Model I/III

(From Page 3)

```
1000'*** START MAIN PROGRAM **********
1010K$=INKEY$                     'FLUSH PRIOR ENTRIES
1020GOSUB 7000                    'DISPLAY SWITCH STATUS
1030GOSUB 6010                    'BLINK BLOCK
1040'KEYBOARD ROUTINE
1050K$= INKEY$: IF K$="" THEN 1240 'IF NONE-CHK AUTO
1060IF K$=UP$ THEN DX= Z: DY=-D: GOTO 1270 'DRAW
1070IF K$=DN$ THEN DX= Z: DY= D: GOTO 1270
1080IF K$=LF$ THEN DX=-D: DY= Z: GOTO 1270
1090IF K$=RT$ THEN DX= D: DY= Z: GOTO 1270
1100IF K$=NE$ THEN DX=-D: DY=-D: GOTO 1270
1110IF  K$=NW$  THEN  DX=  D:  DY=-D: GOTO 1270
1120IF  K$=SE$  THEN  DX=-D:  DY=  D: GOTO 1270
1130IF  K$=SW$  THEN  DX=  D:  DY=  D: GOTO 1270
1140IF K$=WR$ THEN W =-W: GOTO 1240 ' CHK AUTODRAW
1150IF K$=AD$ THEN A =-A: GOTO 1240
1160IF K$=CP$ THEN C =-C: GOTO 1240
1170IF K$=RD$ THEN GOSUB 2010     'READ DSPLY INTO ARRAY
1180IF K$=PR$ THEN 5000           'PRINT ARRAY TO DSPLY
1190IF K$=ST$ THEN GOSUB 2010: GOTO 3010 'SAVE TO TAPE
1200IF K$=LT$ THEN 4010           'LOAD DSPLY FROM TAPE
1210IF K$=CL$ THEN CLS: GOTO 500 'CLR SCREEN & RESTART
1220GOTO 1030                     'UNDEFINED KEY-GET ANOTHER
1230' CHECK AUTO-DRAW
1240GOSUB 7000                    'UPDATE SWITCH READOUT
1250IF A=F THEN 1030              'NO AUTODRAW GET NEW K$
1260' DRAW ROUTINE
1270IF C=T THEN 1310              'RESTORE BLOCK,MOVE CURSOR
1280IF W=T THEN SET(PX,PY): GOTO 1340 'SET & MOVE
1290RESET(PX,PY): GOTO 1340 'ERASE BLOCK & MOVE CURSOR
1300' RESTORE BLOCK
1310IF SB THEN SET(PX,PY): GOTO 1340 'MOVE CSR
1320RESET(PX,PY)
1330' MOVE CURSOR
1340PX= PX + DX: PY= PY + DY 'INCREMENT X AND Y
1350IF PX < XL THEN PX=XH: GOTO 1370 'CHECK PX & PY
1360IF PX > XH THEN PX=XL
1370IF PY < YL THEN PY=YH: GOTO 1390
1380IF PY > YH THEN PY=YL
1390SB= POINT(PX,PY)              'SAVE BLOCK-STATUS
1400SET(PX,PY)                    'SET NEW POSITION
1410GOTO 1030                     'GET NEW K$
1420'
2000'*** SAVE DISPLAY INTO ARRAY
2010' SEE "GETTING STARTED WITH TRS-80 BASIC"
2090RETURN
2100'
2200'EMPTY THE ARRAY
2210' SEE "GETTING STARTED WITH TRS-80 BASIC"
3000'*** SAVE ARRAY TO TAPE **********
3010' SEE "GETTING STARTED WITH TRS-80 BASIC"
3050GOTO 1010
3060'
4000'*** READ DISPLAY FROM TAPE **********
4010' SEE "GETTING STARTED WITH TRS-80 BASIC"
5000'*** WRITE ARRAY TO DISPLAY **********
5010' SEE "GETTING STARTED WITH TRS-80 BASIC"
5070GOTO 500
5080'
6000'*** BLINK BLOCK **********
6010IF POINT(PX,PY) THEN RESET(PX,PY): RETURN
6020SET(PX,PY): RETURN
7000'*** DISPLAY SWITCH STATUS **********
7010IF W=1 THEN SM$="SET" ELSE SM$="RESET"
7020IF C=1 THEN SC$="POSITION ONLY" ELSE SC$="WRITE"
7030PRINT@0, USING FM$; SM$, SC$;
7040RETURN
9000'ERROR HANDLER
9010IF ERR/2+1 <> 5 THEN ON ERROR GOTO 0
9020PRINT@0, WN$; CHR$(30);
9030FOR DL=1 TO 500: NEXT DL
9040RESUME 1010
```

which will work with Model I or Model III. See the "Getting Started with TRS-80 BASIC" manual, page 329 for instructions and details on the I/O routines we left out. (The program will work without these routines, you just won't be able to save your creations on tape.)

You might also enjoy the examples of the uses for the INKEY$ function like this technique for varying the speed of a moving object:

```
1100DEFINT A-Z
1110CLS: PRINT @ 26, "MOTION"
1120PRINTTAB(12);"PRESS '->' TO SPEED UP, '<-' TO SLOW DOWN"
1130T=40               'INITIAL DELAY VALUE
1140I=10               'INCREMENT FOR CHANGING SPEED
1150FST$=CHR$(9)       'SPEED UP CHARACTER
1160SLO$=CHR$(8)       'SLOW DOWN CHARACTER
1170A$=CHR$(179) + CHR$(140)        'MOVING OBJECT
1180B$=STRING$(2,8) + STRING$(8,25) 'ERASE AND ADVANCE
1190PRINT @ 8*64,"";   'INITIAL POSITION
1200PRINT A$;          'DISPLAY OBJECT AT CURRENT POSIT.
1210K$=INKEY$          'CHECK KEYLATCH
1220IF K$="" THEN 1270 'IF EMPTY - LEAVE SPEED AS-IS
1230IF K$<>FST$ THEN 1260 'WAS -> PRESSED?
1240T=T-I: IF T<1 THEN T=0: GOTO 1280 'YES: SPEED UP
1250GOTO 1270
1260IF K$=SLO$ THEN T=T+I 'IF <- PRESSED, SLOW DOWN
1270FOR D=1 TO T: NEXT D 'DELAY (SPEED CONTROL)
1280PRINT B$; 'ERASE AND ADVANCE CURSOR POSITION
1290GOTO 1200
```

Or learn how to get repeating keys on your Model I:

```
400 CLS
410 A$=INKEY$: IF A$<" " THEN 410
420 PRINT A$;
430 FOR I=16437 TO 16444: POKE I,0: NEXT I
440 GOTO 410
```

Try converting the Alpha keys to graphics characters:

```
1355 IF ASC(A$)>96 THEN 1410 'A$ IS A SHIFTED A-Z
1410 A$=CHR$(ASC(A$)+33)  'MOVE CODE TO GRAPHICS RANGE
1420 PRINT A$: GO TO 1340
```

Or a rolling dice animation:

```
100 ' DICE WITH GRAPHICS
110 '
120 ' CLEAR SCREEN
130 CLS
140 '
150 ' RESERVE SPACE FOR STRING CHARACTERS
160 CLEAR 1000
170 '
180 ' GRAPHICS DATA
190 ' OUTLINE OF THE DIE
200 DATA 160, 176, 176, 176, 144, 26, 24, 149, 24, 24, 24, 24, 24,
         170, 26, 24, 170, 25, 25, 25, 149
210 DATA 26, 24, 149, 24, 24, 24, 24, 24, 170, 26, 24, 130, 131,
         131, 131, 129
220 DATA -1
230 '
240 ' BLANK INSIDE OF DIE ('-1' MEANS END)
250 DATA 32, 32, 32, 26, 8, 8, 8, 26, 32, 32, 32, -1
260 '
270 ' FACES OF THE DIE ('-1' MEANS END)
280 ' ONE
290 DATA 26, 25, 42, -1
300 ' TWO
310 DATA 25, 25, 42, 26, 26, 24, 24, 24, 42, -1
320 ' THREE
330 DATA 26, 42, 42, 42, -1
340 ' FOUR
350 DATA 42, 25, 42, 26, 24, 24, 24, 26, 42, 25, 42, -1
360 ' FIVE
370 DATA 42, 25, 42, 26, 24, 24, 42, 26, 42, 24, 24, 24, 42, -1
380 ' SIX
390 DATA 42, 42, 42, 26, 26, 24, 24, 24, 42, 42, 42, -1
400 DEFINT A-Z  'INTEGER ALL VARIABLES
410 DIM DF$(6)  'DF$() WILL HOLD THE SIX DIE FACES
420 '
430 ' READ OUTLINE INTO OT$
440 READ IC: IF IC=-1 THEN 480
450 OT$=OT$ + CHR$(IC): GOTO 440
460 '
470 ' READ BLANKING FIELD INTO BL$
480 READ IC: IF IC=-1 THEN 520
490 BL$=BL$ + CHR$(IC): GOTO 480
500 '
510 ' READ DIE FACES INTO DF$()
520 FOR I = 1 TO 6
530  READ IC: IF IC=-1 THEN 550
540  DF$(I)=DF$(I) + CHR$(IC): GOTO 530
550 NEXT I
560 '
570 ' INITIALIZE DIE POSITIONS
580 F1=404: D1=F1+65: F2=424: D2=F2+65
590 '
```

(Continued on Page 10)

Model I/III Bugs, Errors, and Fixes

Model III TRSDOS 26-312

In version 1.2 of Model III TRSDOS, the DUMP command requires that any hexadecimal addresses which begin with a letter (A - F) be preceded by a zero (0). For example, the command:

```
DUMP PROG2 (START=9E00,END=AF00,TRA=BE00,RELO=BDF0)
```

will cause an Illegal Parameter Error to occur. To correct this, change the command to read:

```
DUMP PROG2 (START=9E00,END=0AF00,TRA=0BE00,RELO=0BDF0)
```

Notice the addition of the numeral zero in front of each of the hexadecimal addresses which begins with a letter. The zero is not needed in front of a hexadecimal number whose first digit is a numeral from zero to nine.

Inventory Control 26-1553

If you suspect that the index in your Model I version 1.1 ICS program has "crashed" (you are receiving ERROR 64's or other indications), the following procedure may help IF the problem is only in the index itself:

1. Place both program and data disks in the computer.
2. At `DOS Ready` type in `BASIC` **ENTER** and answer both of the power-up questions `Memory Size?` and `How many files?` with **ENTER**.
3. Type in `LOAD "DATASORT"` **ENTER**
4. Change line 80 to read:
   ```
   80 FOR X=1TON:V(X)=X:NEXT:CLOSE1
   ```
5. Type in `SAVE "DATASORT"` **ENTER**
6. Type in `RUN "ICS"` **ENTER**
7. At the ICS main menu, choose the first option to add an item. Now add a fictitious item (for example, put everything in as zeros).
8. Now go back to the main menu. At this point the computer will be forced to resort your data.
9. At the main menu, after the sort has been completed, choose the last option to exit the program.
10. Type in `LOAD "DATASORT"` **ENTER**
11. Change line 80 back to its original form:
    ```
    80 FOR X=1TON:INPUT#1,V(X):NEXT:CLOSE1
    ```
12. Type in `SAVE"DATASORT"` **ENTER**
13. Now type `RUN"ICS"` **ENTER** and delete the fictitious item that was added to force the sort.
14. This completes the changes, and everything should work correctly now.

Accounts Payable 26-1554

There are some discrepancies in Model I/III Version 2.0 Accounts Payable between the report numbers at the top of the page, and the number gotten if you run the same report without leaving the Reports Menu. You may correct this problem by using the following procedure:

1. Make a BACKUP copy of your Accounts Payable diskette which contains the REPORTS program. Make the following changes on the BACKUP copy.
2. In BASIC type `LOAD"REPORTS"` **ENTER**
3. Change the beginning of lines 91, 113 and 137 to include the underlined material. The remainder of each of these lines does not change, as indicated by the ellipsis (...).
   ```
   91 R=R2:A$="VENDOR LISTING":GOSUB37:...
   113 R=R4:A$="CASH REQUIREMENT":GOSUB73...
   137 R=R3:A$="AGED ACCOUNTS PAYABLE": GOSUB37:...
   ```
4. In line 155, change R3 to R. The first portion of line 155 should now read:
   ```
   155 LPRINTTAB(TA)RA$" "A$" - REPORT #"R:LPRINT...
   ```
5. Save this correct program by typing `SAVE"REPORTS"` **ENTER**
6. Return to TRSDOS and make a BACKUP of this disk. Be sure that you use this corrected version from now on.

There is an error in the Model I version 1.2 Accounts Payable manual.

The manual currently states that while selecting invoices, if the first character of your invoice number is 'A,' and at invoice selection you type only "A," the computer will automatically select all invoices beginning with A.

The last sentence in the NOTE on page 21 should be changed to read:

If you typed 'A' for invoice number, the computer would automatically select the first invoice number beginning with 'A.'

Disk Payroll 26-1556

Model I/III Disk Payroll version 2.0 has a rounding problem when negative numbers are rounded. This problem can be corrected by the following procedures:

1. BACKUP the diskette containing the PR4INPUT program. Make the following change to the BACKUP copy.
2. In BASIC, type: `LOAD"PR4INPUT"` **ENTER**
3. Change line 6920 to read:
   ```
   6920 E#(I)=FIX(E#(I)*100#+SGN(E#(I))*.5D0)/100#: RETURN
   ```
4. To save the change, type `SAVE"PR4INPUT"` **ENTER**
5. Return to TRSDOS and make a BACKUP of your corrected diskette.

Profile 26-1562

Problem: While running 3.0 PROFILE (on Model III only) it is possible for the system to hang up during the SORT if an extreme disk error occurs caused by faulty media, hardware or the like. When this occurs you will have to RESET the system. Data may be lost, and in any case the diskettes on which the problem occurs should not be used. To continue it will be necessary to work with your Backups (after applying the patch given below).

Solution: To keep the problem from occurring apply the following patch to the 3.0 Model III Profile program:

```
PATCH SORT (ADD=781A,FIND=F5,CHG=C9)
```

Note: The above patch should be made to all copies of Model III 3.0 PROFILE even if the problem has not occurred. This patch will be made to the next version released.

Surveying 26-1577

Errors will occur in the Model I/III version 1.0 Surveying program if bearings are used in the Traverse Closure function rather than angles. One of the errors which has been experienced is an `FC ERROR IN 12040`.

To correct the problem, make the following changes to the Traverse Tape:

1. CLOAD the Traverse Tape.
2. Change lines 5600, 5610, 5630, and 5670 to read:
   ```
   5600 PRINT F$"◆DO YOU WISH TO CHANGE A BEARING? ";:
        GOSUB8:IFWFTHEN120ELSEIFIN$="N"THEN PRINT:GOTO
        4820 ELSEIFIN$<>"Y"THEN PRINTCHR$(27);:GOTO 5600
   5610 PRINT@64,F$:C%=64:FORK=0TOU-1
   5630 C%=C%+64:Q%=T%(K):X=A8(K):GOSUB 6000:
        GOSUB13000:PRINTTAB(19)"BEARING"K+1"IS"
        TAB(35)A$:NEXT
   ```

(Continued on Page 10)

BASIC Line Input

Jeffrey Carlson Fort Dodge, Iowa

Here is a subroutine to imitate Disk BASIC's LINE INPUT statement in Level II BASIC:

```
10 'LINE INPUT IMITATOR BY REV. JEFFERY CARLSON - NO RIGHTS
   RESERVED
20 CLS: CLEAR 1000
30 'REPLACEMENT FOR INPUT "COMMENTS"; IN$
40 PRINT"INFORMATION PLEASE ? ";: GOSUB 100: CLS
50 PRINT"THE INFORMATION IS : ": PRINT IN$: END
100 ' SUBROUTINE 13=ENTER 14=CURSOR ON 15=CURSOR OFF
110 PRINT CHR$(14);
120 A$=INKEY$: IF A$="" THEN 120 ELSE IF A$=CHR$(13) THEN 130 ELSE
    PRINT A$;: B$=B$+A$: GOTO 110
130 IN$=B$: B$="": PRINT CHR$(15): RETURN
```

Calorie Computer

Michael Mayer-Kielmann Fort Leavenworth, Kansas

I have enclosed with this letter a program using an algorithm which returns the number of calories used by a person during a run. The results are adjusted for the body weight of the runner as well as the pace at which he/she runs.

