TRSTimes

Volume 8. No. 2 - Mar/Apr 1995 - $4.00

my favorite computer

TRSTimes magazine

Volume 8. No. 2 - Mar/Apr 1995 - $4.00

PUBLISHER-EDITOR
Lance Wolstrup
CONTRIBUTING EDITORS
Roy T. Beck
Dr. Allen Jacobs
TECHNICAL ASSISTANCE
San Gabriel Tandy Users Group
Valley TRS-80 Users Group
Valley Hackers' TRS-80 Users Group

TRSTimes is published bi-monthly by TRSTimes Publications. 5721 Topanga Canyon Blvd., Suite 4. Woodland Hills, CA 91367. U.S.A. (818) 716-7154.

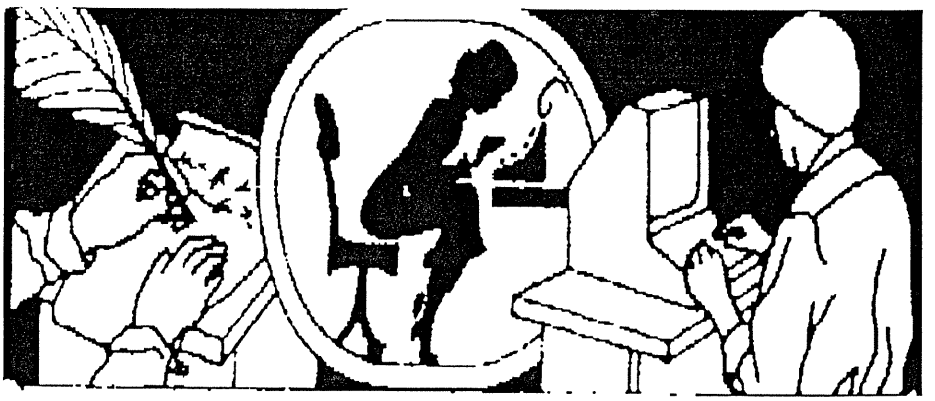
Publication months are January, March, May, July, September and November.

1995 subscription rates (6 issues):
UNITED STATES $21.00
CANADA $22.00 (U.S.)

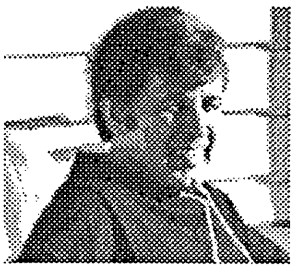
EUROPE, CENTRAL & SOUTH AMERICA:
$26.00 for surface mail or $34.00 for air mail. (U.S. currency only)

ASIA, AUSTRALIA & NEW ZEALAND:
$28.00 for surface mail or $36.00 for air mail. (U.S. currency only)

Article submissions from our readers are welcomed and encouraged. Anything pertaining to the TRS-80 will be evaluated for possible publication. Please send hardcopy and, if at all possible a disk with the material saved in ASCII format. Any disk format is acceptable, but please note on label which format is used.



LITTLE ORPHAN EIGHTY



Who said that it never rains in Southern California? The last couple of winters have been wet, especially this year. It seemed like it rained for weeks on end, dark, dreary and cold, and I now remember why I came here in the first place.

The rains also caused me to bang up my brand-new car. A Dodge Shadow decided run a stop-sign and cross a main thoroughfare against traffic. Unfortunately, I was on the main thoroughfare and I was the traffic, and even though I slammed on my brakes when it darted out in front of me, because of the rain, I skidded into it. The front of my car collided with the Dodge, hitting the passenger side.

Steven, strapped in his seat belt, turns to me and says: "Way to go, Dad!" He was fine, as was I. But, believe me, I was ticked. I jumped out of the car, ran over to the other car and ripped open the door. Boy, was I gonna yell at this guy.

But then I saw the driver, a little old lady, cowering in the seat, scared to death, so my yelling consisted of "Are you OK, Ma'am?"

As it turned out, she was fine also, not even a scratch, but her Dodge was totalled, and my car suffered $3000 worth of body damage. The good news is that it runs fine and the body work will be done as soon as the insurance company releases the check - but it is annoying to drive a new car that is dented. Oh, well!!

Because of my fling in the hospital in November, our earthquake repair was postponed until — yes, you guessed it — January. On the very first rainy day (did I expect anything less?) I moved the family and TRSTimes to a furnished apartment in a great complex just a few blocks from our condominium. This place has 4 swimming, pools, many tennis courts, volleyball and basketball courts, as well as a ping-pong table. Boy, we were going to get a nice workout. It rained darn near every day we were there. Oh, well!!

We were going to bring only the bare necessities, so I brought a weeks worth of clothes — 7 pairs of slacks, 7 shirts, 7 pairs of socks, etc. My clothes took up about 1/8th of the 2 large walk-in closets in our bedroom. My lovely wife managed to bring enough clothes to fill up the other 7/8th — and a bit of overflow. "Never know when the Queen might come to dinner!" she explained in her best British accent. By the 3rd week there I realized that some of the furniture was ours, having been moved in within the last couple of days. I also recognized the microwave oven, many of her favorite cooking utensils, as well as her best china. I would have been content to eat off of paper plates. Ah, the joys of marriage. Oh, well!!

Finally, after 6 weeks, the main repairs on our condo were completed, so we moved back (yes, it rained that day). Moving has taken a lot of time and we are still opening boxes and putting things away. Isn't it amazing how much junk you can accumulate? But it is so nice to be back home — with no cracks in the walls and floors. Now we just bide our time until the next one, hoping that it is not the BIG ONE.

A bit of errata from the last issue - 8.1. It seems that my brain is still foggy. For some reason, I insisted on spelling the word CAPITAL as CAPITOL. This occured in my article, ODDS & ENDS FOR MODEL 4 beginning on page 4 and it requires the program listing to be changed as follows:

```
110 V=1:A$="S T A T E S  & C A P I T A L S":
GOSUB 21
200 V=9:A$=CHR$(31):GOSUB 20:H=20:
A$="1. What is the capital of...":GOSUB 23
210 V=11:A$="2. What state has the capital city...":
GOSUB 23
```

Sorry about that!!

Now for the good news. I spent a little time 'surfing the Internet' last month, and I found a site that has TRS-80 programs available free for the downloading. The address is:

musie.phlab.missouri.edu

in the /pub/trs directory. There is Model I/III and 4 stuff, as well as material for CoCo and Model 100. Check it out.

Finally, many thanks to Danny Myers, Daniel Franco, David Meny, Kelly Bates, Dave Garr, Wayne Westmoreland, Henry Herrdegen, Frank Slinkman and Roy Beck for their contributions to this issue.