I am using this algorithm in a personal program which calculates my weekly running statistics for me. I have found it very useful as it converts tabular information into an equation. Therefore, rather than cascading through a series of IF-THEN statements, only one equation need be executed. This saves quite a bit of execution time.

Unfortunately, the relationship between calorie use and running is not exactly linear. However, the formula used in line 200 of the program yields results within only one calorie of the table I used. In addition, using the formula saves interpolation when desired values fall between table entries.

Since runners are often nuts about keeping statistics of their running, I thought I would share this with others who are both runners and TRS-80 owners.

PROGRAM LISTING:

```
100 REM CALORIES USED RUNNING
110 REM BY M. MAYER-KIELMANN
120 REM JANUARY 1981
130 CLS
140 INPUT"CURRENT WEIGHT ";W
150 INPUT"DISTANCE RUN ";D
160 PRINT"ENTER MINUTES AND SECONDS USING A PERIOD IN-"
170 PRINT"STEAD OF A COLON (E.G. FOR 16:30 ENTER 16.30)"
180 INPUT"TIME ";T
190 T1=INT(T)+(T-INT(T))/.6
200 C=(W*(.735993-(T1/D*.01325))+3.6)*D
210 PRINT"CALORIES USED: ";INT(C+.5)
220 END
```

COMMENTS:

The program will give accurate results to +/− 1 calorie over a range of 120-220 pounds of weight. The range of the pace is 5:20 to 10:40 minutes/mile.

Line 190 converts seconds to a true decimal value.

Line 200 implements this formula:

Calories = Weight x (Value) x Distance

where:

Value = (Y-intercept − (Pace x Slope) + correction)

The Y-intercept and slope were obtained by performing a least squares regression analysis of a standard table listing calories used per one mile run at various paces. The correction is applied to compensate for error in calculation to 120 lbs. body weight. (Note that the correction is actually applied after the product of the value and the weight has been obtained.)

Line 210 prints the results of the calculations rounded to the nearest whole calorie.

The program was written in Level II, Model I, TRS-80 BASIC.

The coefficient of determination of the least squares regression line was 97.5%. The standard error of the estimate came to .004798.

Pascal Super Sketch

David Lin Pompton Plains, New Jersey

The enclosed program is a Model I TINY PASCAL version of the Super Sketch program which was published in the May, 1980 issue of the TRS-80 newsletter. It has the same features that the BASIC version had, except the following:

1. Flashing cursor is used instead of solid.
2. (CLEAR) key will clear the screen and move the cursor to the center of the screen.
3. (ENTER) key will move the cursor to the center of screen without erasing the picture.
4. (@) key combined with the arrow keys will make the cursor transparent to the picture (flashing cursor in BASIC version).
5. (SPACEBAR) will terminate the program.

Editors Note: Some additional information you may want, is that you use the arrow keys to "draw" on the video. If you press more than one arrow key, you can draw diagonal lines. To draw continuously, simply hold down the appropriate arrow key.

```
(* SUPER SKETCH TINY PASCAL VERSION*)
(* BY DAVID LIN POMPTON PLAINS NJ *)
VAR X, Y, INCX, INCY, KEY, FLAG, DELAY: INTEGER;
    SET, RESET: INTEGER;
BEGIN
        WRITE (15,28,31); FLAG:=1; X:=63; Y:=23;
        REPEAT KEY:=MEM(%3840);
         CASE KEY OF
           (* ARROW KEYS *)
            %28: BEGIN INCY:=-1;              INCX:=-1 END;
            %30: BEGIN INCY:=1;               INCX:=-1 END;
            %48: BEGIN INCY:=-1;              INCX:=1 END;
            %50: BEGIN INCY:=1;               INCX:=1 END;
            %8:  INCY:=-1;                   %10: INCY:=1;
            %20: INCX:=-1;                   %40: INCX:=1;
           (*CLEAR KEY*)
            %2:  BEGIN WRITE (15,28,31); X:=63;
                   Y:=23 END;
           (* ENTER KEY*)
            %1:  BEGIN X:=63; Y:=23 END;
           (* SPACE BAR*)
            %84: FLAG:=0
         END;
         X:=X+INCX; Y:=Y+INCY;
         IF X>127 THEN X:=127;
         IF X<0   THEN X:=0;
         IF Y>47  THEN Y:=47;
         IF Y<0   THEN Y:=0;
         INCX:=0; INCY:=0;
           (* SHIFT KEY*)
            KEY:=MEM(%3880); IF KEY=1 THEN
                 BEGIN SET:=0; RESET:=1 END ELSE
                 BEGIN SET:=1; RESET:=0 END;
           (* @ KEY *)
            KEY:=MEM(%3801); IF KEY=1 THEN
                 BEGIN IF POINT(X,Y) THEN
              BEGIN SET:=1; RESET:=0 END ELSE
              BEGIN SET:=0; RESET:=1 END END;
         PLOT (X,Y,RESET);
         DELAY:=100; REPEAT DELAY:=DELAY-1 UNTIL DELAY<0;
         PLOT (X,Y,SET);
        UNTIL FLAG=0
END.
```

Single Disk COPY program

James R. Reed   Dallas, Texas

This is a BASIC program that I use alot since I only have one disk drive with my Model I TRS-80. It works like COPY except it reads the program or data into memory first then stops so I can put another disk in and change the name if desired before writing and verifying it. I wrote the program when I had only 32K memory but since increasing to 48K I haven't needed to increase its string space as it is used mostly for copying machine language programs from one disk to another.

I would like to see the Newsletter emphasize the use of the BASIC CLEAR statement when also using DEFINT. I have seen DEFINT used before the CLEAR string space in some magazine articles. This causes the variable to go back to single precision and shows no error."

Here is my program:

```
10 CLS
20 PRINTTAB(25) "COPYFILE": PRINT
30 CLEAR 19500
40 DEFINT X,L
50 DIM B$(75), C$(75)
60 INPUT"NAME OF FILESPEC ";A$
70 PRINT
80 OPEN"R", 1, A$
90 L=LOF(1)
100 FIELD 1, 128 AS B$, 128 AS C$
110 IF L>75 THEN 330
120 IF L=0 THEN PRINT "CANNOT FIND "A$: CLOSE: KILL A$: GOTO 310
130 PRINT A$ " HAS" L "SECTORS REQUIRING" INT(L/5+1) "GRANS"
140 FOR X=1 TO L
150 GET 1, X
160 B$(X)= B$
170 C$(X)= C$
180 NEXT
190 CLOSE: PRINT
200 INPUT"PRESS ENTER IF NEW NAME IS NOT REQUIRED";A$
210 OPEN"R", 1, A$
220 FOR X=1 TO L
230 PRINT@ 512, "COPYING " X
240 LSET B$= B$(X)
250 LSET C$= C$(X)
260 PUT 1, X
270 PRINT@ 512, "VERIFYING" X
280 GET 1,X
290 IF B$<>B$(X) OR C$<> C$(X) THEN PRINT "VERIFY ERROR!!": GOTO
   360 ELSE NEXT
300 CLOSE
310 PRINT:PRINT "FINISHED"
320 PRINT: CLEAR 50: END
330 PRINT:PRINT "THE FILESPEC " A$ " IS TOO LARGE"
340 PRINT"CANNOT HAVE MORE THAN 75 SECTORS AND IT HAS"L
350 GOTO 300
360 PRINT "COPYING ABORTED."
370 CLOSE
380 KILL A$
390 GOTO 310
```

Egyptian Numerals

Bud Myers   Washburn, Maine

On seeing the Roman numeral conversion program in the January Newsletter it occurred to me that while many programs are available for such conversions, I have yet to see one to illustrate the Egyptian method of numeration.

As a junior high teacher I present this system each year as background in the development of our present number system. Therefore, it seemed a program to display the Egyptian equivalent of numbers, as well as for Roman numerals, would fill at least my own need.

TRS-80 graphics symbols were certainly not intended to represent spirals, but the principle seemed to be more important than the actual symbols (and there is some difference in the way various scholars represent the symbols anyway) so I forged ahead.

I am pleased that I have been instrumental in persuading the school department for which I work to purchase three TRS-80 machines — two Model III's and a Model I. The interest in them has been so great that I occasionally take my own Model I to handle the overflow.

I have recommended Radio Shack products to several other neighboring school districts, and will continue to do so; I feel they are an outstanding value.

The Egyptian numeral program will run on 16K Level II Model I's or 16K Model III BASIC Model III's.

```
1 '========          EGYPTIAN NUMBER CONVERSION          ========
2 '                           BY BUD MYERS
3 '//////////                 PROGRAM LIST                //////////
4 CLEAR 500
5 CLS: PRINT CHR$(23)"   EGYPTIAN NUMBERS": FOR Z=1
  TO 1500: NEXT: CLS
6 CLS: PRINT@469,"WRITTEN BY BUD MYERS": FOR Z=1 TO
 1250: NEXT: CLS
7 PRINT"   EGYPTIANS USED THESE SYMBOLS TO
 REPRESENT NUMBERS:"
8 O$=CHR$(149):
 T$=CHR$(151)+CHR$(171):
 H$=CHR$(139)+CHR$(175)
9 TH$=CHR$(186)+CHR$(181):
 TT$=CHR$(150):
 HT$= CHR$(152)+CHR$(129)
10 M$=CHR$(157)+CHR$(174):
  S$=CHR$(194)
11 PRINT O$+S$ " FOR ONE": PRINT
12 PRINT T$+S$ " FOR TEN": PRINT
13 PRINT H$+S$ " FOR HUNDRED": PRINT
14 PRINT TH$+S$ " FOR THOUSAND": PRINT
15 PRINT TT$+S$ " FOR TEN THOUSAND": PRINT
16 PRINT HT$+S$ " FOR HUNDRED THOUSAND": PRINT
17 PRINT M$+S$ " FOR MILLION": FOR Z=1 TO 5000: NEXT
18 PRINT "   NOTE: THIS PROGRAM WILL SHOW THE
    SYMBOLS IN DESCENDING"
19 PRINT "   SEQUENCE, BUT THE EGYPTIANS DID NOT
    ALWAYS DO SO."
20 FOR Z=1 TO 3500: NEXT
21 FOR S=1 TO 10
22   A$(S)=" ": B$(S)=" ": C(S)=0
23 NEXT
24 CLS: PRINT@64,"WHAT NUMBER (UP TO 9,999,999) SHALL I
    CHANGE";: INPUT A#: IF A#> 9999999 THEN 54
25 A$=STR$(A#): L=LEN(A$)-1
26 A$(1)=LEFT$(A$,1)
27 FOR N=2 TO L
28   A$(N)=MID$(A$,N,1)
29 NEXT
30 A$(N)=RIGHT$(A$,1)
31 FOR M=L+1 TO 1 STEP -1: P=P+1
32   B$(M)=A$(P)
33 NEXT
34 FOR Q=1 TO L+1
35   C(Q)=VAL(B$(Q)): NEXT
36 IF C(7)=0 THEN 38
37 FOR R=1 TO C(7): PRINT M$+S$;: NEXT: PRINT: PRINT
38 IF C(6)=0 THEN 40
39 FOR R=1 TO C(6): PRINT HT$+S$;: NEXT: PRINT: PRINT
40 IF C(5)=0 THEN 42
41 FOR R=1 TO C(5): PRINT TT$+S$;: NEXT: PRINT: PRINT
42 IF C(4)=0 THEN 44
43 FOR R=1 TO C(4): PRINT TH$+S$;: NEXT: PRINT: PRINT
44 IF C(3)=0 THEN 46
45   FOR R=1 TO C(3): PRINT H$+S$;: NEXT: PRINT: PRINT
46 IF C(2)=0 THEN 48
47   FOR R=1 TO C(2): PRINT T$+S$;: NEXT: PRINT: PRINT
48 IF C(1)=0 THEN 50
49   FOR R=1 TO C(1): PRINT O$+S$;: NEXT: PRINT: GOTO 51
50 PRINT: PRINT "THE EGYPTIANS HAD NO SYMBOL FOR ZERO."
51 PRINT: INPUT "SHALL I DO ANOTHER"; YN$
52 IF LEFT$(YN$,1)="Y" THEN P=0: GOTO 21
53 CLS: END
54 CLS: PRINT CHR$(23)
55 PRINT@128, "YOU'RE NOT PAYING ATTENTION!!":
    PRINT"NUMBERS UP TO 9,999,999 ONLY.":
   FOR Z=1 TO 1500: NEXT: GOTO 21
```

Model I Disk Fix

O. R. Fonorow   Cheyenne, Wyoming

I have finally discovered what caused the majority of my problems with my mini-disk drive so I thought I'd write and tell you in case others are experiencing similar problems.

It turns out that the only thing wrong with my disk drive is that the cable slips a fraction on the card edge (maybe from vibration, the cover, etc.). It seems tight and I may never have found it out except that a motor drive (test) speed program kept telling me that I didn't have any disk drives hooked up when I actually did.

Whenever I experience a Disk I/O error I turn the disk off, reconnect (lift slightly) the cable to the card edge—and everything works beautifully. You might want to mention this in your newsletter. If it's already been in—then you might want to mention it again since I didn't see it the first time.

Double Precision Square Root

Lee A. De Boer   Worthington, Ohio

REF: March 1981, TRS-80 Microcomputer News
Double Precision Square Root

Whenever possible, I like to use the defined functions in lieu of subroutines because they are easier to use in programs. The following program listing and print out of a double precision square root program shows how nesting defined functions can be utilized to simplify programming. However, the defined function capability is limited to those who have Disk BASIC.

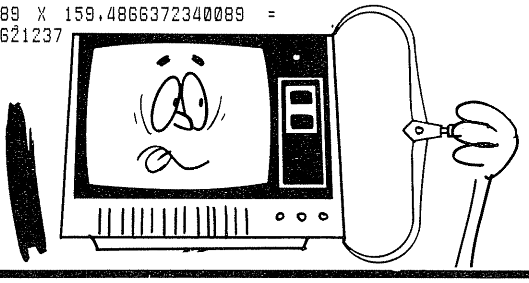
```
10 'DOUBLE PRECISION SQUARE ROOT DEFINED FUNCTION
20 ' BY LEE A. DE BOER
30 CLS
40 '
50 '1ST ORDER PRECISION
60   DEF FNS2#(S#,S2#)=(S2#+S#/S2#)/2#
70 'DOUBLE PRECISION SQUARE ROOT
80   DEF FNSQ#(S#)=FNS2#(S#,FNS2#(S#,SQR(S#)))
90 PRINT
100 INPUT"ENTER ANY POSITIVE NUMBER"; X#
110 IF X#<0 THEN 100
120 Y#=FNSQ#(X#)
130 PRINT Y# " IS THE SQUARE ROOT OF " X#
140 PRINT Y# " X " Y# " =" Y#*Y# 'CHECK ACCURACY
150 GOTO 90
160 '
170 '
180 END
```

```
ENTER ANY POSITIVE NUMBER? 144
12  IS THE SQUARE ROOT OF  144
12 X 12 = 144

ENTER ANY POSITIVE NUMBER? 68
8.246211251235321  IS THE SQUARE ROOT OF  68
8.246211251235321  X  8.246211251235321 = 68

ENTER ANY POSITIVE NUMBER? 235.878469
15.35833548923841  IS THE SQUARE ROOT OF   235.878469
15.35833548923841   X   15.35833548923841   =   235.878469

ENTER ANY POSITIVE NUMBER? 25435.98745621236545
159.4866372340089  IS THE SQUARE ROOT OF
    25435.98745621237
159.4866372340089  X  159.4866372340089  =
    25435.98745621237
```



GET YOUR DATA FOR FREE

A program to convert a machine language program for use with BASIC

By William Terrell

How many of you have avoided using the 'USR' command to call machine language programs from BASIC? Was it because it is cumbersome to change the machine code into decimal data for BASIC to READ and POKE into high memory? I know that was my excuse!