And now..... *Welcome to TRSTimes 8.2*

BEAT THE GAME

By Daniel Myers

MISSION IMPOSSIBLE

The Scott Adams Adventures

You've always wanted to be a secret agent, right? Right! Well, now you have the chance...and there isn't much time left, so it's best to get started. Play the tape recorder, and listen carefully. Then get the recorder, and go west and south to the grey room. Sit in the chair, then push the red button first and the white one next. Now, get up and look. There's a picture of you here. Pick it up; it's important!

Now comes the hardest part of the game. You have to kill a little time until the saboteur croaks, then look in various rooms until you find him. There's nothing you can do about that, so for a while you're just going to be on your own. Think of it as a good opportunity to practice map-making skills.

Once you've found the dead saboteur, frisk him ("Frisk Saboteur"). Get his picture, then get him (yes, you're going to be lugging a dead body around for a while). Now, make your way to the white door. There's no way I can give you directions for this, since there's no telling where you found the saboteur (that's why you should have made a map!). When you get there, "Show Picture." The program will always assume you're showing the right one, if you have it. The door will open and you will be inside. Now, drop the saboteur (whew! he was heavy!), and break the window with the tape recorder.

The TV camera will come on and scan the room. Show Picture" again...this time, it's the picture of the saboteur. The camera will scan the picture, then the saboteur, and turn itself off. Now you can go out the window and pick up the yellow key on the window ledge. Go back inside, and drop the two pictures you're carrying, since you won't need them anymore. Push the white button and the door will open, allowing you to leave. Now, go South, down, and South again to the grey room.

Sit in the chair and unlock the yellow button. Press the yellow button first, then the white one. Stand up and get the picture, then head North, down, and North to the maintenance room. Get the mop and shake it. Drop the mop and get the key. Also pick up the wirecutters. Now, head back East to the corridor and push the yellow button. Drop the picture, then go South, up, and South to the grey room again.

Sit in the chair and unlock the blue button. In this order: blue, red, white, press each button and get out of the chair. Drop the key, get the picture, and go North. Then turn West and pick up the pail. Now, go East, North, and North to the blue door, and show the picture. Once inside, drop the key and go West to the store room. Get the radiation suit and wear it, then fill the pail with water. Return East and open the control room door. It won't be easy, but you can do it! After that, go East into the break room and leave the pail there.

Now, go West into the control room, then down to where the bomb is. Cut the wires and get the bomb. Then, go up and East to the break room. Get the pail, then pour water. WHEW!!! The bomb is deactivated! You have successfully completed your mission!! (Now you can take it easy for a while...let's just hope you don't start glowing in the dark!)

GHOST TOWN

The Scott Adams Adventures

Okay, pardner, it's time for a little trip to a gen-u-ine ghost town! But don't worry, them's **friendly** ghosts. So, mount up and let's get going!

Start things off by going into the Barbershop, where you'll find a Stetson hat. Shake the hat and drop it. Get the key that fell out, then leave (you don't need the hat for anything). Drop the key in the street.

Go West along the street and you will come to the Saloon and the Dry-Goods Store. First go to the

Saloon and get the bell, then go to the Dry-goods Store and pick up the matches, and shovel. Back in the street, go West once again, which brings you to the Telegraph Office and the Hotel.

Go into the Hotel, then East into the empty room. Drop the bell and return to the street. Now enter the Telegraph Office, and move the safe. Connect the two loose wires so that the telegraph key now works. You'll be needing it later. Leave the Telegraph office, then go West until you come to the fork in the road.

From the fork go South to the edge of the ravine. Burn the sagebrush, then enter the ravine. Here you will find charcoal (from the burnt brush) and the entrance to a mine. Go into the mine. Although it's dark and you can't see it, there is a silver bullet here. Get the bullet, then go down.

You can move in the dark safely so long as you always move in the right direction.

Now get the candle and light it. Ah, you can see again! Go South, and dig roof. You have found your second treasure, a gold nugget. Get that, and go back North and Up out of the mine. Remember to pick up the charcoal before leaving the ravine!

Now go to the fork in the road, and drop off the shovel and the charcoal. From there, go East back into town until you come to the stable. Enter the stable, and then the stall. Get the horseshoe, then make your way back out to the street.

Now head along East to the Dry-Goods store. Drop the candle outside, treasures inside, then go out and East again to the Jail. Pick up the key, then use horseshoe. It's magnetic and will open the door to the Jail. Enter the jail, and unlock the inner door with the key. Drop the key, and go through the door into the cell. Pick up the hammer, then leave the Jail, making sure you also take the derringer with you. Drop the derringer in the street.

It's time to play blacksmith, so go all the way West to the stable, and then enter the stall. This time, mount the horse, and you will into the manure pile outside the stall. Get up (phew!) out of the manure pile, and re-enter the stall. There will now be a hole in the wall leading to a store room. Go through the hole and get the keg of nail3s. Back in the stall, empty the keg and drop it. Now get the nails, and shoe the horse. Drop the hammer and mount the horse.

Say giddyap (the magic word!), and the horse will take off. Eventually, you will be thrown, and that's the last you'll ever see of Old Paint. Brush yourself off, and enter the teepee in the hidden canyon. Pick up the two treasures there, then go back outside.

There doesn't seem to be any way out, but have no fear! Beat the tom-tom, and the ghost of Geronimo will appear. Say How, and ZAP! guess where you are? Right, you're back in the manure pile again (hehehe). Get out of that, then head along to the Dry-Goods store and drop off the treasures (don't forget to drop the spurs, too!).

Somewhere along the line here you may have heard mysterious ghostly sounds and or voices. The sound of the bell indicates that a ghostly piano player is now visible in the Saloon, and the voice gives you a clue as to what to do about him. If you are near the Saloon when you hear the bell, go inside, and applaud the ghost.

He (it?) will stand up, take a bow, and vanish. The piano, however, will remain behind, as a solid object. If you open the piano, a map will fall out. This map tells you to "dig roof", but since you've already done that, you don't need the map, so you can just leave it there.

Once you've dropped off the Indian treasures and (possibly) applauded the ghost, return West to the fork in the road. Get the shovel and charcoal, then go North to the field. Dig here, and you will find some yellow powder (it's sulphur). Get the powder, then go to the manure pile in the stall.

Holding your nose, dig around in the manure, and you will uncover some white crystals. Get those, then head into the stall. Mix the stuff you're carrying around, and you will make some gunpowder. Fill the keg with that, then get the keg and go to the Telegraph Office.

Drop the keg in the office. Under no circumstances should you touch the telegraph key, or BOOOOM! (time to restore the game!). By this time, it's probably getting dark outside. Don't worry, you'll be able to make it to the hotel before sunset.

Just leave the Telegraph Office, enter the Hotel, and go East to the room where you dropped the bell. Ring the bell, and, like magic, a bed appears! Drop the bell, get into bed, and have a good night's sleep.

When morning arrives, get up, then move the bed, revealing a roll of tape. Get the tape and leave the room. On your way out, go to the counter and get the cashbox. Now return to the Saloon, tape the

mirror, and break it, thus revealing a hidden office. Drop the tape, go through the hole into the office, and get the Go board.

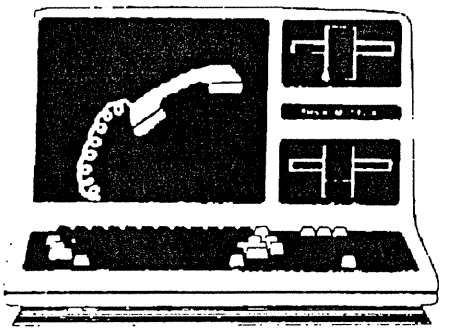
Now it's time for another trip to the Dry-Goods Store. Drop off the cashbox and the Go board. Now, pass Go, and collect $200 (tricky, huh?). After that, leave the store and head West to the fork, then South to the ravine. This time, jump across the ravine.

You are now in the mountains. Go West along the trail to the line shack. Enter the shack, and tap the telegraph key. Boom! The gunpowder in the keg just went off! Now, look at the floor and you'll notice a loose plank. Get the plank, drop it, then go down the hole into the root cellar. Collect the pelts, then go back up and make your way across the ravine and into town. As you pass where the Telegraph Office used to be, you'll see a smoking open safe. Look inside, and pick up the gold dust.

Continue East and pick up the derringer, then keep going East until you come to Boot Hill. Shoot the rattlesnake with the derringer (it's a water pistol!), then dig a grave and (gulp!) go into it. Here you find a coin and a purple worm. If you want to indulge in some gratuitous violence, you can kill the worm. In any case, drop the shovel and get the coin.

Now climb back out, and make another trip to the Dry-Goods store. Drop all the treasures, then go back out into the street. Get the candle, then wait for sunset. Once it's dark, go into the Saloon. A ghostly square dance is in progress (that's what the fiddle strings are all about). Still in the dark, do a little dancing, and you will win a prize.

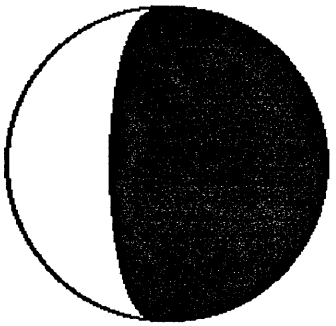
Now you can light the candle. The dancers will vanish, and you can now make your final trip to the Dry-Goods store. Drop the cup you just won, and say "Score". All right! You did it, you collected all thirteen ghost treasures! After all that, why not take a vacation? I know this little deserted island that would be just perfect.....