What we needed was a program to read the machine code, convert it to decimal, and create lines of DATA for our BASIC program. I've now written such a program. As background, for your understanding and use of this program, I'll describe some of the factors in the program design. Then I'll show you how to use it.

The program itself is quite simple, but I needed decisions on five points before I could start:

- DATA line numbers
- Line number increments
- Start address for machine code being read
- End address for machine code being read
- Location for this routine

Do I provide for user input of the first DATA line number or do I define the line number within the program? I opted for both. User input is provided by use of the USR command in the command mode. Since this is limited to a maximum starting line number of 32767, I also provided a default starting line number of 50010. The reason for this strange starting number will become clearer as you read on.

Do I provide for user input of the line increment or do I define it within the program? There did not seem to be a good reason for the extra complications of user input here so I defined the increment as ten (decimal) within the program. Then, if I put ten bytes of data on each line, the least significant digits of the line number serve also as a byte counter.

How was the program to know where to find the machine language program it was to read? Again, I could have provided for user input. However, it seemed as logical (and easier) to define the starting address as just above the user response to 'MEMORY SIZE?.' Therefore, if the programmer has loaded his machine code in protected high memory, this program can start reading there.

The program also needs to know where the machine code ends. As before, I could have provided for user input but simplicity again won out. Let the program read and convert machine code until it reaches the end of memory.

Finally, I needed to decide where in memory to load this program. I considered three locations: 1. BASIC's I/O buffer, 2. just below protected memory, and 3. just above reserved RAM (where the BASIC program usually starts). BASIC's I/O buffer seemed to be used for things I didn't fully understand. Therefore, it was likely that the TRS-80 would find a way to mess up my program if I put it there. Depending on length, each machine language program may require a different response to the 'MEMORY SIZE?' prompt. Thus, loading the program just below protected memory seemed to present some complications when loading the program. Also, BASIC uses this area when you 'RUN' a program. Only the third choice—just above reserved RAM—didn't seem to have any real drawbacks. The choice was obvious.

I used Radio Shack's Editor/Assembler, a very useful tool, to prepare the program (see listing). Initially the program sets three BASIC pointers. The Start of BASIC pointer (40A4H) is set to the address just beyond this program. The End of BASIC pointer (40F9H) is set two bytes beyond this address. And the USR Calling Address pointer (408EH) is set to the first byte of this program.

(Continued on Page 9)

Free Data (From Page 8)

Then, the origin of this program is defined. I used 42E9H (normally where the BASIC program storage starts in a Level II Model I 16K machine) as my program origin, but Disk or Model III users may have to make some adjustments.

The next segment of the program provides for user input by the use of the USR command. A call to a ROM routine at 0A7FH puts the user defined starting line number into the HL register pair. If we enter the program at 'START' instead, the starting line number is defined as 50010.

The program uses IX and IY as pointers. The position in the BASIC program is tracked by IY. Initially, IY is set at two more than the stored PROTECT MEMORY pointer (in 40B1H). This is one location before the start of the machine language program we're reading with our program.

For each new line of data (starting at 'DOLINE') the program performs several operations. It checks to see if it is at the end of memory. If not, it inserts a dummy line pointer so that BASIC won't become confused. Next, it inserts the line number (stored as HL) and increments it to the next number. Then it inserts 88H, which is the token for 'DATA.' Finally, the program sets itself to read ten bytes of data.

Starting at 'GETBYT,' the program gets (up to) ten bytes of data and translates each into decimal in ASCII code. It checks for the end of memory before getting each byte. The program has provision to suppress leading zeros, so that we don't clutter our DATA lines with useless zeros. After inserting the decimal digits into the BASIC line, the program inserts a comma to separate the data bytes.

The conversion of a byte of hexadecimal information to decimal (see subroutine starting at 'CONVRT') is a subtraction process using the Z-80's alternate register set. First, one hundred is subtracted from the byte until there is an overflow (or 'carry'). The overflow is restored and the number of hundreds which fit is stored (in ASCII code) as the most significant digit. The process is repeated with ten as the number being subtracted to get the middle digit. After this, the balance is the least significant digit.

Whenever a line is completed, a zero (the token for the end of a line) is inserted. Two more zeros are inserted at the end of the BASIC program as soon as IY is found to be at the end of memory.

Finally, the new end address for the end of the BASIC program is stored in the END OF BASIC pointer location. The dummy line pointers inserted earlier are corrected using a ROM routine. Then it's back to BASIC for more programming or RUN or whatever.

I've probably told you more about the program design than you wanted to know. Now let's get to the use of the program which I've named 'AUDATA' (for AUtomatic DATA).

Turn on your Model I non-disk TRS-80 and respond to `'MEMORY SIZE?'` with one less than the lowest address of your machine language program. Enter `'SYSTEM'` and load your machine language program. Respond to the `'*?'` prompt after your program has loaded with `'AUDATA.'` After AUDATA is loaded, respond to the prompt `'*?'` with a `'/'` (slash **ENTER**). You are now back in BASIC and ready to go!

You may key in your BASIC program, or CLOAD it or go ahead and execute AUDATA. When you are ready to execute AUDATA, decide whether DATA lines starting at 50010 are suitable or whether you wish to define your own starting line number for DATA lines. If 50010 is alright:

Enter `SYSTEM` **ENTER** and respond to the `'*?'` prompt with `17134` **ENTER**. When `READY` appears, your DATA lines have been created.

If you want to set your own starting line number for the DATA statements, use:

Enter `A=USR` (line number) **ENTER** in the TRS-80 command mode. You do not need to set the entry points, AUDATA has already done that for you. When `READY` appears, your DATA lines have been converted.

Remember that the line number for the USR command must be less than or equal to 32767. A choice ending with the two least significant digits equal to ten is convenient for the reason discussed earlier.

Now your BASIC program has your machine language program converted to BASIC DATA statements starting with the line number you have chosen (or 50010 as appropriate). Suppose that there were some bytes at the end of memory that you really didn't want (they were not part of your machine language program). Just delete them with DELETE or EDIT. Now you are ready (if you didn't do it earlier) to put a FOR-NEXT loop into your program to POKE the code into protected memory. Remember that addresses above 32767 must be converted in accordance with the instructions of the Level II manual.

With this new tool for the programmer eliminating the drudgery of converting machine language programs into DATA statements, I hope to see some fantastic new programs. Get to work!

Editor's Notes: We ran AUDATA in a 48K Model I with no problems. The program simply begins wherever you have set the MEMORY SIZE.

A couple cautions: First, since this program converts to the end of available memory, don't try this procedure in a 48K Model I on a fifty byte program that loads at the top of 16K. The program will try to convert the upper 32K of memory into DATA statements!

Second, while "playing around" with AUDATA, I discovered a couple of things you should be aware of. Both of these things seem to occur if you run AUDATA twice (there really should be no reason to run it twice). The first thing I noticed was that I got two sets of identical line numbers, one set following the other. The implication of this is that whatever line numbers you decide to use when you run AUDATA, they MUST be the highest line numbers in the existing program. If your existing program has a line number 1000, don't tell AUDATA to number from 200. Line 200 would follow line 1000, and that would not be good. The next thing I noticed was that the DATA values which I got on the second run were different from what I got the first time.

The first time through, I had put 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 at the bottom of reserved memory. The first time through AUDATA, I got a DATA statement with the numbers 1 through 10 as expected. The second time through, I got 254, 253, 252, ... A third time through and I again had 1 through 10. Moral: Use AUDATA once for each program to be converted. If you must use it a second time, reload the machine language program which is to be converted.

Enjoy!

Assembly Language Listing

```
;* * * * * *
; 'AUDATA' PROGRAM TO AUTOMATICALLY CONVERT
; MACHINE LANGUAGE PROGRAMS STARTING 1 LOCATION
; ABOVE MEMORY SIZE THRU END OF MEMORY TO DATA
; LINES IN BASIC
;                  W. F. TERRELL OCTOBER 1980
;* * * * * *
;          TO EXECUTE:
;                  USE /17134 AFTER LOADING OR
;                  AFTER 'SYSTEM' FOR DATA LINES
;                  STARTING AT 50010

;                  USE A=USR(LINE NUMBER) IN
;                  COMMAND MODE (LINE NUMBER >=0
;                  AND <=32767)

;*** SET POINTERS ***
           ORG     40A4H         ;FIX ST BAS PTR
           DEFW    STBAS+1
           ORG     40F9H         ;FIX END BAS PTR
           DEFW    STBAS+3
           ORG     408EH         ;USR CALLING ADDRESS
           DEFW    USRIN
```

(Continued on Page 10)

Free Data (From Page 9)

```
00310            ORG    42E9H           ;NORMAL BASIC STORAGE
00320
00330 ;*** PROVIDE FOR USER DEFINE DATA LINE NUMBERS **
00340 USRIN      CALL   0A7FH           ;GET LIN NO IN HL
00350            JR     CONTIN          ;BYPASS DEFAULT LIN NO
00360
00370 ;*** SET UP PROGRAM POINTERS ***
00380 START      LD     HL,50010        ;STARTING LINE NO
00390 CONTIN     LD     DE,10           ;LIN NO INC
00400            LD     IX,(40F9H)      ;END OF BAS+2
00410            DEC    IX              ;BACK UP TO LIN END+1
00420            DEC    IX
00430            LD     IY,(40B1H)      ;PROT MEM ADDR
00440            INC    IY              ;TO MEM SIZE
00450            INC    IY
00460
00470 ;*** SET LIN PTR, LINE NUM, AND 'DATA' ***
00480 DOLINE     LD     A,(IY+1)        ;GET NXT BYTE
00490            CPL                    ;COMPLEMENT A
00500            LD     (IY+1),A        ;STOR IT
00510            CP     (IY+1)          ;DID IT CHANGE?
00520            JR     NZ,ENDPR1       ;IF SO, END IT
00530            CPL                    ;RESTORE ORIG BYTE
00540            LD     (IY+1),A        ;STOR IT
00550            LD     (IX+0),5        ;FILL LIN PTR LO
00560            INC    IX              ;TO LIN PTR HI
00570            LD     (IX+0),5        ;FILL LIN PTR HI
00580            INC    IX              ;TO LIN NO LO
00590            LD     (IX+0),L        ;FILL LIN NO LO
00600            INC    IX              ;TO LIN NO HI
00610            LD     (IX+0),H        ;FILL LIN NO HI
00620            INC    IX              ;TO NXT LOC
00630            LD     A,88H           ;'DATA' TOKEN
00640            LD     (IX+0),A        ;STOR IT
00650            ADD    HL,DE           ;INC LIN NO
00660            LD     B,10            ;10 DATA BYT/LIN
00670
00680 ;*** GET AND DECODE 10 DATA BYTES ***
00690 GETBYT     INC    IX              ;TO NXT LOCATION
00700            INC    IY              ;TO NXT ML BYTE
00710            LD     A,(IY+0)        ;GET BYTE
00720            CPL                    ;GET COMPLEMENT
00730            LD     (IY+0),A        ;STOR IT
00740            CP     (IY+0)          ;DID IT CHG?
00750            JR     NZ,ENDPRG       ;IF SO, END IT
00760            CPL                    ;RESTORE ORIG BYT
00770            CALL   CONVRT          ;CONVERT TO DECIM
00780            PUSH   AF              ;SAVE LSD
00790            LD     A,H             ;PUT MSD INTO A
00800            CP     30H             ;IS IT 0?
00810            JR     Z,TENS          ;IF SO,BYPASS
00820            LD     (IX+0),H        ;STOR 100'S DIGIT
00830            INC    IX              ;TO NXT LOC
00840            JR     TENSIN          ;BYPASS 0 CHECK
00850 TENS       CP     L               ;BOTH 0?
00860            JR     Z,ONES          ;IF SO, BYPASS
00870 TENSIN     LD     (IX+0),L        ;STOR 10'S DIGIT
00880            INC    IX              ;TO NXT LOC
00890 ONES       POP    AF              ;GET LSD
00900            LD     (IX+0),A        ;STOR LSD
00910            INC    IX              ;TO NEXT LOC
00920            LD     A,','           ;GET COMMA
00930            LD     (IX+0),A        ;STOR IT
00940            EXX                    ;BACK TO MAIN REG SET
00950            DJNZ   GETBYT          ;DO 10 BYTES
00960
00970 ;*** SET UP END OF LINE TOKEN ***
00980            XOR    A               ;CLEAR A
00990            LD     (IX+0),A        ;END OF LIN MARKER
01000            INC    IX              ;TO NXT LOC
01010
01020 ;*** GO DO NEXT LINE OF DATA ***
01030            JR     DOLINE          ;TO NXT LINE
01040
01050 ;*** PROVIDE END OF PROGRAM TOKENS ***
01060 ENDPRG     XOR    A               ;CLEAR A (IGNORE ERR MSG)
01070            LD     (IX-1),A        ;LOAD 3 0'S END PRG
01080 ENDPR1     XOR    A               ;LOAD 2 0'S END PRG
01090            LD     (IX+0),A
01100            INC    IX
01110            LD     (IX+0),A
01120            INC    IX
01130
01140 ;*** DO HOUSEKEEPING AND RETURN TO BASIC ***
01150            LD     (40F9H),IX      ;END PROG PTR
01160            CALL   1AF8H           ;FIX LIN PTRS
01170            JP     6CCH            ;BACK TO BASIC
01180
01190 ;*** SUBROUTINE TO CONVERT BYTE TO ASCII ***
01200 CONVRT     EXX                    ;ALT REG SET
01210            LD     C,100           ;C=100
01220            LD     D,10            ;D=10
01230            LD     B,30H           ;CLEAR CTR TO ASCII 0
01240 SUBC       SUB    C               ;SUB 100
01250            INC    B               ;INC CTR
01260            JR     NC,SUBC         ;REPEAT UNTIL OVERFLOW
01270            DEC    B               ;BACK UP CTR 1
01280            ADD    A,C             ;RESTORE OVERFLOW
01290            LD     H,B             ;SAVE MSD IN H
01300            LD     B,30H           ;CLEAR CTR TO ASCII 0
01310 SUBD       SUB    D               ;SUB 10
01320            INC    B               ;INC CTR
01330            JR     NC,SUBD         ;REPEAT UNTIL OVERFLOW
01340            DEC    B               ;BACK UP CTR 1
01350            ADD    A,D             ;RESTORE OVERFLOW
01360            LD     L,B             ;SAVE MIDDLE DIG IN L
01370            ADD    A,30H           ;CONV TO ASCII
01380            RET
01390
01400 STBAS      DEFB   0
01410
01420 INIT       LD     BC,6CCH         ;RETURN TO BASIC
01430            PUSH   BC              ;PUT ON STACK
01440            JP     1B49H           ;CLEAR W/'NEW'
01450            END    INIT
```

Editor's Note: Mr. Terrell got a field overflow error during assembly in line 01060, which he indicates you should ignore. When I assembled this I did not get any errors. I left the comment in, just in case.

Model I/III Bugs, etc. (From Page 5)

Surveying (cont.)

```
5670 X=A8(K):GOSUB 6000:GOSUB13000:PRINTTAB(27)A$
     TAB(38);:FL=8:GOSUB10:IFWFTHEN4820ELSEA$=IN$:
     GOSUB12000:PRINT:IFITHEN5650ELSEGOSUB690:A6(K)=D:
     T%(K)=Q%:GOSUB700:A8(K)=X:GOTO5650
```

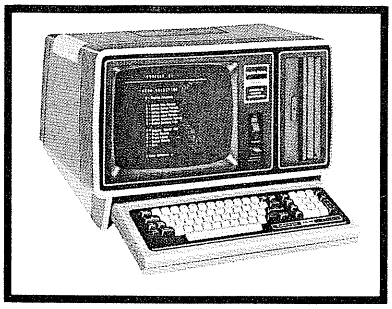
3. After you have completed and verified the corrections, type in `CSAVE"TF"` **ENTER**

Model I/III (From Page 4)

```
600 '  MAIN PROGRAM
610 '
620 '  PRINT OUTLINES
630 PRINT@F1, OT$;: PRINT@F2, OT$;
640 '
650 '  ROLL THE DICE
660 FOR I=1 TO RND(11) + 9  'ROLL 10 TO 20 TIMES
670   GOSUB 760: V1= IR ' GET 1ST RND VAL AND SAVE IT
680   PRINT@D1, BL$;: PRINT @D1, DF$(V1);
690   GOSUB 760: V2= IR ' GET 2ND RND VAL AND SAVE IT
700   PRINT@D2, BL$;: PRINT @D2, DF$(V2);
705   FOR DL=1 TO 66: NEXT
710 NEXT I
720 PRINT@788, "TOTAL IS"; V1+V2
730 INPUT"PRESS <ENTER> FOR NEXT ROLL";X: CLS
740 GOTO 630
750 END
760 '    RANDOM NUMBER GENERATOR
770 '  ON EXIT, IR= RANDOM VALUE FROM 1 TO 6
780 IR=RND(6): RETURN
```

Whether or not you are already an expert programmer, you will find that this 340 page book includes these and many other programming examples that you will be able to use in future programs.