PHASES OF THE MOON

TRS-80 Model 4 - Basic

by Daniel P. Franco



This program calculates the phase of the moon for a given year and month. The user inputs the year, the month, and the number of consecutive months data are required for. Output includes Ephemeris Time of each phase beginning with the new moon.

```
10 '* moonph/bas
20 '* for TRS-80 Model 4
30 '* by Daniel P. Franco
40 '*
50 '*
60 '*
70 '*
80 '* input section
90 '*
100 CLS
110 DEFDBL A-Z
120 PRINT "Enter Year: ";:INPUT YEAR
130 LEAP=YEAR MOD 4
'if leap = 0 then year is a leap year
140 PRINT "Enter Month: ";:INPUT MONTH
150 PRINT "Output For How Many Months: ";:
INPUT COUNT
160 IF LEAP <> 0 THEN 200 ELSE 330
170 '*
180 '* calculations for decimal years
190 '*
200 IF MONTH = 1 THEN YD = .0424375935815675#
210 IF MONTH = 2 THEN YD = .1232059168497121#
220 IF MONTH = 3 THEN YD = .2039742401178567#
230 IF MONTH = 4 THEN YD = .2874804726493284#
240 IF MONTH = 5 THEN YD = .3709867051808#
250 IF MONTH = 6 THEN YD = .4544929377122716#
260 IF MONTH = 7 THEN YD = .537999170243743#
270 IF MONTH = 8 THEN YD = .6228743574068779#
280 IF MONTH = 9 THEN YD = .7063805899383497#
290 IF MONTH = 10 THEN YD = .7898868224698213#
300 IF MONTH = 11 THEN YD = .8733930550012927#
310 IF MONTH = 12 THEN YD = .9568992875327647#
320 GOTO 460
330 IF LEAP = 0 GOTO 340
340 IF MONTH = 1 THEN YD = .0424375935815675#
350 IF MONTH = 2 THEN YD = .1245748714813756#
360 IF MONTH = 3 THEN YD = .2053431947495202#
370 IF MONTH = 4 THEN YD = .2888494272809917#
380 IF MONTH = 5 THEN YD = .3723556598124635#
390 IF MONTH = 6 THEN YD = .4558618923439348#
400 IF MONTH = 7 THEN YD = .5393681248754063#
410 IF MONTH = 8 THEN YD = .6242433120385416#
420 IF MONTH = 9 THEN YD = .7077495445700128#
430 IF MONTH = 10 THEN YD = .7912557771014848#
440 IF MONTH = 11 THEN YD = .874762009632956#
450 IF MONTH = 12 THEN YD = .958268242164428#
460 K = ((YEAR+YD) - 1900) * 12.3685
470 K = CINT(K)
480 COUNT = K + COUNT
490 T = K/1236.85
500 T2 = T ^ 2
510 T3 = T ^ 3
520 PI=3.141592653589796#
530 R=PI/180
540 '*
550 '* sun mean anomaly
560 '*
570 SMA = 359.2242# + (29.10535608000003# * K)-
(.0000333*T2)-(3.47E-06*T3)
580 IF SMA > 360 THEN SMA=SMA/360:
SMA=SMA-FIX(SMA):SMA=SMA*360
590 '*
600 '* moon mean anomaly
610 '*
620 MMA = 306.0253#+(385.81691806#*K)+
(.0107306*T2)+(1.236E-05*T3)
630 IF MMA > 360 THEN MMA=MMA/360:
MMA=MMA-FIX(MMA):MMA=MMA*360
640 '*
650 '* moon's argument of lattitude
660 '*
670 F = 21.2964+(390.67050646#*K)-(.0016528*T2)-
(2.39E-06*T3)
680 IF F > 360 THEN F=F/360:F=F-FIX(F):F=F*360
690 '*
700 '* mean phase of the moon
710 '*
720 JD=2415020.75933#+(29.53058868000003#*K) +
(.0001178*T2)-(1.55E-07*T3)+(.00033*SIN((R*166.56)+
(R*132.87)*T)-((R*.009173*T2)))
730 SMA=SMA*R
740 MMA=MMA*R
750 F=F*R
760 '*
770 '* true phase corrections for new and full moon
780 '*
790 IF K-FIX(K)=0 OR K-FIX(K) =.5 OR K-FIX(K)=-.5
THEN 800 ELSE 970
800 JD=JD+((.1734-.000393*T)*SIN(SMA))
810 JD=JD+(.0021*SIN(2*SMA))
820 JD=JD-(.4068*SIN(MMA))
```

```
830 JD=JD+(.0161*SIN(2*MMA))
840 JD=JD-(.0004*SIN(3*MMA))
850 JD=JD+(.0104*SIN(2*F))
860 JD=JD-(.0051*SIN(SMA+MMA))
870 JD=JD-(.0074*SIN(SMA-MMA))
880 JD=JD+(.0004*SIN((2*F)+SMA))
890 JD=JD-(.0004*SIN((2*F)-SMA))
900 JD=JD-(.0006000001#*SIN((2*F)+MMA))
910 JD=JD+(.001*SIN((2*F)-MMA))
920 JD=JD+.0005*SIN(SMA+(2*MMA))
930 GOTO 1290
940 '*
950 '* true phase corrections for first and last quarter
960 '*
970 JD=JD+(.1721-.0004*T)*SIN(SMA)
980 JD=JD+.0021*SIN(2*SMA)
990 JD=JD-.628*SIN(MMA)
1000 JD=JD+.0089*SIN(2*MMA)
1010 JD=JD-.0004*SIN(3*MMA)
1020 JD=JD+.0079*SIN(2*F)
1030 JD=JD-.0119*SIN(SMA+MMA)
1040 JD=JD-.0047*SIN(SMA-MMA)
1050 JD=JD+.0003*SIN(2*F+SMA)
1060 JD=JD-.0004*SIN(2*F-SMA)
1070 JD=JD-.0006000001#*SIN(2*F+MMA)
1080 JD=JD+.0021*SIN(2*F-MMA)
1090 JD=JD+.0003*SIN(SMA+2*MMA)
1100 JD=JD+.0004*SIN(SMA-2*MMA)
1110 JD=JD-.0003*SIN(2*SMA-MMA)
1120 '*
1130 '* additional first quarter corrections
1140 '*
1150 IF K => 0 AND K-FIX(K) = .25 THEN 1170
ELSE 1160
1160 IF K < 0 AND K-FIX(K)=-.75 THEN 1170
ELSE 1220
1170 JD=JD+.0028-.0004*COS(SMA)+.0003*COS(MMA)
1180 GOTO 1290
1190 '*
1200 '* additional last quarter correction
1210 '*
1220 IF K => 0 AND K-FIX(K) = .75 THEN 1240
ELSE 1230
1230 IF K < 0 AND K-FIX(K) =-.25 THEN 1240
ELSE 1290
1240 JD=JD-.0028+.0004*COS(SMA)-.0003*COS(MMA)
1250 GOTO 1290
1260 '*
1270 '* calendar date calculation
1280 '*
1290 JD=JD+.5
1300 Z=INT(JD)
1310 FRAC=JD-FIX(JD)
1320 IF Z < 2.29916E+06 THEN A=Z
1330 IF Z => 2.29916E+06 THEN
ALPHA=INT((Z-1867216.25#)/36524.25#)
1340 IF Z => 2.29916E+06 THEN
A=Z+1+ALPHA-INT(ALPHA/4)
1350 B=A+1524
1360 C=INT((B-122.1)/365.25)
1370 D=INT(365.25*C)
1380 E=INT((B-D)/30.6001)
1390 DOM=B-D-INT(30.6001*E)+FRAC
1400 IF E<13.5 THEN M=E-1
1410 IF E>13.5 THEN M=E-13
1420 IF M>2.5 THEN Y=C-4716
1430 IF M<2.5 THEN Y=C-4715
1440 DAYINT=INT(DOM)
1450 DAYFRAC=DOM-FIX(DOM)
1460 TOTSEC=DAYFRAC*86400!
1470 TOTHOURS=(TOTSEC/60)/60
1480 HOUR =INT(TOTHOURS)
1490 MINLEFT=TOTHOURS-FIX(TOTHOURS)
1500 TOTMIN=(MINLEFT*60)
1510 MIN=INT(TOTMIN)
1520 SECLEFT=TOTMIN-FIX(TOTMIN)
1530 SEC=(SECLEFT*60)
1540 IF K => 0 AND K-FIX(K)=0 THEN PHASE$="NEW
MOON"
1550 IF K=> 0 AND K-FIX(K)=.25 THEN
PHASE$="FIRST QUARTER"
1560 IF K=> 0 AND K-FIX(K)=.5 THEN
PHASE$="FULL MOON"
1570 IF K=> 0 AND K-FIX(K)=.75 THEN
PHASE$="LAST QUARTER"
1580 IF K < 0 AND K-FIX(K) = 0 THEN PHASE$="NEW
MOON"
1590 IF K < 0 AND K-FIX(K) = -.75 THEN
PHASE$="FIRST QUARTER"
1600 IF K < 0 AND K-FIX(K) = -.5 THEN
PHASE$="FULL MOON"
1610 IF K < 0 AND K-FIX(K) = -.25 THEN
PHASE$="LAST QUARTER"
1620 PRINT USING "#### ## ##    ##\  \ ##\ \
##.##\ \   \
\";Y,M,DAYINT,HOUR,"Hours",MIN,"Min.",SEC,"Sec.",
PHASE$
1630 K=K+.25
1640 IF K = COUNT GOTO 1660
1650 GOTO 490
1660 END
```

Escape From Soviet Science And Detention Base (SSADB)

Adventure Game for TRS-80 Model 4

by David Meny

Escape From SSADB is a text adventure in the tradition of great adventures like Zork by Infocom and the adventures by Scott Adams. This pro-gram isn't as complicated as Infocom's full parser, but uses only two word commands like GET KEY or HIT COMPUTER.

In Escape From SSADB, you are a government agent captured by the Russians and taken to a secret base hidden somewhere in the United States. While you are there you learn of the Soviet's plans to fire nuclear missiles from inside the US and make it look like an inside job, thus making the Soviets safe from counter attack. You must save the US from nuclear attack by getting out of your cell, finding proof about the missiles and escape the Top-Secret base. If you succeed you will be a national hero; if you fail the US is doomed.

Listed below are a few simple commands used by the program. Remember, if you use a command like THROW ROCK the program will follow by asking you what you want to throw it at.

GET (object)	to take an object
DROP (object)	to put down an object
THROW (object)	to throw an object at something
SCORE	to show your current score
QUIT (or Q)	to leave the game
INVENTORY (or I)	to show what you are carrying
GO NORTH (or N)	to go to the north (The same for all other compass directions (N,S,E,W,U and D))

Now take your trusty brick and newspaper (you'll see in the game) in hand and venture forth in to the realm of spies and agents.

```
100 DIM A(17,10),P(5),O1$(11),O2$(11),M$(27),T(5),
ROOM$(17),IN$(11),TR(11)
110 REM **                              **
120 REM **   ESCAPE FROM              **
130 REM **     S.S.A.D.B.             **
140 REM **        BY                  **
150 REM **   DAVID MENY               **
160 REM **                            **
170 REM **                            **
180 POKE &HB94,PEEK(&HB94) OR 2
190 GOSUB 4230:REM INITIALIZE
200 REM
210 REM
220 REM
230 PRINT:GOSUB 3240:REM ROOM DESCRIPTION
240 GOSUB 1900:REM OBJECTS IN ROOM?
250 REM ** INPUT HANDLING **
260 KW=0
270 IF SIREN=1 AND SET=0 THEN MV=1:SET=1
280 IF SET=1 THEN MV=MV+1:IF MV=3 AND RO=6
AND BOX=0 THEN PRINT:
PRINT "A worker comes running in and glances around.
He sees you and calls the guards":
PRINT "who take you away and torture you.":
GOTO 3140
290 IF MV=3 AND RO=6 AND BOX=1 THEN PRINT:
PRINT "A worker comes in and glances around. He
glances around and not noticing":
PRINT "anything, he heads to the balcony.":
TR(11)=7
300 IF MV=3 THEN TR(11)=7
310 PRINT:INPUT ">",A$
320 IF A$="" THEN PRINT:PRINT "What?":
GOTO 270
330 AS=ASC(A$)
340 IF AS<65 OR AS>90 THEN PRINT:
PRINT "Capital letters please.":GOTO 270
350 M=LEN(A$):IF M<7 THEN A$=A$+" ":
GOTO 350
360 B$=LEFT$(A$,3)
370 IF B$="INV" OR B$="I  " THEN GOSUB 730:
GOTO 270
380 IF B$="SCO" THEN GOSUB 820:GOTO 270
```

```
390 IF B$="QUI" OR B$="Q  " THEN GOSUB 850:
GOTO 270
400 IF B$="HEL" OR B$="CLU" THEN PRINT:
PRINT "Sorry, there is no help for lost agents in this
game.":GOTO 270
410 IF B$="HID" THEN GOSUB 2360:GOTO 270
420 IF B$="STA" THEN GOSUB 2410:GOTO 270
430 IF B$="LOO" OR B$="L  " THEN GOTO 230
440 IF B$="N  " THEN C$="N  ":GOSUB 920:GOTO 270
450 IF B$="S  " THEN C$="S  ":GOSUB 920:GOTO 270
460 IF B$="E  " THEN C$="E  ":GOSUB 920:GOTO 270
470 IF B$="W  " THEN C$="W  ":GOSUB 920:GOTO 270
480 IF B$="U  " THEN C$="U  ":GOSUB 920:GOTO 270
490 IF B$="D  " THEN C$="D  ":GOSUB 920:GOTO 270
500 N=1
510 IF MID$(A$,N,1)=" " THEN C$=MID$(A$,N+1,3):
IF LEFT$(C$,1)<>" " THEN 540 ELSE 530
520 IF N<M THEN N=N+1:GOTO 510
530 PRINT:
PRINT "In that form, I don't know the word ";A$:
GOTO 270
540 IF B$="GO " OR B$="MOV" OR B$="WAL" THEN
KW=1:GOSUB 920:GOTO 270
550 IF B$="EXA" OR B$="SEA" THEN
GOSUB 2590:GOTO 270
560 IF B$="TAK" OR B$="GET" THEN KW=1:GOSUB
1600:GOTO 270
570 IF B$="DRO" OR B$="PUT" THEN KW=1:GOSUB
1140:GOTO 270
580 IF B$="UNL" THEN KW=1:GOSUB 2110:GOTO 270
590 IF B$="LOC" THEN KW=1:GOSUB 2440:GOTO 270
600 IF B$="OPE" THEN KW=1:GOSUB 1280:GOTO 270
610 IF B$="CLO" THEN KW=1:GOSUB 2510:GOTO 270
620 IF B$="REA" THEN KW=1:GOSUB 1440:GOTO 270
630 IF B$="PUS" OR B$="PRE" THEN KW=1:
GOSUB 2260:GOTO 270
640 IF B$="PUL" OR B$="UNP" THEN KW=1:
GOSUB 2980:GOTO 270
650 IF B$="TYP" THEN KW=1:GOSUB 3030:GOTO 270
660 IF B$="POU" OR B$="SPR" THEN KW=1:
GOSUB 2910:GOTO 270
670 IF B$="INS" THEN KW=1:GOSUB 1830:GOTO 270
680 IF B$="KIC" OR B$="HIT" THEN KW=1:
GOSUB 1970:GOTO 270
690 IF B$="MIX" OR B$="COM" THEN KW=1:
GOSUB 2800:GOTO 270
700 IF B$="THR" THEN KW=1:GOSUB 2190:GOTO 270
710 PRINT:
PRINT "I don't know that word.":GOTO 270
720 REM **** BEGIN SUBROUTINES ****
730 REM ** INVENTORY **
740 PRINT:PRINT "  You are carrying:"
750 PRINT
760 FOR I=1 TO 8
770 IF TR(I)=200 THEN PRINT "    ";IN$(I):AADS=1
780 NEXT I
790 IF AADS<>1 THEN PRINT "    nothing"
800 AADS=0
810 RETURN
820 REM ** SCORE **
830 PRINT:
PRINT "Out of 200 points, you got ";SCORE" points."
840 RETURN
850 REM ** QUITING **
860 PRINT:
PRINT "Out of 200 points, you got "SCORE" points."
870 PRINT
880 INPUT "Are you sure you want to quit";F$
890 IF F$="" THEN 870
900 IF F$="Y" OR F$="f" OR F$="YES" THEN END
910 PRINT:PRINT "Ok.":GOTO 270
920 REM ** MOVEMENT COMMAND **
930 C$=LEFT$(C$,1)
940 IF RO=10 AND GUARD=1 THEN PRINT:
PRINT "As you try to leave the guard spies you and
captures you. He alerts others":
PRINT "and soon you are tortured and killed.":
GOTO 3140
950 IF RO=9 AND C$="S" AND PLUG=0 THEN PRINT:
PRINT "As you step into the record room, the camera
spies you and alerts the guards.":
PRINT "They take you away and torture you which
eventually kills you.":GOTO 3140
960 IF RO=13 AND C$="S" AND TR(11)=16 THEN
PRINT:PRINT "As you step into the missile site, a
worker working on the MX-13":
PRINT "sees you and calls for guards. They take you
away and torture you. You":
PRINT "die in their hands.":GOTO 3140