Remember, this manual is included free with each Model III BASIC or Disk BASIC Model III (either the book is in the box, or we included a card for you to mail. If you did not get the book, be sure you mailed us the card so we can send you a copy of this great book!). If you have a Level II Model I, or you are considering an upgrade to Level II or Model III BASIC, this book should be of interest to you. Ask your Radio Shack dealer for a copy of 26-2107 (Suggested retail price $6.95).



Model II

Product Line Manager's News

Profile II Users (Again)

I have received many letters from customers and from our Computer Centers asking for minor changes or enhancements to Profile II.

Obviously, many of the changes can be made with patches, but most of the enhancements would require massive changes to the original source code and a complete re-assembly of several or most of the modules of Profile II.

The major enhancements have become a prime target for a future version of Profile, but I will try to keep you up to date on the changes that can be implemented in this version with patches.

One of the more requested changes is the ability to change the sort from an "ascending" sort to a "descending" sort when printing a report. Ascending sorts are nice for alphabetizing, but not too good if you want to rank numbers from biggest to littlest.

To make sorting in either direction possible, you need to create two DO files as follows:

```
BUILD ASCEND
PATCH PRINT/EFC A=31E1 F=F2FB31 C=FAFB31
```

and

```
BUILD DESCEND
PATCH PRINT/EFC A=31E1 F=FAFB31 C=F2FB31
```

From TRSDOS, typing `DO DESCEND` will patch the print module for a "descending" sort. Typing `DO ASCEND` will return the print module to the original configuration. If you try to do the same patch twice, you will get a "String Not Found" error message. This message just tells you that the Print module is already patched for the sort you want. You can continue as usual.

The next patch is for use with the DAISY WHEEL printer (or any other single sheet printer). It will allow you to print a single page at a time. After each page, the printer will STOP to allow you to load another sheet of paper into the printer.

Installing a "MESSAGE" flag at the bottom of the screen would have been a major change, so there is no warning or indicator flashing at you. The printer just pages up, everything stops and waits for you to change paper and press any key.

There are three patches to make this happen. The first patch only needs to be made one time. It can be installed and left so there is no need for it to be in a DO file. Just make this patch:

```
PATCH PRINT/EFC A=5E48 F=00000000000000
                       C=3E04CFC2485EC9
```

The next two patches should be in DO files with the file names SINGLE and CFORMS. These DO files are created using:

```
BUILD SINGLE
PATCH PRINT/EFC A=5A1F F=C25F5A C=CD485E
```

and

```
BUILD CFORMS
PATCH PRINT/EFC A=5A1F F=CD485E C=C25F5A
```

From TRSDOS, typing `DO SINGLE` will install the patch for single sheet printing, and `DO CFORMS` will re-patch the print module back to its original form.

By the way, on all of these patches, you might want to include the command "M" as the last line of the DO file. If you do this, you will be taken to the Profile II menu as soon as the patch is complete.

If you are using the BASIC MENU from the March newsletter, make the last command line `BASIC MENU` instead of M. You could also make the DO files (especially the UP-DOWN sort options) some of your Menu select options. For example, `SYSTEM"DO ASCEND"` or `SYSTEM"DO SINGLE."`

Would you like to create segments 2, 3, or 4 with less than 256 bytes in each record? If you need, for example, more information than the 85 bytes of segment one, but you don't really need the complete 256 bytes available for segment two, install the following patch:

```
PATCH CREATE/EFC A=51C7 F=C3ED51 C=000000
```

This patch is on one of the "creation" modules of Profile II, so it will not have any effect on your existing Profile II data bases. You will see its effect the next time you start to create a new data base. After you have created segment 1 and start to create segment 2, you will be asked to input the file size (1-256 bytes). If you just press ENTER, you will get the default 256 bytes in each record of the new segment.

It is very difficult to document what the maximum number of records will be if you use this last patch. This opens the door to a lot of combinations. You will need to watch your FREE SPACE on the disk (At the bottom of each directory) very closely, or make some BACKUPs before you go searching for the upper limit to the number of records you can have, or you may find out the hard way!

If you are currently using multi-drives and only have a 2 segment file, you can now make use of your third drive.

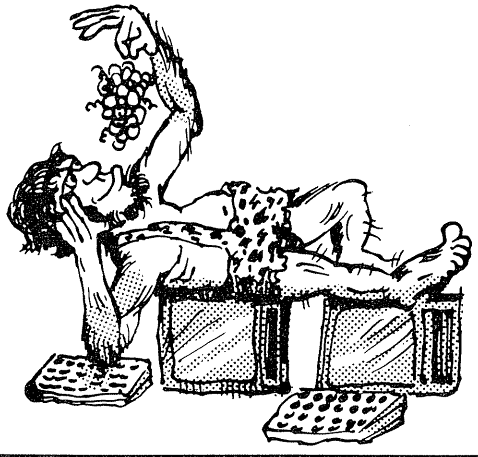
The maximum number of records, even for a 2 segment file, has been limited to 1800 records on mutli-drive systems. The limiting factor has been that 1800 256-byte records will fill a disk, and you cannot span one segment over two drives. Split the 256 byte segment 2 into two segments of 128 bytes each, and place one segment on drive one, and the other on drive two. This should allow you over 3000 records in the file.

Again, watch the amount of free space on all disks, especially drive 0, as you expand the file. Drive 0 will fill up fast. Although it only handles one 85 byte record for each of the 128 byte records on drives one and two, it also contains the TRSDOS operating system and all of the Profile II program modules.

So much for Profile II for this month. It seems that every month, when it comes time to work on my input for the newsletter, Profile II or Scripsit usually end up being the subjects. Next month I will try to put both of these on ice and cover a few of the other Model II products.

Meanwhile, I do like to have input from the field on enhancements to current products, but please, let me have them in writing, and not over the phone.

Until next month.



Model II Bugs, Errors and Fixes

Model II TRSDOS Terminal Utility

Here is some information, and three patches which you may find useful if you choose to access CompuServe™ or Dow Jones using the TRSDOS 2.0 Terminal Utility rather than the Model II Videotex Software (26-2221):

1. The Model II Terminal Utility will report parity errors by displaying a reverse video "P." When you are using CompuServe or Dow Jones, you will encounter a large number of false parity errors. To eliminate this display, make the following patch:
   ```
   PATCH TERMINAL A=34C2 F=200C C=0000 (ENTER)
   ```
2. The TRSDOS 2.0 Terminal Utility has a "video filter" option which screens out video control characters which may have undesirable effects on the displayed material. Many ASCII teletypes use special control codes which instruct the teletypes to perform certain functions which are unnecessary on a video display. When the video filter screens out a character, a "±" will be displayed instead. The purpose of this is to inform the user that some undisplayable code has been screened out. If you find it more desirable to have NO character displayed when a character is screened out, use the following patch:
   ```
   PATCH TERMINAL A=36A7 F=7F C=00 (ENTER)
   ```
3. To determine which characters are to be screened out, the Terminal Utility compares the current character to be displayed with a list of characters to be screened out. If the character to be displayed is in the list, it is screened out. If the character is not in the list, it is displayed as is. Here is a reproduction of the list:
   ```
   377B: 01, 02, 03, 04, 05, 06, 07, 0B
   3783: 0C, 0E, 0F, 10, 11, 12, 13, 14
   378B: 15, 16, FF, FF, 1E, 1F, FF, FF
   3793: FF, FF, FF
   ```
   Any value other than FFH indicates that the matching character is to be screened out. A value of FFH is a place holder for future values to be added.

   If, for instance, you do not want the screen to be cleared if an escape character (1BH) is received, then you need to add the escape character (1BH) to the list of screened out characters. To do this, find an unused position in the screen out list (a position with FFH in it), and patch this location to put the 1BH in place of the FFH. Notice that in the above list, the third position in the third row has an FFH. The numbers along the left side of the list indicate the Hexadecimal address where the first item in that row is. If the first item in the third row is in location 378B, then the third item is in what position? Remember to count in Hexadecimal! First position, 378B, Second position, 378C, third position 378D. Got it? The patch to screen out the 1BH character would look like this:
   ```
   PATCH TERMINAL A=378D F=FF C=1B (ENTER)
   ```
   You will probably want to use all three of these patches before you go on line with CompuServe or Dow Jones.

General Ledger 26-4501

In version 1.2 of General Ledger, the batch total may not equal the Document total after you print the document, or during posting. To correct these problems, follow these steps:

1. BACKUP the diskette(s) which contains the "Txentry" and "Txpost" programs. Make your correction on the BACKUP copy.
2. To correct the print error, in BASIC type:
   ```
   LOAD"Txentry" (ENTER)
   ```
3. Change lines 1450, 6730, and 6740 to read as shown below. Note: you may already have made one change to line 6730. That change was listed in the March, 1981 Newsletter, and was not correct.
   ```
   1450 IFHI<LOTHEN1405ELSEFORI=0TONA-1:S#(I)=Z#:
        NEXT:BT#=Z#
   6730 IN#=0:FORQ=0TONA-1:IF S#(Q)>0THEN
        IN#=IN#+S#(Q)
   6740 NEXT:LPRINTTAB(17)"BATCH TOTAL BY ACCOUNT"
        TAB(42);:GOSUB5100:LPRINT"":L=L+1
   ```
4. Add the following lines to the program:
   ```
   6345 IFIN#>0THENBT#=BT#+IN#
   6745 IN#=BT#:LPRINTTAB(17)"BATCH TOTAL BY ENTRY"
        TAB(42);:GOSUB5100:LPRINT" ":L=L+1:RETURN
   ```
5. To save these changes, type `SAVE"Txentry"` (ENTER)
6. To correct the problem in the posting program, in BASIC type:
   ```
   LOAD"Txpost" (ENTER)
   ```
7. Change lines 420, 730, and 740 to read as follows. Note, you may have made a correction to line 730 which was printed in the March, 1981 Newsletter. That correction was in error, and the line should be changed to read as shown here:
   ```
   420 FORJ=0TONA-1:S#(J)=Z#:NEXT:BT#=Z#:RETURN
   730 IN#=0:FORQ=0TONA-1:IF S#(Q)>0THEN
       IN#=IN#+S#(Q)
   740 NEXT:LPRINTTAB(17)"BATCH TOTAL BY ACCOUNT"
       TAB(42);:GOSUB5100:LPRINT"":L=L+1
   ```
8. Add the following lines:
   ```
   745 IN#=BT#:LPRINTTAB(17)"BATCH TOTAL BY ENTRY"
       TAB(42);:GOSUB5100:LPRINT" ":L=L+1:RETURN
   3345 IFIN#>0THENBT#=BT#+IN#
   ```
9. To save these changes, Type in:
   ```
   SAVE"Txpost" (ENTER)
   ```
10. Return to TRSDOS and make BACKUP copies of the diskette(s) which you just corrected.

VisiCalc™ 26-4511

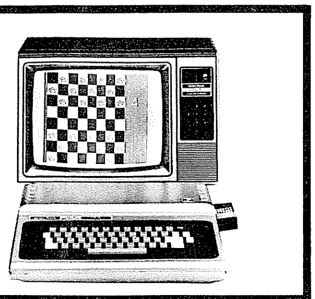
There have been some problems with VisiCalc printing reports in which a set of extra linefeeds may occur. This problem has been traced to the user setting the Printer to Top of Form manually. In order to set Top of Form on a printer from VisiCalc, use the / S E T (ENTER) (ENTER) sequence. The / S E lets you execute a TRSDOS command from VisiCalc. The T is the TRSDOS Top of Form command. The first (ENTER) executes the Top of Form, and the second (ENTER) returns you to VisiCalc. As soon as you have completed this sequence, physically move the paper so that it is aligned properly in your printer.

Profile II 26-4512

It is possible, when you initially set up your Profile II, that you will create more segments than you actually need. If this happens, the following procedure will allow you to reduce the number of segments:

1. Make a BACKUP of your Profile II master diskette, and the diskette with too many segments. For clarity, we will call the BACKUP of the master Profile disk "A" and the BACKUP of the disk with too many segments "B."
2. Use the "A" diskette (BACKUP copy of the Master diskette) to create a new file with the data segments defined exactly as you want them.
3. After you have created the file on A, expand it to the same size as the original file on B.
4. Return to TRSDOS. Now put the B diskette (BACKUP of the diskette with too many segments) in the computer, execute a

(Continued on Page 21)



Color Computer

Product Line Manager's News

With June upon us, I hope that you have gotten your first suntan for the summer season out of the way and are back at the keyboard of your Color Computer.

There seems to be a certain amount of feedback concerning the size of RAM available in the 4K and 16K Color machines. Let me try to "clear the air," so to speak, about RAM in the various machines with color.

Before we start, let's set some ground rules and standard procedures through which our Color machines operate. The computer uses "addresses" or "memory locations" for keeping track of the various functions necessary for it to operate correctly. By changing the value of only one address, certain functions of the computer will no longer operate. Some of these functions are in ROM and cannot be changed by the programmer, while others are stored in RAM and can be changed with software commands. (For you 4K owners, try typing in `POKE 65314,255` **ENTER** and see what it does. You'll have to turn off the machine to regain control of the computer video output to the screen. It still works OK, but not in a form that you understand. For you Extended BASIC owners, you'll need to ENTER a program line and run it — i.e.

```
10 POKE 65314,255: GOTO 10
```

Just press **BREAK** to get back to normal video.) During the power-up sequence, the computer first checks the Program Pak slot to see if a cartridge is present. If there is one present, the computer gives control over to the Program Pak cartridge; if not, the computer continues on with the initialization of BASIC. During the start-up of BASIC, the computer will check to see what RAM is available for it to use. This is why, when the machine is first turned on, you can do a `? MEM` (that's PRINT MEM in computer shorthand) and it can answer you with a value corresponding to the available RAM. Now, on to the differences in the machines:

With the 4K machine (yes, there really is a set of 4K RAM chips in there), when you do a `? MEM` the computer responds with 2343. "But, it's a 4K machine!," you say. "Where did the 1657 bytes go?" "Did someone forget a set of chips?" No, it's all there and actually it's a 4096 byte machine. Here's what happens to the 1753 bytes: Decimal addresses (remember the memory locations —sometimes referred to in Hexadecimal) from 0 through 1023 (1024 bytes) are reserved in RAM for system use. Addresses from 1024 through 1535 (512 bytes) are used for the text screen memory, 200 bytes somewhere between 1536 and 4096 is reserved for string storage. When the sequence is complete (it only takes milliseconds), it should leave you with 2360 bytes of "free" RAM for program and variable storage. "Then where did the 2343 come from?," you ask. Well, somewhere in the process, an additional 17 bytes are allocated and I don't honestly know where they go. By using the PEEK and POKE functions available in Color BASIC, you should be able to utilize a portion of the RAM which the computer has grabbed for itself. Instead of storing variable values in your program space, POKE them into the text screen area of memory (use the CHR$ value with numbers between 0 and 255), and PEEK them back later when you need them. You might even try POKEing them in the range of Decimal 256-1023, but I can't guarantee what will happen. The RAM memory that the computer has taken is being shared with you. The computer will not overtake your 2343 bytes but you can use a portion of the RAM memory which the computer has taken (if, in your program, the computer does not need or use it). By using PEEK and POKE (syntax for these are in the Getting Started and Going Ahead with Extended BASIC Manuals), you have effectively raised the amount of RAM memory available for use. A BASIC program cannot be placed in the shared area of RAM, but it certainly doesn't stop you from trying to use that RAM for your own purposes.