970 IF RO=6 AND BOX=1 THEN PRINT:
PRINT "You can't go that way while you are crouching in
the box.":RETURN
980 IF RO=7 AND TR(11)=7 AND C$="W" THEN
PRINT:PRINT "As you try to leave, the worker grabs you
and calls for the guards. When the":
PRINT "guards come, they take you away to be tortured.
THE END!":GOTO 3140
990 IF C$="N" AND A(RO,1)=0 THEN PRINT:
PRINT "You can't go that way.":RETURN
1000 IF C$="S" AND A(RO,2)=0 THEN PRINT:
PRINT "You can't go that way.":RETURN
1010 IF C$="E" AND A(RO,3)=0 THEN PRINT:
PRINT "You can't go that way.":RETURN
1020 IF C$="W" AND A(RO,4)=0 THEN PRINT:
PRINT "You can't go that way.":RETURN
1030 IF C$="U" AND A(RO,5)=0 THEN PRINT:
PRINT "You can't go that way.":RETURN
1040 IF C$="D" AND A(RO,6)=0 THEN PRINT:
PRINT "You can't go that way.":RETURN
1050 IF C$="N" THEN RO=A(RO,1)
1060 IF C$="S" THEN RO=A(RO,2)
1070 IF C$="E" THEN RO=A(RO,3)
1080 IF C$="W" THEN RO=A(RO,4)
1090 IF C$="U" THEN RO=A(RO,5)
1100 IF C$="D" THEN RO=A(RO,6)
1110 IF RO=3 AND SOP1=0 THEN
SCORE=SCORE+10:SOP1=1:GOSUB 3100
1120 IF RO=10 AND DSF=0 THEN
SCORE=SCORE+30:DSF=1:GOSUB 3100
1130 GOTO 230
1140 REM ** DROP ROUTINE **
1150 FLAG=0:OBJECT=0
1160 PRINT
1170 FOR I=1 TO 8
1180 IF TR(I)=200 THEN FLAG=1
1190 NEXT I
1200 IF FLAG=0 THEN PRINT "You aren't carrying
```

```
anything.":RETURN
1210 FOR I=1 TO 8
1220 IF C$=O1$(I) AND TR(I)=200 THEN OBJECT=I
1230 NEXT I
1240 IF OBJECT=0 THEN PRINT "You're not holding
that item.":RETURN
1250 TR(OBJECT)=RO
1260 PRINT "Ok."
1270 RETURN
1280 REM ** OPEN ROUTINE **
1290 IF C$<>"DOO" AND C$<>"CAB" AND C$<>"VAU"
THEN PRINT:PRINT "You can't open that.":RETURN
1300 IF C$<>"DOO" THEN 1360
1310 IF C$="DOO" AND DOOR=0 AND RO=15 THEN
PRINT:PRINT "The door is locked.":RETURN
1320 IF C$="DOO" AND DOOR=1 AND RO=15 THEN
PRINT:PRINT "The door swings open, revealing a
passageway to the north.":
DOOR=2:A(10,2)=15:A(15,1)=10:RETURN
1330 IF C$="DOO" AND DOOR=1 AND RO=10 THEN
PRINT:PRINT "The door swings open, revealing a
passageway to the south.":
DOOR=2:A(10,2)=15:A(15,1)=10:RETURN
1340 IF C$="DOO" AND DOOR=2 AND RO=15 OR
RO=10 THEN PRINT:
PRINT "The door is already open.":RETURN
1350 PRINT:
PRINT "You can't see a door here.":RETURN
1360 IF C$<>"VAU" THEN 1400
1370 IF C$="VAUL" AND VAULT=0 AND RO=1 THEN
PRINT "You have to type in the correct code for the vault
to open.":RETURN
1380 IF C$="VAUL" AND VAULT=1 AND RO=1 THEN
PRINT "It's already open.":RETURN
1390 PRINT:
PRINT "You can't see the vault here.":RETURN
1400 IF C$="CAB" AND CAB=0 AND RO=14 THEN
PRINT "The cabinet is now open.":
CAB=1:TR(7)=14:RETURN
1410 IF C$="CAB" AND CAB=1 AND RO=14 THEN
PRINT "The cabinet is already open.":RETURN
1420 PRINT:PRINT "You can't see a cabinet here.":
RETURN
1430 RETURN
1440 REM ** READ ROUTINE **
1450 FLAG=0
1460 PRINT
1470 IF C$<>"PLA" THEN 1500
1480 IF RO<>6 THEN PRINT "You don't see the plaque
here.":RETURN
1490 PRINT "The plaque reads: 'Press in case of
emergency.'":RETURN
1500 FOR I=1 TO 8
1510 IF C$=O1$(I) AND TR(I)=200 THEN FLAG=1
1520 NEXT
1530 IF FLAG=0 THEN PRINT "You don't have that
item.":RETURN
1540 HH$=LEFT$(C$,3)
1550 IF HH$<>"NOT" AND HH$<>"NEW" AND
HH$<>"SLI" AND HH$<>"FIL" THEN PRINT "You can't
read that!":RETURN
1560 IF HH$="NEW" AND TR(2)=200 THEN PRINT
"The newspaper reads: 'Signs of MX-13 missiles being
shipped to Russia...' It":
PRINT "goes on but you can't make anymore out.":
RETURN
1570 IF HH$="SLI" AND TR(6)=200 THEN PRINT "The
slip reads: 'Porch needs fixing and new code is '971'.'":
RETURN
1580 IF HH$="FIL" AND TR(7)=200 THEN PRINT "The
file reads: 'TOP-SECRET Information on MX-13 and
deployment.'":RETURN
1590 IF HH$="NOT" AND TR(3)=200 THEN PRINT
"The note reads: 'New scientific breakthrough: analgesic
and liquid in beaker":PRINT "makes great acid!'":
RETURN
1600 REM ** GET ROUTINE **
1610 PRINT
1620 IF C$="GUA" THEN PRINT "You can't get that.":
RETURN
1630 IF C$="WOR" THEN PRINT "You can't get that.":
RETURN
1640 FLAG=0:HHG=0:OBJECT=0
1650 FOR I=1 TO 8
1660 IF TR(I)=200 THEN HHG=HHG+1
1670 NEXT
1680 IF HHG>=6 THEN PRINT "You can't carry
anymore.":RETURN
1690 FOR I=1 TO 8
1700 IF TR(I)=RO OR TR(2)=220 THEN FLAG=1
1710 NEXT I
1720 IF FLAG=0 THEN PRINT "There is nothing here to
pick up.":RETURN
1730 IF TR(2)=220 AND C$="NEW" AND TR(4)=220
THEN OBJECT=2:INSERT=0:
TR(4)=15:GOTO 1790
1740 IF TR(2)=220 AND C$="NEW" THEN
OBJECT=2:INSERT=0:GOTO 1790
1750 FOR I=1 TO 8
1760 IF C$=O1$(I) AND TR(I)=RO THEN OBJECT=I
1770 NEXT
1780 IF OBJECT=0 THEN PRINT "I don't see that object
here.":RETURN
1790 TR(OBJECT)=200
1800 PRINT "Ok.":IF TR(7)=200 AND SOP=0 THEN
SCORE=SCORE+50:SOP=1:GOSUB 3100
1810 IF OBJECT=4 AND DGF=0 THEN
SCORE=SCORE+20:DGF=1:GOSUB 3100
1820 RETURN
1830 REM ** INSERT ROUTINE **
1840 IF C$<>"NEW" THEN PRINT:
PRINT "You can't insert that.":RETURN
1850 INPUT "  Under what";F$:F$=LEFT$(F$,3)
1860 IF F$<>"DOO" THEN PRINT:
PRINT "You can't insert the newspaper under that.":
RETURN
1870 IF RO<>15 THEN PRINT:
PRINT "You can't see a door here.":RETURN
1880 PRINT:PRINT "Ok.":INSERT=1:TR(2)=220
1890 RETURN
1900 REM ** OBJECTS IN ROOM? **
1910 PRINT
1920 FOR I=1 TO 11
1930 IF TR(I)=RO THEN PRINT O2$(I)
1940 NEXT I
1950 RETURN
1960 END
1970 REM ** HIT ROUTINE **
```

```
1980 PRINT
1990 IF RO<>15 AND RO<>7 THEN PRINT "I can't use
that word here.":RETURN
2000 IF C$<>"DOO" AND C$<>"WOR" THEN PRINT
"You can't hit that.":RETURN
2010 IF C$="WOR" THEN GOTO 2080
2020 IF RO=15 AND C$="DOO" AND KEE=0 THEN
PRINT "You hear a sound from the other sound of the
door.":KEE=1:GOTO 2040
2030 IF RO=15 AND C$="DOO" AND KEE=1 THEN
PRINT "It doesn't work this time.":RETURN
2040 IF INSERT=0 THEN PRINT:PRINT "Sorry,
because of that last action, there is no way for you to
proceed.":GOTO 3140
2050 IF INSERT=1 THEN PRINT:
PRINT "There is now something on the newspaper."
2060 TR(4)=220
2070 RETURN
2080 IF RO=7 AND C$="WOR" AND WORKER=0 THEN
PRINT "The worker falls over the porch, yelling. As he
falls, something falls out":PRINT "of the worker's
pocket":TR(8)=7:TR(11)=255:WORKER=1:RETURN
2090 IF RO=7 AND C$="WOR" AND WORKER=1 THEN
PRINT "The worker is dead, because of you!":RETURN
2100 RETURN
2110 REM ** UNLOCK ROUTINE **
2120 PRINT
2130 IF RO<>15 AND RO<>10 THEN PRINT "I can't
use that word here.":RETURN
2140 IF C$<>"DOO" THEN PRINT "I can't unlock
that.":RETURN
2150 IF TR(4)<>200 THEN PRINT "You have nothing to
unlock the door with.":RETURN
2160 IF DOOR=0 THEN PRINT "The door is now
unlocked.":DOOR=1:GOTO 2180
2170 IF DOOR=1 OR DOOR=2 THEN PRINT "The door
is already unlocked."
2180 RETURN
2190 REM ** THROW ROUTINE **
2200 IF RO<>10 THEN PRINT:
PRINT "I can't use that word here.":RETURN
2210 IF C$<>"BRI" THEN PRINT:
PRINT "You can't throw that.":RETURN
2220 INPUT "  At what";F$:F$=LEFT$(F$,3)
2230 IF F$="GUA" AND GUARD=1 THEN
PRINT:PRINT "You hit the guard in the back of the head
with the brick and he becomes":
PRINT "unconscious.":TR(9)=255:TR(10)=10:
GUARD=0:TR(1)=10:SCORE=SCORE+20:
GOSUB 3100:RETURN
2240 IF F$="GUA" AND GUARD=0 THEN PRINT:
PRINT "The guard is already unconscious.":RETURN
2250 PRINT "You can't throw the brick at that.":
RETURN
2260 REM ** PUSH ROUTINE **
2270 IF RO<>6 AND RO<>11 AND RO<>12 AND
RO<>13 THEN PRINT:
PRINT "I can't use that word here.":RETURN
2280 PRINT
2290 IF C$="ONE" THEN PRINT "The elevator doors
close...":FOR I=1 TO 100:NEXT:
PRINT "Then they open....":RO=11:RETURN
2300 IF C$="TWO" THEN PRINT "The elevator doors
close...":FOR I=1 TO 100:NEXT:
PRINT "Then they open....":RO=12:RETURN
2310 IF C$="THR" THEN PRINT "The elevator doors
close...":FOR I=1 TO 100:NEXT:
PRINT "Then they open....":RO=13:RETURN
2320 IF C$="BUT" AND RO=6 AND GHT=0 THEN
PRINT "A siren sounds and a worker will be in here any
minute. The button goes in so":
PRINT "you can't press it again.":SIREN=1:GHT=1:
RETURN
2330 IF C$="BUT" AND RO=6 AND GHT=1 THEN
PRINT "You can't push the button again.":RETURN
2340 PRINT "You can't push that.":RETURN
2350 RETURN
2360 REM ** HIDE ROUTINE **
2370 IF RO<>6 THEN PRINT:
PRINT "I can't use that word here.":RETURN
2380 INPUT "  In what";F$:F$=LEFT$(F$,3)
2390 IF F$="BOX" THEN PRINT:PRINT "Ok, you are
now crouching in the box.":BOX=1:RETURN
2400 PRINT:PRINT "You can't hide in there.":RETURN
2410 REM ** STAND ROUTINE **
2420 IF BOX=0 THEN PRINT:
PRINT "You are already standing.":RETURN
2430 IF BOX=1 THEN PRINT:PRINT "Ok, you are now
standing outside the box.":BOX=0:RETURN
2440 REM ** LOCK ROUTINE **
2450 IF RO<>10 AND RO<>15 THEN PRINT:
PRINT "I can't use that word here.":RETURN
2460 IF C$<>"DOO" THEN PRINT:
PRINT "You can't lock that.":RETURN
2470 IF TR(4)<>200 THEN PRINT:PRINT "You have
nothing to lock the door with.":RETURN
2480 IF DOOR=0 THEN PRINT:
PRINT "The door is already locked.":RETURN
2490 IF DOOR=1 OR DOOR=2 THEN PRINT:
PRINT "Ok, the door is now closed and locked.":
DOOR=0:A(15,1)=0:A(10,2)=0:RETURN
2500 RETURN
2510 REM ** CLOSE ROUTINE **
2520 IF C$<>"DOO" AND C$<>"CAB" THEN
PRINT:PRINT "You can't close that.":RETURN
2530 IF C$="DOO" AND DOOR=2 AND RO=15 THEN
PRINT:PRINT "The door is now closed.":
DOOR=1:A(10,2)=0:A(15,1)=0:RETURN
2540 IF C$="DOO" AND DOOR=2 AND RO=10 THEN
PRINT:PRINT "The door is now closed.":
DOOR=1:A(15,1)=0:A(10,2)=0:RETURN
2550 IF C$="DOO" AND DOOR=0 OR DOOR=1 AND
RO=10 OR RO=15 THEN PRINT:
PRINT "The door is already closed.":RETURN
2560 IF C$="CAB" AND CAB=1 AND RO=14 THEN
PRINT:PRINT "The cabinet is now closed.":RETURN
2570 IF C$="CAB" AND CAB=0 AND RO=14 THEN
PRINT:PRINT "The cabinet is already closed.":RETURN
2580 PRINT "You can't close that here.":RETURN
2590 REM ** EXAMINE ROUTINE **
2600 PRINT
2610 IF C$="DOO" AND RO=15 OR RO=10 THEN
PRINT "The large, wooden door has a keyhole and a
small space below it. Through":PRINT "the keyhole you
can see a key, unreachable from this side.":RETURN
2620 IF C$="BRI" AND TR(1)=200 THEN PRINT "The
brick is just a normal, heavy brick.":RETURN
2630 IF C$="NEW" AND TR(2)=200 THEN PRINT "The
```