The situation is similar in the Extended BASIC machine with a few exceptions (primarily memory locations and extra ROMs). Decimal addresses from 0 through 1023 (1024 bytes) are used for system purposes. Addresses 1024 through 1535 (512 bytes) are used for the text screen memory. Decimal addresses from 1536 through 13823 (12288 bytes) can be used (in steps of 1536 bytes) for graphics displays. Using the `PCLEAR 1` command, the minimum amount of RAM is reserved for the graphics screen memory (1536 through 3071 or 1536 bytes), leaving you with 13095 bytes of RAM available plus 200 bytes of string storage space, based on a 16K machine having 16384 bytes of RAM memory to begin with. (The same 16 or 17 bytes disappears in Extended BASIC also.) If this is not enough RAM, and you are not going to be using the computer to generate graphics, then try POKEing and PEEKing the graphic screen memory locations in RAM (decimal 1536 through 3071 using the CHR$ values with numbers from 0 to 255). This would, in effect, give you 14631 bytes of RAM memory available for use. As in the standard Color computer, a BASIC program cannot be stored in the graphic memory locations, but values can be POKEd and PEEKed there. Try it, you might like it.

Volume III of the TRS-80 Applications Sourcebook has been sent to press with over 2300 listings of programs for the various TRS-80s. (Some from us and a bunch from independent/outside vendors). For those of you who have programs that you want listed in the Sourcebook, any local Radio Shack store should be able to supply you with the proper form to fill out and send in. The listing fee is $10.00 and is good for one year from the first date that your listing shows up in a Sourcebook. (If you miss a deadline, never fear, your listing will show up in the next one and your year does not start until it shows up in print.) For those of you who listed programs in the first edition, now is the time to count your pennies and renew your listing. We'll be trying to mail you a notice that your time is almost up prior to when it expires so you won't miss any editions. Remember, the Sourcebook is only for applications software. No systems software will be accepted. Radio Shack determines what is systems software. So far, the TRS-80 Applications Software Sourcebook has been a tremendous success with 10s of thousands of copies sold since its introduction. Thank you for your support.

New Subject— Old Program:

Dig out last month's Newsletter and enter the house program (or at least the setup portion for graphics— lines 1-20). To line 17 add a `GOSUB 1000`. Next key in the following subroutine:

```
1000 DRAW"BM0,30;R52;U30": PAINT(0,0),3,4
1010 DIM W(36,16)
1020 GET (12,12)-(48,48),W
1030 DRAW"BM16,16;R8;E4;R8;F4;R8;G4;
     L8;NE4;H4;L8;G4;L8;NE4;D4;R4;BR4;NU4;R8;NU8;R8;BR4;R4;NU4;
     E4;U4;G4;L24"
1040 CIRCLE(18,24),5: CIRCLE(38,24),5
1050 PAINT(18,18),2,4: PAINT(30,14),2,4:
     PAINT(42,18),2,4: PAINT(46,20),2,4:
     PAINT(42,22),2,4: PAINT(32,22),2,4:
     PAINT(24,22),2,4: PAINT(14,22),2,4:
     PAINT(18,24),2,4: PAINT(38,24),2,4
1060 DIM V(36,16)
1070 GET (12,12)-(48,48),V
1080 PCLS: RETURN
```

(Continued on Page 14)

Color Computer (From Page 13)

So far we have introduced 2 new commands: GET and CIRCLE. GET allows you to store graphics in a predefined array for future call-back to the graphics screen. The size of the graphics stored in the array is limited to the size of the array and the available memory for the array. You simply DIMension the array, then specify the upper left x,y coordinate and the lower right x,y coordinate along with the variable name for the array. (In this case W and V). I'm only going to offer a brief description for CIRCLE this month: CIRCLE with the center point at (x,y) coordinate, having a radius of 5. (More on CIRCLE in a later article.)

Now add these lines to use what you've just done:

```
300 FOR C=250 TO 15 STEP -15
310 PUT(C,162)-(C+36,178),W
320 PUT(C-15,162)-(C+21,178),V
330 NEXT C
340 PUT(15,162)-(51,178),W:GOTO 300:
    'OPTIONAL LINE FOR REPEAT
```

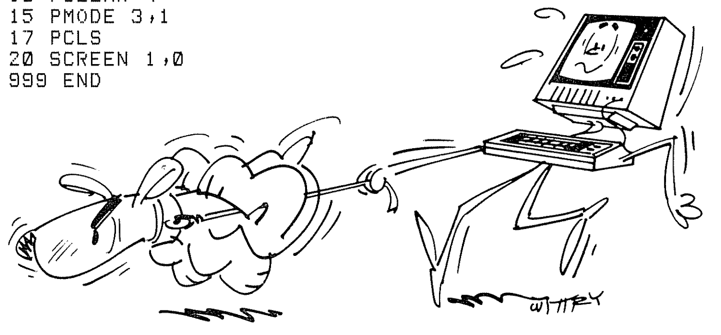
If you do not want a repeat, simply omit line 340. The program line 999 from last month's program will prevent the program from crashing into the sub-routine.

Now an explanation: In lines 1000-1080 we have drawn a blue square and saved it in one of the arrays. We have also drawn a small car and saved it in an array. After the house is drawn, we then have a loop in the program which PUTs the car alternating with the blue square across the screen. It happens fast enough that we now have animated motion of traffic on the screen in front our house. If line 340 is deleted, then the car will stop in front of the house. (By the way, the program will run without the house being drawn, namely just the subroutine with the PUTs in line 300-340 as long as you have the first 6 lines from last month's program setting up graphics screens on the video.)

Try some experiments with these new instructions. If you run into any problems, refer to the Going Ahead with Extended BASIC manual. You'll find a wealth of information waiting for you.

Editor's Note: For those of you who may not have seen the previous version of this program, this program requires Extended Color BASIC with 16K of memory. The lines you need from the previous version are:

```
1 DRAW"S4"
5 CLS
10 PCLEAR 4
15 PMODE 3,1
17 PCLS
20 SCREEN 1,0
999 END
```

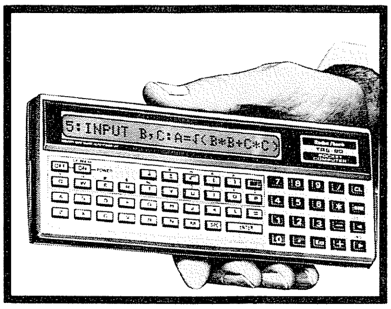


SAFE

Steve Lewallen   Centerville, Ohio

I am 10 years old, and live in Centerville, Ohio. This program was done on a Radio Shack Color Computer with 16K memory and Joysticks.

```
10 REM BY STEVE LEWALLEN, 1981
20 CLS
30 PRINT@100, "SAFE"
31 PRINT@196, "BY STEVE LEWALLEN, 1981"
32 FOR D=1 TO 1000: NEXT D
40 FOR G=1 TO 240 STEP 1: NEXT G
50 PRINT "YOU ARE A BURGLAR AND HAVE ENCOUNTERED A SAFE. TO OPEN
   THE SAFE USE THE LEFT JOYSTICK, TO SEARCH FOR THE RIGHT NUMBER.
   THE COMPUTER WILL ACT AS A SAFE AND GIVE YOU A SORT OF CLUE:
   YOU WILL HEAR A 'CLICK' WHEN YOU PASS THE NUMBER."
60 PRINT: PRINT "YOU WILL ALSO SEE THE WORD CLICK WHEN YOU ARE ON
   THE NUMBER. PRESS THE RED BUTTON TO GO TO THE NEXT NUMBER. MOVE
   THE JOYSTICK SLOWLY."
70 FOR G=1 TO 10087: NEXT G
80 CLS
90 CLS
100 R=RND(63)
110 A=JOYSTK(0)
120 P=PEEK(65280)
130 PRINT@1, A;
140 IF A=R+1 THEN 210
150 IF A=R AND (P=126 OR P=254) THEN 280
160 IF A=R THEN 230
170 GOTO 110
180 CLS0: SOUND 200,1
190 GOTO 110
200 GOTO 270
210 SOUND 1,1
220 GOTO 110
230 PRINT@196, "***CLICK***";
240 FOR X=1 TO 300: NEXT X
250 CLS
260 GOTO 110
270 GOTO 90
280 PRINT"FIRST DIGIT SOLVED"Q;
290 FOR X=1 TO 300: NEXT X
300 CLS
310 R=RND(63)
320 A=JOYSTK(0)
330 P=PEEK(65280)
340 PRINT@1, A;
350 IF A=R+1 THEN 410
360 IF A=R AND (P=126 OR P=254) THEN 490
370 IF A=R THEN 430
380 GOTO 320
390 GOTO 320
400 GOTO 320
410 SOUND 1,1
420 GOTO 320
430 PRINT@196, "***CLICK***";
440 FOR X=1 TO 99: NEXT X
450 CLS
460 GOTO 320
470 GOTO 300
480 Q=1 TO 100 STEP 1: NEXT Q
490 PRINT "SECOND DIGIT SOLVED"Q
500 FOR X=1 TO 350: NEXT X
510 CLS
520 R=RND(63)
530 A=JOYSTK(0)
540 P=PEEK(65280)
550 PRINT@1, A;
560 IF A=R+1 THEN 630
570 IF A=R AND (P=126 OR P=254) THEN 690
580 IF A=R THEN 650
590 GOTO 530
600 SOUND 200,1
610 GOTO 530
620 GOTO 680
630 SOUND 1,1
640 GOTO 530
650 PRINT@196,"***CLICK***";
660 CLS
670 GOTO 530
680 GOTO 510
690 QW=RND(6)
700 PRINT "YOU HAVE OPENED THE SAFE, AND TO YOUR WONDERING EYES DO
    APPEAR:"
710 ON QW GOTO 720, 730, 740, 750, 760, 770
720 PRINT@196,"A ROLLS ROYCE": GOTO 790
730 PRINT @196,"TWO COPS THAT HANDCUFF YOU": GOTO 790
740 PRINT@196,"A POT OF GOLD": GOTO 790
750 PRINT@196,"NOTHING": GOTO 790
760 PRINT@196,"A SECRET DOOR TO THE MINT": GOTO 790
770 PRINT@196,"10 OF EACH RADIO SHACK ITEM": GOTO 790
780 END
790 V$=INKEY$
800 PRINT@300,"DO YOU WISH TO PLAY AGAIN (Y OR N)": INPUT V$: IF
    V$="Y" GOTO 90
810 CLS
820 PRINT "COME BACK AND WE WILL DO IT AGAIN SOMETIME."
830 END
```



Pocket Computer

Product Line Manager's News

As we begin another Summer here in Texas, I have some very exciting news for you . . . the introduction (Official) at the end of June of the long awaited printer for your Pocket Computer. Since it's June already, I thought I'd give you all (or yawl to you seafarers) a sneak preview!

So, let's get right to it with some statistics to wit:

> 16 Column Dot-Matrix impact printer.
> Replaceable ribbon cartridge.
> Can use ordinary 1¾ inch Electronic Cash Register (ECR) paper.
> Built-in Cassette Interface.
> Built-in rechargeable batteries.
> PRINT and LIST speed is about one line per second.
> Cat. No. 26-3505, $149.00.

Now for some specifics. The printer comes packed with an AC Charger/Adapter, one ribbon cartridge, three rolls of paper, detachable cassette cable and Owner's Manual. Instead of including a mediocre case with the printer, we felt it would be better to provide a very high quality carrying case as an option for $17.95 (Cat. No. 26-3508). The case is well padded, will hold the printer with the PC installed as well as an extra ribbon, roll of paper, several pens, the Quick Reference card, printer Owner's Manual and some business cards . . . and will fit nicely in a three inch attache or can be carried via its own handle. In addition, we have designed the case so you can use the PC/Printer combination without taking it out of the case. You will also find that you can use a second case to hold the Minisette-9, the AC Adapter/Charger, Cassette Cable and program tape . . . thus providing the complete Pocket Computer system at hand ready for travel.

You are probably wondering why we put another Cassette Interface in the printer when you already have an interface. Well, if we had left it out, it would have required some extra cables, made the unit(s) rather cumbersome and only saved you a few dollars . . . we just felt the trade-offs dictated including the interface. Besides, you will probably find it handy to have another interface. One feature of this interface that I am sure you will find useful is the Remote switch. This switch allows you to Rewind or Fastforward the tape without unplugging the cassette remote plug, if you have a Minisette-9 or other similar recorder.

Let's look at the Printer in operation. When you slide the PC onto the Printer (just like the present interface) and turn them both on, the PC can tell that it is connected to the printer. At this point with the Printer 'Print switch' ON, both the PRINT and LIST instructions will run continuously instead of halting the PC as before and they do not display on the LCD. The LIST (ln) and PAUSE instructions work the same as before. This allows you to list an entire program and to PRINT a little over 70 characters with the printer wrapping around at the end of a line. (see accompanying photos) In the LIST mode, the program lines are indented from the line numbers to make it easy to read a program. The PRINT instruction works nearly the same as it does on the LCD, the differences are in the handling of large numbers due to the 16 character line versus the 24 character LCD. The Owner's Manual explains the only differences in this regard.

```
SAVINGS AND LOAN
SAVINGS:SHFT A
LOANS:SHFT B
EXIT:SHFT K
```

```
1:"K"END
2:IF A=UPAUSE
   Z$:RETURN
3:PAUSE A;" ":
   RETURN
5:IF A=0LET A=
   -1
6:IF A=ULET C=
   CW+Y:A=1
7:IF A<0BEEP 2
   :Y=0
```

From a mechanical standpoint, the printer is rugged and reliable. In fact, we have several printers that have logged over 500,000 lines of print without a failure. So I am sure that you will receive many years of service from this unit.

I mentioned earlier that the printer uses ordinary ECR paper. We will supply this paper in special 1 inch diameter rolls to fit inside the printer. This paper is available as Cat. No. 26-3506 at $1.75 for a box of 6 rolls and each roll is about 18 feet long. To give you an idea of what this means in terms of a listing, an average 1424 step PC program with multiple statement lines will use about 22 inches of paper. The printer will print about 8000 lines on a full charge and may be used while it is charging although the charging time will increase substantially.

The printer ribbon is contained in a small plastic cartridge and uses a continuous nylon band with dark blue/black ink. The ribbon is self inking and should last three to four rolls of paper under normal conditions. Replacement ribbons are available under Catalog Number 26-3507 for $2.75 each.

As I said earlier, this is a sneak preview so not every Radio Shack store will have this printer by the time you read this. If all goes as planned, every Computer Center and most Radio Shacks in major cities should have at least one unit by mid-June. We are planning an introduction in the press around the end of June so if you want one (for that new PC you bought during our May sale), go down and put in your order as they may be in rather short supply at first.

I hope this news has whet your BASIC appetite and that you are as excited as I am about this new capability for the TRS-80 Pocket Computer!

Pocket Bugs, Errors and Fixes

The BUG: Cat. No 26-3516, Business Statistics. Two programs in this package, Multiple Regression (MR) and Forecasting (FC) will not accept negative numbers as variables. This is due to an error in the PC ROM algorithm which raises a number to a power . . . it gives an error if you try to raise a letter variable containing a negative number to a power. If you try to raise a simple negative number (i.e. −2) to a power, it gives the wrong answer

(Continued on Page 16)

Pocket Computer (From Page 15)

for all even powers, i.e. −2 raised to the second power gives −4.

The FIX: In the MR program, change line 110 to read:

```
110 D=D+X*X:E=E+Y*Y:F=F+Z*Z
```

In the FC program, change line 150 to read:

```
150 G=G+X*A(J):F=F+X*X:I=I+A(J): NEXT K
```

After you make these changes, it is possible to re-record the changed program back onto your original program cassette without hurting the other programs, IF you are VERY CAREFUL! Proceed as follows:

1. Locate the desired program on tape.
2. Load it into the PC.
3. Rewind to the beginning of the program and verify it.
4. Rewind the tape to the beginning of the program again.
5. With the recorder in Play, use the Pause key to stop the recorder the moment you hear the first steady tone of the program.
6. Press Stop on the recorder. Release the Pause key.
7. Remove the tape cassette and using a pencil, back up the tape about 1½ inches from where you stopped it using the Pause key. This will correctly position the tape to record over the original program.
8. Make the modifications to the program in the PRO mode as indicated above. Now double check them.
9. Place a piece of tape over the record protect notch on the cassette and put the cassette back in the recorder. Make sure the recorder is connected to the PC interface and set the controls for Record.
10. Type CSAVE "program name," program name being whichever program (MR or FC) you are working with and then press ENTER.
11. When the tape stops, it has re-recorded your corrected program back on your tape. Now rewind the tape to the beginning of the program and verify it the same as in step #3 above. You should now have a good corrected copy of the program. You can do the same to the program on the other side of the tape simply by locating the program and setting up the tape as in #7 above. Don't forget to remove the piece of tape.

Well, that's it for this month, enjoy your printer and until next month . . . more Pocket Power to you.

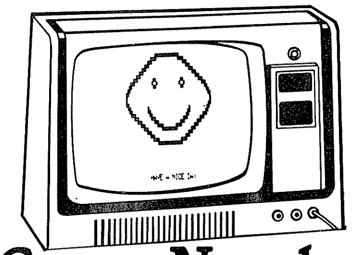
TC

Mr. Shawn Stricklin Richardson, Texas

Last Christmas, our family bought a Color Computer. We have all enjoyed it immensely; we also enjoy the Microcomputer News. I am always looking for new programs, of which there are few. I wrote the program below as much for Radio Shack as for your readers. The program is as follows:

```
10 CLS0
15 E=63
20 FOR Q=1 TO 15
30  W=W+1: E=E-1
40  SET(W,Q,3)
50  SET(E,Q,3)
60 NEXT: W=63: E=0
70 FOR Q=31 TO 16 STEP -1
80  W=W-1: E=E+1
90  SET(W,Q,4)
100  SET(E,Q,4)
110 NEXT
120 FOR R=16 TO 47
130  SET(R,16,3): NEXT
140 FOR A=15 TO 48
150  SET(A,2,4): SET(A,3,4): NEXT
160 FOR S=30 TO 33: FOR D=4 TO 15
170  SET(S,D,4): NEXT D,S
180 SET(15,4,4): SET(48,4,4)
190 FOR Y=17 TO 46
200  SET(Y,18,3): SET(Y,19,3)
210  SET(Y,30,3): SET(Y,31,3)
220 NEXT
230 SET(46,20,3): SET(46,29,3)
240 FOR Z=17 TO 20: FOR X=18 TO 29
250  SET(Z,X,3)
260 NEXT X,Z
270 FOR C=1 TO 1000: NEXT
280 RUN
```

Well, there it is! Have fun!



Guess Number

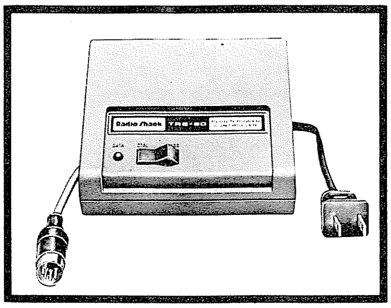
Les Childers of Pierre, South Dakota

Here's a fairly simple program that I developed for my recently purchased TRS-80 Pocket Computer . . . Note: the range of possible values that the computer will generate for you to guess is from about −50 to almost 84000!

Program "GUESS NUMBER −2" or whatever filename you assign.

```
10:PRINT"ENTER NUMBER 1 TO 1000"
20:INPUT X
30:IF X> 1000 BEEP 2: PAUSE"EXCESSIVE INPUT": GOTO 10
40:X= INT(X* X/ 12+ 10- 30)
50:PRINT"YOUR GUESS IS"
60:INPUT Y
70:IF Y= X THEN 110
80:IF Y> X THEN 130
90:PRINT"YOUR GUESS IS LOW"
100:GOTO 50
110:BEEP 5: PRINT" YOUR GUESS IS CORRECT"
120:GOTO 10
130:PRINT"YOUR GUESS IS HIGH"
140:GOTO 50
150:END
```

Mr. Childers mentions that you can vary the formula in line 40 to suit your needs. Line 40 serves as a simple "pseudo-random number generator."



Peripherals

Product Line Manager's News

This month I will try to make good some of my promises!

Yes, there is a PLUG 'N POWER controller. They should be in stock at your nearby, friendly dealer by now. (Please don't tell Murphy! Your "now" is two months later than my "now," so they really should be there.)

The TRS-80 PLUG 'N POWER CONTROLLER (26-1182) works with any TRS-80 Model I, Model III, or Color Computer using the cassette output. At $39.95, the controller is a real bargain. The package contains not only the hardware to control the wireless modules sold by Radio Shack, it also provides software for all configurations of these computers including Level I!

The modules sold by R.S. normally allow a push button control center (61-2680) to control off/on and light dimming operations. Sixteen units can be controlled from the unit at a time. By setting a "house code" on the bottom of the unit up to 16 sets of 16 modules can be controlled.

The computer controlled unit can command all 256 possible modules under software control — no switches are necessary. These are the devices sold by R.S.:

Appliance Module (OFF/ON function)	61-2681
Lamp Dimmer (OFF/ON and dimmer function)	61-2682
Wall Switch Module (OFF/ON and dimmer function)	61-2683
Universal Appliance Module (Has three wire grounded socket)	61-2684

The controller plugs into the cassette socket on the CPU. There is a cassette socket in the device to allow connection of the cassette. A switch allows either normal cassette or controller functions, so there is no need for plugging and unplugging cables.

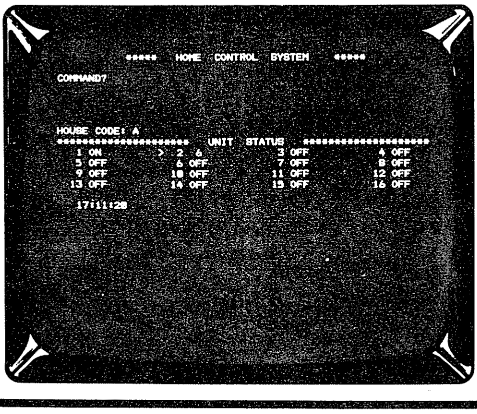
There is software provided for:

Model I/III Level I Machines (4K)
Model I Level II Machines and
Model III Model III BASIC Machines (16K,32K,48K)
Color Computers (4K,16K)

The use of the TAPEDISK utility on Model I's and the TAPE utility on Model III's is supported so that Disk users can transfer the programs to TRSDOS.

After loading the appropriate program into your computer, the software will ask for the time of day. This sets up the Real Time Clock (RTC) used for event timing. This RTC is available even in Level I and Model I systems without the expansion interface.

The software provides two modes of operation. The DIRECT COMMAND MODE allows control of the modules directly from the keyboard. After entering an appropriate "house code" the last known status of the sixteen units associated with that code are displayed.



```
          *****  HOME  CONTROL  SYSTEM   *****

COMMAND?



HOUSE CODE: A
*********************   UNIT  STATUS   *********************
   1 ON          >  2  6            3 OFF            4 OFF
   5 OFF            6 OFF           7 OFF            8 OFF
   9 OFF           10 OFF          11 OFF           12 OFF
  13 OFF           14 OFF          15 OFF           16 OFF

    17:11:20
```

Perhaps we should mention one important consideration at this point. This class of device allows only one way communication. The CPU can COMMAND, but it has no way in the world of telling if the command was obeyed.

In addition to ON/OFF commands the dimmer modules can be set to 10 units of brightness (or dimness — depends upon if you are an optimist or a pessimist). In the example shown here unit "2" is set to a level of "6." The use of DIM n or BR n commands can alter the status of the module.

There are commands to control all switch actions at once as well as a Quit command which transfers control back to BASIC.

There is even a HELP command for those of us who can't remember from one day to the next how in the heck the thing works! The HELP command is automatically invoked if some dummy enters an improper command.

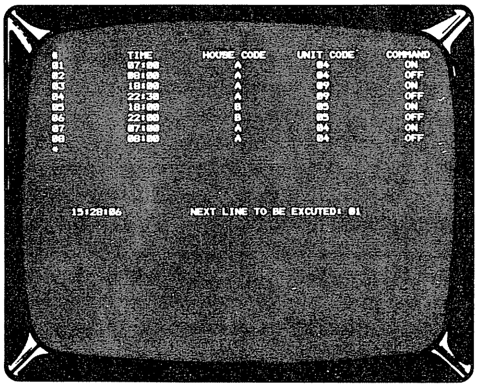
In the PROGRAMMING Mode, modules may be programmed to execute unit commands at a predetermined time. Neat for fooling burglars, setting up nifty sequential displays, etc., etc., etc. . . .

The total number of command lines available varies with the computer:

Level I	32 events
Level II/Model III BASIC	45 events
Color	30 events

Command functions allow INSERTION of new lines, REPLACEment of lines, EXECUTION of lines, you can DELETE lines, there is a LIST and PAGE display and a return to the direct command mode.

By setting all timed events in one 24 hour period a one day routine can be provided which will repeat each day. The programmer can also set up multi-day patterns with the use of "dummy" commands at certain times in the clock cycle.



```
#          TIME        HOUSE CODE     UNIT CODE     COMMAND
01         07:00           A             04            ON
02         08:00           A             04            OFF
03         18:00           A             09            ON
04         22:30           A             09            OFF
05         18:00           B             05            ON
06         22:00           B             05            OFF
07         07:00           A             04            ON
08         08:00           A             04            OFF
*



   15:28:06          NEXT LINE TO BE EXCUTED: 01
```

We don't pack it up here folks. There's more. The controller can be used with other BASIC programs (except Model I/III Level I programs) as a subroutine. Use of the program module in another BASIC program does not result in loss of data in the programmed event list, but the RTC function stops. The Owner's Manual gives short examples of this feature.

In addition, command lines stored in the program can be saved to disk (providing you have one!). AND . . . last but not least, the manual provides a shortened controller program for 16K, 32K, or 48K Disk BASIC to be used in more elaborate special function programs you may want to dream up. We tried to think of everything, but we gave you the tools to "do your own thing" if you want too. (Continued on Page 20)

Line Printer VII Product Review

Gary Katz East Brunswick, New Jersey

I have just purchased your new product, the Line Printer 7 and I am truly overjoyed about it and I would like to pass on a few hints about this fantastic printer. First off, I would like to remind the readers of this newsletter that the seven is a GRAPHICS PRINTER! Because of this, I designed this program to produce an actual screen printer. All of the other routines printed the content of the screen with ASCII characters, or a black on white picture. I would like all of you Line Printer Seven owners who also have a Model I or III to try out this routine:

```
1000 FOR Y=0 TO 47
1010   FOR X=0 TO 127
1020     IF POINT(X,Y)
             THEN LPRINT CHR$(28); CHR$(3); CHR$(255);
             ELSE LPRINT CHR$(28); CHR$(3); CHR$(128);
1030   NEXT X
1040   LPRINT CHR$(18)
1050 NEXT Y
```

What this routine does is scan the screen just as all the previous programs did, but if the Pixel (X,Y) is set, the printer will print nothing but a space of three graphics characters. If the Pixel (X,Y) is not set, the printer will print a string of three solid graphic characters. When the printout is done, (and it will take some time!) the printout will be an exact duplicate of what was on the screen at the time of the running of this routine EXCEPT alphanumerics. I am now in the process of writing a combination program, but I just wanted to send this in right now along with my commendations on an excellent product as well as some following hints and tips.

A second hint that I have found out about is that one can store graphic codes in a string just as it can be done with CHR$(x) and strings on the video display. Using this, a person can literally design their own alphabet or character set, store it in a one-dimensional array (string) and then can print them out as graphic characters on the printer. When doing this, make sure that the printer is in the graphic mode, or else one could end up with a whole mess of line feeds! That, I found out the hard way.

Again, I want to commend you, all of you at Radio Shack, for a job well done.

(A fascinated 13 year-old)

Shoot

Barry Levinson San Francisco, California

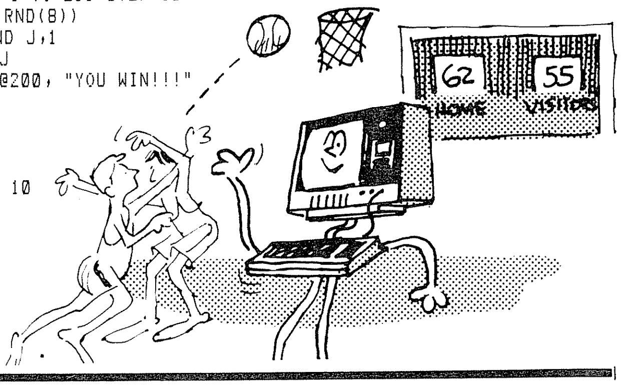
I own a TRS-80 Color Computer and I enjoy programming it. My favorite area in programming is programming games, mainly because it is fun to play them when I am done programming them. I have enclosed a copy of one of my favorite games which I have written in hopes of having it put in the Newsletter.

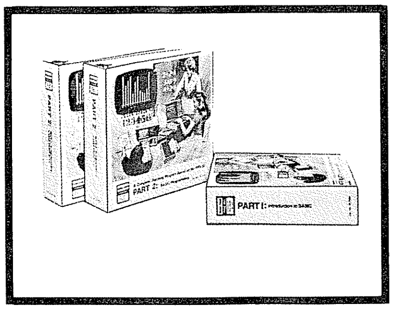
There are a few changes you might want to make: Add line `325 GOSUB 150: GOSUB 111` and if you have a preference for another color, change line 10 to `CLS (1-8)` (put in the number of the color you want).

The directions for the game are: Press Ⓕ to fire (you will have to press it several times, I do not want this to be too easy). Press Ⓖ and Ⓗ to move along the bottom of the screen. I love to program the Color Computer but I find myself restricted with just BASIC. I know that the Color Computer has the capability to use other languages, but when will they come out, or at least a book on machine languages?

Editors Note: Barry forgot to add line 700, so we added a little "reward" for you when you shoot twenty of the "enemy." Barry, we appreciate your comments, and can assure you that "at least a book on machine language" is coming. Have patience, you should have received the expanded version of "Getting Started with COLOR BASIC" which we began mailing to all owners of Color Computers in April. The back of that manual contains a lot of information which should let you program high resolution graphics, create music, and do a lot of other very neat things.

```
1 A=1: B=2: C=3: D=4: E=5: F=6: G=7: H=8
9 GOTO 1000
10 CLS
20 A=A+1: B=B+1: C=C+1: D=D+1
30 E=E+1: F=F+1: G=G+1: H=H+1
100 I=I+1: J=J+1
105 GOSUB 111
106 GOTO 120
110 J=J+1
111 A$=INKEY$
112 IF A$="H" THEN K=K+1
113 IF K<480 THEN K=480
114 IF K>512 THEN K=512
115 PRINT@K, "++";
116 IF A$="G" THEN K=K-1
117 RETURN
120 L=L+1
125 A$=INKEY$
126 PRINT@1,XX;
130 IF A>500 OR B>500 OR C>500 OR D>500 OR E>500 OR F>500 OR G>500
    OR H>500 OR I>500 OR J>500 OR L>500 THEN END
140 IF INKEY$="F" THEN 300
141 GOSUB 111
145 GOSUB 150
146 GOTO 250
150 PRINT @A, "*";
160 PRINT @B, "*";
170 PRINT @C, "*";
180 PRINT @D, "*";
190 PRINT @E, "*";
200 PRINT @F, "*";
210 PRINT @G, "*";
220 PRINT @H, "*";
230 PRINT @I, "*";
240 PRINT @J, "*";
245 PRINT @L, "*";
246 RETURN
250 GOTO 10
300 II=K
305 GOSUB 150
306 GOSUB 111
310 II=II-32
311 IF II<1 THEN 10
320 PRINT@II,"/";
340 IF II=A OR II=B OR II=C OR II=D OR II=E OR II=F OR II=G OR
    II=H OR II=I OR II=J OR II=L THEN XX=XX+1
341 IF XX=20 THEN 700
342 IF II=A THEN A=0
343 IF II=B THEN B=0
344 IF II=C THEN C=0
345 IF II=D THEN D=0
346 IF II=E THEN E=0
347 IF II=F THEN F=0
348 IF II=G THEN G=0
349 IF II=H THEN H=0
350 IF II=I THEN I=0
351 IF II=J THEN J=0
352 IF II=L THEN L=0
353 GOTO 310
700 FOR J=1 TO 255 STEP 10
710   CLS(RND(8))
720   SOUND J,1
730 NEXT J
740 PRINT@200, "YOU WIN!!!"
750 END
1000 I=9
1010 J=10
1020 L=11
1030 GOTO 10
```





Preparing Your School for the Computer Age

The Education Page in the April issue of Microcomputer News contained descriptions of some of the teaching and administrative materials made available by the Education Division of Radio Shack. This month's article presents more information about using computers as the medium, object, and the manager of instruction.

The TRS-80 as a Medium of Instruction: COMPUTER ASSISTED INSTRUCTION

Computer Assisted Instruction (CAI) is nothing new to the educational scene. Numerous approaches for the use of computers to supplement existing educational practices have been well established and have proven effective in increasing student achievement.

CAI is based on the use of the computer as the medium of instruction — as a means to assist in teaching subjects such as reading, mathematics, language arts, physics, business, and chemistry. The techniques that have proven effective for this use of computers have centered on individualized learning sessions where the computer generates and presents exercises in an appropriate subject area for a student to solve.

The computer can measure the student's performance and adjust the difficulty of the problems. As a student's performance indicates an increased level of understanding, the computer can generate more difficult exercises. If a student's score indicates a problem in a particular area, the computer can adjust the difficulty level to provide easier exercises for additional practice.

There are several advantages to this use of computers for instruction:

- Instruction is highly INDIVIDUALIZED. A student moves at his own pace. He/she is constantly challenged, but never threatened.
- The computer can assist the teacher as a DIAGNOSTIC tool for measuring individual strengths and weaknesses. As a student moves through the lessons, the computer can record performance information to be reported to the teacher later.
- The individualized learning process is NON-THREATENING to the learner. The computer can be programmed to correct mistakes in a positive way and to praise appropriate responses. To the student, a session at the computer can become like an exciting game.
- The approach is commonly a SUPPLEMENTAL one. The teacher introduces and explains new concepts; the computer can present exercises on those concepts, provide immediate feedback and reinforcement to the student, and record the scores. The teacher teaches; the computer assists the teacher by reinforcing concepts, collecting performance information, and reporting that information to the teacher.
- The computer is a TIME-SAVING device for the teacher. Since it can not only generate exercises, but also grade those exercises and report the grades to the teacher, it frees the teacher to spend more time with individual students.

CAI systems have been commonly implemented using expensive minicomputer timesharing systems. The microcomputer offers educators an alternative to the timesharing systems — an alternative that not only costs less, but one that provides the added benefits of portability through freedom from modems and telephone lines.

With the TRS-80 microcomputer and Radio Shack courseware, a school can have a CAI system with the features we have described and more, and at a lower cost per student hour than has ever been possible before.

CAI courseware packages currently available from Radio Shack (or soon to be released) include:

- High Motivation Reading Series (including Charles Lindbergh/ Amelia Earhart and Hound of the Baskervilles)
- AlphaKey™
- K-8 Math Program, Volume One
- K-8 Math with Student Management Program, Volume One
- K-8 Math Worksheet Generator
- Color Math
- Euclid Geometry Tutor
- Essential Math Program, Volume One
- Numeric Data Entry Practice
- Investigations in Integral Calculus
- Advanced Graphics
- Vector Addition
- Interpreting Graphs in Physics
- Graphical Analysis of Experimental Data

The TRS-80 as an Object of Instruction: THE COMPUTER EDUCATION SERIES

Computers are becoming more and more common in today's world. No longer limited to applications in research institutions or large businesses, computers — including small microcomputers — are turning up in applications ranging from supermarket checkouts, automated bank tellers, and small business systems to such consumer items as improved thermostats, television tuners, microwave ranges, and ignition systems on some new automobiles. No other product of twentieth-century technology seems destined to have such a profound influence on the way we conduct our daily lives — the way we operate our businesses, manage our homes, and educate our children.

It is not surprising that more and more educators are investigating the impact of computers on the education process. They are looking not only at computers as tools for teaching, but also at the need for schools to prepare students to function in a world increasingly influenced by and dependent on the use of computers. In a matter of only a few years, people who know nothing about computers may well be among the educationally disadvantaged portions of the population.

Radio Shack is addressing these needs with our Computer Education Series designed for use in the classroom. Two parts of this series, Part 1: Introduction to BASIC and Part 2: BASIC Programming, are now available. Part 3: Advanced BASIC will be available soon. Each part is a complete teaching program designed for classroom use with the TRS-80 microcomputer at the secondary and college level. Each includes a teacher's manual, student workbooks, and a set of transparencies for an overhead

(Continued on Page 20)

Education (From Page 19)

projector. (Additional student workbooks are available separately.) Developed and field-tested in public schools, this material is designed for use by all teachers regardless of their knowledge of microcomputers and programming skills. Additional programs are forthcoming in the Computer Education Series.

The TRS-80 as a Manager for Teachers and Administrators: EDUCATIONAL MANAGEMENT SYSTEMS

Radio Shack offers a variety of materials to assist teachers and administrators in creating their own CAI materials and performing clerical and record-keeping tasks.

Programs which help teachers create their own CAI materials are called AUTHORING SYSTEMS.

Teachers can write any kind of program if they know BASIC, the language used by most microcomputers. But the large number of teachers who do not program in BASIC can still use the computer to the fullest advantage with Radio Shack's authoring systems. If, for example, a teacher cannot find suitable CAI material in the desired subject area, he/she can custom-design courseware to meet his/her own specifications. The teacher can design materials to lead students through an instructional sequence as quickly or as slowly as desired, with as much or as little computer assistance as desired.

"Screen-oriented" authoring systems permit teachers to type material into the computer exactly as they want it to appear during the student sessions. The teacher types in the screens one at a time, complete with text explanations and graphics which the student will see and questions to which the student will respond. The teacher also directs the sequence so the computer will present the screens in the desired order. The teacher can instruct the computer to go back to a certain screen if a student misses a certain number of exercises, or to advance to a different screen if another criterion is met. The computer will automatically keep track of the student's responses and the average response time. Radio Shack's TRS-80 AUTHOR 1 is a screen-oriented authoring system which includes these features and many more.

"Command-oriented" authoring systems allow the teacher to write a set of instructions that the computer can interpret. When the computer executes the instructions, text and graphics appear on the screen and the computer awaits student responses. A command-oriented authoring system's flexibility is derived from the wide range of commands it has available. Radio Shack's TRS-80 PILOT Plus is a command-oriented authoring system with many powerful one-letter instructions for text, graphics, and calculations.

Other types of authoring systems are available from Radio Shack as well. QUICK QUIZ is a mini-authoring system that allows teachers to create, administer, edit, and print out multiple-choice tests, and record student scores. Another type of system prompts the teacher through the lesson-creating procedure with questions. By typing in appropriate responses, the teacher instructs the computer to build the instructional sequence. Additional authoring systems of this type will be offered by Radio Shack.

We also have several programs which help with clerical and record-keeping tasks. PROFILE is a general-purpose computerized filing system. Teachers, administrators, and other school personnel can use it to store student records, grades, inventories, library information, and many other types of records. It allows the user to type in the information, then later to display it on the computer screen, print it on a printer, update it, and sort it. With teacher-designed record formats, PROFILE is an extremely versatile program.

VISICALC is another management program with many school applications. It sets up an "electronic worksheet" on which the user may type numbers, labels, and information for calculations. The program automatically updates the calculations each time the user gives it new numbers. VISICALC may be used for computing grades, enrollment figures, and budgets. Another effective use of VISICALC is to make projections in budgets and enrollment figures, easily changing one or two numbers to see how they would effect the whole picture.

SCRIPSIT is Radio Shack's word processing program. With it, a user can type a letter, article, or other document into the computer. The text for the document appears on the screen as it is typed. Before printing it out, the typist can correct any errors and change the spacing, margins, etc. as desired. The document can be stored on a diskette to be used again later. SCRIPSIT is a very flexible program that turns your TRS-80 into a powerful word processor.

Other EDUCATIONAL MANAGEMENT programs will soon be available from Radio Shack.

Home Kindergarten

E. A. Tampasis   Decatur, Illinois

This is a program which parents might consider using with their pre-school kids.

This program was written for the TRS-80 Model I, Level II.

The program is simple and unique. It displays the alphabet and the numbers from 0-9 for a short time of 3-5 seconds, then the alphabet and numbers scroll up the screen. When the screen is blank, just press any key and that character will roll up the screen slowly. By DELETEing line 140, the alphabet and numbers will stay on the screen and you can input words or write sentences from the keyboard. By using a different CHR$ in line 140 you can get different effects. For instance, CHR$(95) will give you dashed lines across the screen between the times when you press keys on the keyboard.

```
5 REM WRITTEN BY E. A. TAMPASIS
10 CLS
20 PRINT CHR$(23)
30 REM ALPHABET
40 FOR A=65 TO 90
50   PRINT CHR$(A);
60   PRINT CHR$(193);
70 NEXT
80 REM NUMBERS
90 FOR B=0 TO 9
100   PRINT B;
110 NEXT
120 PRINT CHR$(23)
130 PRINT INKEY$;
140 PRINT CHR$(193);
150 GOTO 130
```

Peripherals (From Page 17)

There you have it. A compact piece of hardware . . . easy to set up and use . . . well supported by software.

Remember the limitations: one way control, carrier-current devices have limited range, they can interfere with carrier-current intercoms or similar devices, and are subject to the usual interference to the system by power brown-outs and the like.

All in all, the TRS-80 PLUG 'N POWER controller is a clever device. It should be lots of fun to use.

One final note: the control units are all designed for 110V appliances and lights. I can think of other applications. What about a lamp and photo-cell in one housing — voila! remote volume control. I have always thought it would be neat to have a module which instead of switching AC power, providing instead a multipole relay action for control of low voltage DC devices. We don't have any plans for modules of this type, but if there were enough interest . . . a guy can dream!

That's it this month. We are still at work on articles describing the functional and performance specifications of R.S. printers. 'Til next time . . . More (PLUG 'N) POWER to you!!

Model II Bugs, etc. (From Page 12)

Profile II (cont.)

Command I. Use the TRSDOS KILL command to kill the segments on the B disk which you do not want. The segment names are:

First Segment—***/KEY
Second Segment—***/DAT
Third Segment—***/DA2
Fourth Segment—***/DA3

5. Now COPY the ***/MAP file (*** stands for your Profile file name padded on the right with enough zeros to make eight characters) from diskette A onto diskette B which contains the BACKUP copy of the file with too many segments.
6. COPY the files from disk A onto disk B which have the same names as the files you killed off of disk B in step 4.
7. Disk B should now contain all of your data which was in the segments you wanted to keep, and the number of segments should have been reduced. Verify that the information is intact, and make a BACKUP of this disk.

SPELLING TEST

Peter S. Prentice

The following is a short program I wrote to help my eight year old son to learn his spelling words. The program was written on a Model I 32K Level II Disk system. The program is best used if the child will say the words at the beginning of the program, then proceed and try to spell them. It requires that he eventually memorize the words, but this is done naturally during the course of running the program a few times.

After being asked to enter his name, the program asks if new spelling words are to be entered. A "YES" answer will open a file to allow entry of new spelling words for the week and empty the file of the previous week's words. A "NO" answer will simply read previously stored words into the program. The correct number of words to be entered must then be input and finally the words themselves, one at a time. An opportunity is presented to correct any errors before the words are stored on disk.

As the child spells the words, used words are printed at the top of the screen so that he doesn't have to remember which ones he's used. The remaining words can be seen (if he gets lost) by pressing **ENTER** alone. This will give him a few seconds to look at the remaining words. This time can be adjusted by changing the loop in line 260.

The program will count the number of wrong spellings and the number of times he had to peek. At the end of the program these will be pointed out and the screen will suggest that he run the program again. At no time is the child put down for missing a word but rather is encouraged to try again.

Although my child still objects to having to memorize the words to get a perfect score, it has brought up his grades. I hope that you will find this a helpful program as I have.

Note: Extremely long words may cause problems on the video print out due to the use of a comma to space the words.

```
10 CLS: FOR X=1 TO 4: PRINT CHR$(23): NEXT: PRINT"SPELLING TEST":
   FOR X=1 TO 1000: NEXT: CLS
20 CLS: CLEAR 1000: INPUT"WHAT IS YOUR NAME";N$: CLS
30 INPUT"DO YOU HAVE TO ENTER NEW WORDS";J$: IF LEFT$(J$,1)="Y"
   THEN 40 ELSE IF LEFT$(J$,1)="N" THEN 310 ELSE PRINT"WAKE UP -
   YES OR NO?": GOTO 30
40 CLS: PRINT"HOW MANY SPELLING WORDS DO YOU HAVE "; N$;: INPUT A:
   DIM A$(A), A1$(A), B$(A), A(A)
50 PRINT"TYPE IN YOUR SPELLING WORDS ONE AT A TIME, " ;N$
60 FOR X=1 TO A: PRINT X;", ";: INPUT A$(X): NEXT
70 CLS: FOR X=1 TO A: PRINT X; ", "; A$(X),: NEXT
80 PRINT: PRINT"IF ONE OF THESE IS NOT RIGHT, THEN ENTER THE
   NUMBER, THEN"
90 PRINT"ENTER THE CORRECT SPELLING. ENTER <0> WHEN DONE."
100 INPUT X: IF X<0 OR X>A THEN 100 ELSE IF X=0 THEN 110 ELSE
    PRINT X; ", ";: INPUT A$(X): GOTO70
110 FOR X=1 TO A: A1$(X)=A$(X): NEXT: CLS: GOTO 280
120 FOR X=1 TO A: A$(X)=A1$(X): NEXT
130 CLS: FOR X=1 TO A: PRINT A$(X),: NEXT: PRINT: PRINT"NOW YOU
    SPELL THE WORDS, "; N$; ". "; "HIT ENTER IF YOU ARE LOST.":
    PRINT: INPUT"HIT ENTER TO BEGIN"; V: CLS: FOR Y=1 TO A
140 PRINT@512,Y; ". ";: INPUT B$(Y): CLS: IF B$(Y)="" THEN GOTO
    260
150 FOR X=1 TO A: IF B$(Y)=A$(X) THEN T=1: A(Y)=X
160 NEXT X
170 IF T=1 THEN PRINT@640,"VERY GOOD!!! TRY ANOTHER ONE, "; N$:
    PRINT@0,"";: FOR Z=1 TO Y: PRINT B$(Z),: NEXT Z
180 IF T=1 THEN T=0: A$(A(Y))="  ": NEXT Y: GOTO 200
190 IF T<>1 THEN PRINT@640,"ALMOST GOT IT. TRY AGAIN, "; N$;:
    B$(Y)="": M=M+1: PRINT@0,"";: FOR Z=1 TO Y: PRINT B$(Z),:
    NEXT Z: GOTO 140
200 CLS: PRINT CHR$(23); "THAT WAS WELL DONE, "; N$; "."
210 IF P<>0 THEN PRINT CHR$(23); "YOU HAD TO PEEK"; P; "TIMES."
220 IF M<>0 THEN PRINT CHR$(23); "YOU WERE WRONG"; M; "TIMES."
230 IF M<>0 OR P<>0 THEN PRINT CHR$(23); "YOU BETTER TRY ONE MORE
    TIME!!": PRINT CHR$(23); "HIT ENTER";: INPUT V: M=0: P=0: FOR
    X=1 TO A: B$(X)="": A(X)=0: NEXT: GOTO 120
240 PRINT CHR$(23); "DO YOU WANT TO RUN THROUGH THEM AGAIN?":
    INPUT H$: IF LEFT$(H$,1)="N" THEN END
250 GOTO 120
260 P=P+1: CLS: FOR Z=1 TO A: PRINT A$(Z),: NEXT: FOR Z=1 TO 2000:
    NEXT: CLS: PRINT@0,"";: FOR Z=1 TO Y: PRINT B$(Z),: NEXT Z:
    GOTO 140
270 STOP
280 OPEN"O",1,"WORDS"
290 PRINT#1,A; ",";
300 FOR X=1 TO A: PRINT#1,A$(X); ",";: NEXT: CLOSE: GOTO 130
310 OPEN"I",1,"WORDS"
320 INPUT#1,A: DIM A$(A), A1$(A), B$(A), A(A)
330 FOR X=1 TO A: INPUT#1,A$(X): NEXT: CLOSE: FOR X=1 TO A:A1$(X)=
    A$(X): NEXT: GOTO 130
```

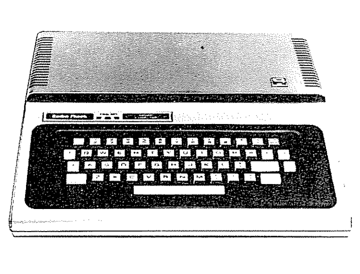
Half Duplex and Our Direct Connect Modem I

We have recently had several questions about our Direct Connect Modem (26-1172) and the fact that it does not have a Full Duplex/Half Duplex switch. The Direct Connect Modem performs in a full duplex mode. What this means is that it is able (when used with an RS-232c port) to transmit and receive information simultaneously.

Many time share systems require, or at least make provision for "half duplex" transmissions. According to Bell specifications, half duplex means that a terminal which is operating half duplex cannot transmit at the same time it is receiving. It also means that the received carrier must go away before the terminal can transmit its carrier and data. The host computer works in the same manner.

Most modem systems which use half duplex, use a modified type of half duplex operation. In these "half duplex" systems the characters sent by the terminal are not echoed back to the terminal by the host computer as they are in full duplex. In this use of "half duplex" it is the terminal's task to place a copy of the transmitted character on the screen. Hence, in these systems, half duplex means "there is no echo back of received characters." The requirement that the various carrier signals be switched on and off is not adhered to.

It is our experience that many of the half duplex systems which you are likely to use will operate using this modified system. If this is true, the 1172 Direct Connect Modem will work with no problems. If you should need to operate in a true half duplex system, the direct connect modem will not work without some software modifications to handle the extra requirements of true half duplex. For a true half duplex system, you may wish to use our Telephone Interface II (26-1171). We do wish to point out that the Model I TRS-80 can be used with the Direct Connect Modem in a half duplex mode by using the Model I's cassette port, our Cassette Comm software (26-1139) and the connecting cable (26-3009).



VIDEOTEX

Product Line Manager's News

Instant Information

That's right! Now you can have almost instant access to information of all types with Radio Shack's Videotex software.

Just think, from your own home you can now have the ability to view major newspapers and wire-services with up to the minute news stories as well as a host of other informational services.

Radio Shack has made agreements with CompuServe in Columbus, Ohio and Dow Jones of New York to provide you with the latest in up-to-date information access on a variety of subjects. Only Radio Shack has both services!

CompuServe has been bringing people and information together for more than a decade. As a major computer services company, they are applying the same technology to home use that they developed from serving industry and government. Their proprietary communications network extends to 35 metropolitan areas of the U.S. through more than 20,000 miles of high speed telephone lines and more than 90 minicomputers assuring that the information received is both error-free and complete. This same network provides you the ability of connecting to their system with only a local telephone call in more than 250 cities throughout the U.S. with more being added each month! CompuServe offers a world of information at your fingertips:

NEWS AND FAMILY INFORMATION: International, national, regional and local news, weather, sports, commentary — all up to the minute and indexed for easy selection of the topics that interest you. The information is available from international news services and from the nation's most prominent news publications. Also, family service information is available on subjects like food recipes, meal preparation, costs per serving, nutrition, personal health, gardening, home decorating, building, crafts, travel and money management.

An encyclopedia-like bank of general reference information on world events, history, geography and famous people is also available.

ELECTRONIC MAIL: Send and receive messages to other CompuServe subscribers or to groups of people across the street or across the country. Participate in a CB-like simulation with your own "handle" and talk with other information enthusiasts.

NATIONAL BULLETIN BOARD: Post messages, items for sale or describe an item on which you would like information. All items are categorized for easy reading and retrieval.

SECURITIES INFORMATION: CompuServe's MicroQuote provides trading statistics and descriptive information updated daily on more than 32,000 stocks, bonds and options. Historical prices and volumes extend back 8 years; dividend history back 12 years. Detailed information on volumes, dividends, earnings per share, ratings and shares outstanding is available from MicroQuote. This information includes securities traded on national exchanges and over-the-counter. Because of the detailed information available from MicroQuote, a nominal fee is charged above the regular CompuServe fee; the amount charged depends on the information requested.

PERSONAL FINANCE: CompuServe offers personal money management information for items such as mortgage loans, compound interest calculations, depreciation analysis, compound growth rates, tax tips and salary calculations.

GAMES: Play mind-challenging games such as Adventure, Hangman, MicroThello, Chess, Backgammon, Civil War Simulation, Craps, Maze, Roulet, Space War, Blackjack, Football, Golf, Fastermind and Lunar Lander. Many can be played with one or more players or with players in other cities.

MicroNET PERSONAL COMPUTING: CompuServe provides you with all the power of multiple main frame computers languages at your very fingertips. You can write programs in Extended BASIC, Fortran, APL, Pascal, BLIS10, SNOBOL, AID and even MACRO for machine language programming. The MicroNET Software Exchange allows you to purchase programs electronically and have them "down-loaded" onto your personal computer error-free and ready to run (this is not applicable to the VideoTex Terminal and requires disk storage for most other computers). Program documentation for MicroNET and the various programming languages is available through CompuServe at a nominal fee for each individual manual.

Has the Business and Financial explosion of information got you down? Do you need to be able to have accurate and up-to-date information particular to your business? Dow Jones information is also at your disposal!

The **Dow Jones Information Services** data base contains exclusive current and past news stories from the Dow Jones News Service, The Wall Street Journal as well as Barron's National Business and Financial Weekly (published by Dow Jones Co., Inc.).

You have access to current price quotations (delayed the mandatory 15 minutes) on more than 6000 stocks and other securities listed on four major U.S. stock exchanges (New York, American, Midwest and Pacific) plus the national over-the-counter market. Dow Jones also offers access to such information as revenues, earnings, dividends, price-earnings ratios, and stock price performance relative to market indicators on 3200 companies and 180 industries through an agreement with Media General, Inc. Precise business information you need to know from sources you can rely on!

You get a capability you've never had before — awareness within 90 seconds of fast-breaking business and financial news. The ability, instantly, to select only the exact information you choose to know more about. Not only the up-to-the-minute news but also the news within the past 90 days. It's simplicity itself to use this data bank. Whether the news happened 90 seconds ago or 90 days ago, all you need to do is press a few keys. In seconds you read a "capsule summary" of just what you wanted to know about your industry or your competitors . . . about labor negotiations, product introductions, pricing moves, overseas monetary developments or domestic energy events. Then, for more detailed information, you can request the full text of the story you want to see.

Dow Jones Information Services could give you the competitive edge you seek and help you serve your clients or customers better.

Only Radio Shack can offer you access to both these worlds of information in one package for one low price. Come visit a store, see the systems in action and we know that you'll agree! Radio Shack offers Videotex software for the following computers:

Model I

Requires minimum of Level II with 4K RAM and communications capability (Expansion Interface with RS-232C and a modem for full duplex operation or the Direct Connect Modem I with software package 26-1139 for half-duplex operation through the cassette interface connector). Order 260-2220 for $29.95.

(Continued on Page 23)

Videotex (From Page 22)

Model II

Requires minimum of 32K RAM with modem. Order 260-2221 for $29.95.

Model III

Requires minimum of Model III BASIC, 16K RAM, RS-232C serial interface and a modem. Order 260-2220 for $29.95.

Color Computer

Requires minimum of 4K RAM, Standard BASIC and a modem. Order 260-2226 for $29.95.

Apple II

Requires minimum of 16K Apple II with communications capability and a modem. Order 260-2223 for $29.95.

"Dumb" Terminal Package

This can be used with any computer or communications terminal that can be linked to a modem. It contains no software, only the CompuServe and Dow Jones packages. Order 260-2224 for $19.95.

Political Computing

Connecticut College New London

The Department of Government at Connecticut College New London has started a new journal for students active in politics. At this point the circulation is limited to New England, but it is expected to grow. In Politics is meant to be a clearing house for student political internships and activities in campaigning and lobbying.

The fall issue will focus on campaigning, and it occurred to the staff at Connecticut College New London that the small personal computer is or will be playing a major role in planning campaign strategies for local elections. In Politics would like to feature any interesting campaigns in which students have used the TRS-80. The journal is also interested in featuring programs produced by students, professors, or computer professionals for planning campaign strategies on the TRS-80.

Send information to:
In Politics
Minor Myers, Jr. Chairman
Dept. of Government
Connecticut College New London
New London, Connecticut 06320

Reduce Fractions

Sheldon Beasley Atlanta, Georgia

As a new subscriber to TRS-80 News, I feel that my obligation would not be complete unless I submit a program or programs to the newsletter. I have included a program that reduces fractions:

```
2 REM SHELDON BEASLEY
4 REM ATLANTA GEORGIA
10 CLS
20 PRINT"INPUT FRACTION (A/B)"
30 INPUT"AS A,B";N,D
40 B=N: C=D
50 FOR X=N TO 1 STEP -1
60   IF INT(N/X)-N/X=0 THEN 90
70   NEXT X
80 GOTO 130
90 IF INT(D/X)-D/X=0 THEN 110
100 GOTO 70
110 IF X=B THEN PRINT "IN LOWEST TERMS": GOTO 130
120 PRINT N/X; "/"; D/X;"IS THE REDUCED FORM OF"; N; "/"; D
130 PRINT"FOR ANOTHER FRACTION TYPE"
140 INPUT"1, ELSE TYPE 0;Q
150 ON Q+1 GOTO 160, 10
160 END
```

National Crisis Center For the Deaf

We would like to provide some information which we hope may benefit some of our readers.

There are 14 million hearing impaired people in the United States. Many of these people cannot use the telephone because they are deaf. When a deaf person needs help in an emergency he/she cannot use the telephone in a "normal" way to call for assistance. Through the use of communication terminals and regular telephones these individuals are able to communicate effectively with facilities which are equipped with similar terminals.

The National Crisis Center for the Deaf is a program from the University of Virginia Medical Center in Charlottesville, Virginia. The purpose of the program is to provide a deaf person with a TTY/TDD 24-hour contact with emergency services.

The National Crisis Center for the Deaf toll-free TTY number across the United States is: 1-800-446-9876. (In Virginia the toll-free number is 1-800-552-3723.) Someone is at the Center to answer TTY calls 24 hours a day.

The people who answer TTY/TTD calls have medical training as well as special training in communicating with deaf people.

Three doctors from the university are the medical directors of the program. Richard F. Edlich, M.D., Ph.D. is the Director of Emergency Medical Services. Daniel A. Spyker, Ph.D., M.D. is the Co-Director of the Blue Ridge Poison Control Center. Wilford Spradlin, M.D. is the Chairman of the Department of Psychiatry.

When a deaf person calls the toll-free number he or she can have help with these different types of emergencies:

— sudden illness or injury: ambulance/rescue squad
— personal crisis: counselor/someone to talk to
— poisoning: poison help
— police: police help
— fire: fire help

When a deaf person calls the National Crisis Center for the Deaf with a TTY/TTD:

1. The phone rings in the Crisis Center at the University of Virginia Medical Center in Charlottesville, Virginia.
2. The phone is answered by an NCCD staff member.
3. The deaf caller and the staff member discuss the problem with help from a computer. The computer and the staff member ask the deaf caller special questions which help the doctors find out more information about the deaf person's problem.
4. Then the NCCD staff member talks with the doctor about the deaf person's problem. If the problem is a poisoning or if the individual has a personal crisis the NCCD staff member tells the deaf caller what to do. If the caller needs a rescue squad, police department or fire department the NCCD staff member calls them by voice phone.
5. The rescue squad, police department, or fire department send help to the deaf person.
6. Later an NCCD staff member calls the deaf person to find out what happened. The NCCD staff member also finds out if the deaf person needs more help.

Radio Shack TRS-80 communications software is designed for communication using ASCII, not TTY/TTD (baudot) format. Ms. Mary Compton, Manager of the Center, stated that ASCII input/output will be available in the "near future." Ms. Compton also indicated that the Center is seeking information which might make the service more useful to deaf individuals. Her non-TTY telephone number is 1-804-924-1847. The address is Box 484, University of Virginia Medical School, Charlottesville, VA 22908.

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

Line Printer III/V Ribbons and Printheads

The ribbon life expectancy for the Line Printer III/V ribbons (26-1414) is two million characters. Please be aware that at 100% duty (constant, non-stop 132 characters per line printing) this is only about five (5) hours of printer time. Two million characters is approximately 418 full pages of print, with each page containing 60 lines per page and 80 characters in each line.

Ribbon life can be shortened if the print head is not cleaned on a regular basis. We strongly recommend that you get and use the special head cleaning kit (700-3010 Suggested Retail Price $1.50) which is available through your local Radio Shack. This kit was designed for use with the Line Printers III and V, and should help protect your printhead as well as allow you to get maximum use from your ribbons.

Printer ribbons and printheads can affect each other. If you use one of these ribbons beyond two million characters, the ribbon may begin to fray. If the ribbon begins to fray, the ribbon cartridge may jam, and if the ribbon cartridge jams, the printhead could be damaged. If the printhead should be damaged, or if the printhead is dirty, this could cause pre-mature fraying of the ribbon, which can cause . . . etc.

If you keep the printhead on your Line Printer III or V cleaned regularly, and change ribbons every two million characters, you should get many hours of trouble free service from either of these printers.

Notes on Previous Newsletters

May, 1981

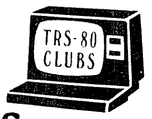
In the article on Printer Codes, we mentioned that the Model III Printer Driver uses a look-up table to determine the exact code to send to the printer for characters with decimal values from 32 to 127. We should have stated, that except for decimal 96, no changes are made. Decimal code 96, (grave accent) is translated to decimal 64 (@) before being output to the printer. What this means is that if you try to send a grave accent (96) to the printer, what will be printed is an @ symbol (64).

Reading Agreement

(From Page 1)

The planned availability date for the new reading series is September 1981.

Radio Shack's Education Division is engaged in an extensive courseware development effort to support the use of the TRS-80 microcomputer in schools. According to C. A. Phillips, Radio Shack senior vice president for Special Markets, "The Philadelphia Reading Program will fill an important niche in our courseware product line. We believe the availability of effective, high quality courseware is essential if we are to continue to be successful in the education market."



More Computer Clubs

Mid-South Microcomputer Users Group
Rt. 11, Box 64
Bowling Green, KY 42101
502-781-5381
CBBS 502-781-3123

TRS-80 Users Group of Atlanta, Ltd.
Bob Green, Pres.
1315 Rustic Ridge Drive, N.E.
Atlanta, Georgia 30319
404-451-9813

COMPUTER SERVICES ADDRESS AND PHONE NUMBERS
8AM to 7PM Central Time

Computer Services
900 Two Tandy Center
Fort Worth, Texas 76102

Model I/III Business Software	
Outside Texas	1-800-433-5641
In Texas	1-800-772-5973
Model II Business Software	
Outside Texas	1-800-433-5640
In Texas	1-800-772-5972
All Other Calls	
Outside Texas	1-800-433-1679
In Texas	1-800-772-5914
Switchboard—	1-817-390-3583

TRS-80 Microcomputer News is published monthly by Radio Shack, a division of Tandy Corporation. A single one year subscription is available free to purchasers of new TRS-80 Microcomputer systems with addresses in the United States, Puerto Rico, and APO or FPO addresses. Subscriptions to other addresses are not available. The subscription rate for renewals and other interested persons is twelve dollars ($12.00) per year, check or money order in U.S. funds. All correspondence related to subscriptions should be sent to: Microcomputer News, P.O. Box 2910, Fort Worth, Texas 76101.

Back issues of Microcomputer News are not available.

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