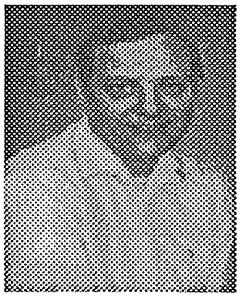
```
newspaper just has some readable writing on
it.":RETURN
2640 IF C$="NOT" AND TR(3)=200 THEN
PRINT "The note just has some writing on it.":RETURN
2650 IF C$="KEY" AND TR(4)=200 THEN
PRINT "I see nothing special about that.":RETURN
2660 IF C$="BEA" AND TR(5)=200 THEN
PRINT "I see nothing special about that.":RETURN
2670 IF C$="SLI" AND TR(6)=200 THEN PRINT "The
slip of paper just has some writing on it.":RETURN
2680 IF C$="FIL" AND TR(7)=200 THEN PRINT "The
file contains top-secret information.":RETURN
2690 IF C$="ASP" AND TR(8)=200 THEN PRINT "The
aspirin has 'BUFFERIN' on it.":RETURN
2700 IF C$="GUA" AND TR(9)=RO THEN
PRINT "The guard is a hard working employee of the
Russians.":RETURN
2710 IF C$="GUA" AND TR(10)=RO THEN
PRINT "The guard is unconscious.":RETURN
2720 IF C$="WOR" AND TR(11)=RO THEN PRINT "The
worker has something in his back pocket.":RETURN
2730 IF C$="PLA" AND RO=6 THEN
PRINT "The plaque has writing on it.":RETURN
2740 IF C$="BOX" AND RO=6 THEN PRINT "The box is
big enough for you to HIDE in it.":RETURN
2750 IF C$="CAB" AND RO=14 AND CAB=1 THEN
PRINT "The cabinet is open.":RETURN
2760 IF C$="CAB" AND RO=14 AND CAB=0 THEN
PRINT "The cabinet is closed.":RETURN
2770 IF C$="PLU" AND RO=2 AND PLUG=0 THEN
PRINT "The plug is connected to an outlet.":RETURN
2780 IF C$="PLU" AND RO=2 AND PLUG=1 THEN
PRINT "The plug is not connected to an outlet.":
RETURN
2790 PRINT "I can't examine that now.":RETURN
2800 REM ** MIX ROUTINE **
2810 IF C$<>"ASP" AND C$<>"BEA" THEN
PRINT:PRINT "I can't mix that.":RETURN
2820 IF C$="BEA" THEN GOTO 2870
2830 IF C$="ASP" AND TR(8)=200 THEN
INPUT "  In what";F$:F$=LEFT$(F$,3)
2840 IF F$="BEA" AND TR(5)=200 THEN PRINT:
PRINT "The beaker begins to bubble and inside there is
an acidy solution.":TR(8)=255:BEAKER=1:RETURN
2850 IF F$="BEA" THEN PRINT:PRINT "You don't have
the beaker.":RETURN
2860 PRINT:PRINT "You can't mix the aspirin with
that.":RETURN
2870 INPUT "  With what";F$:F$=LEFT$(F$,3)
2880 IF F$="ASP" AND TR(8)=200 THEN PRINT:
PRINT "The beaker begins to bubble and inside there is
an acidy solution.":BEAKER=1:TR(8)=255:RETURN
2890 IF F$="ASP" THEN PRINT:
PRINT "You don't have the aspirin.":RETURN
2900 PRINT:PRINT "You can't mix that with the
beaker.":RETURN
2910 REM ** POUR ROUTINE **
2920 PRINT
2930 IF RO<>8 THEN PRINT:
PRINT "I can't use that word here.":RETURN
2940 IF C$="BEA" OR C$="LIQ" AND BEAKER=0
THEN PRINT:PRINT "The liquid is stuck inside (for
now).":RETURN
2950 IF C$="BEA" OR C$="LIQ" AND BEAKER=1
THEN INPUT "  On what";F$:F$=LEFT$(F$,3)
2960 IF F$="WAL" OR F$="NOR" THEN PRINT:PRINT
"The wall to the north, becomes a giant hole, allowing
you to go that way.":A(8,1)=3:TR(5)=255:RETURN
2970 PRINT:PRINT "You can't pour the liquid on
that.":RETURN
2980 REM ** PULL ROUTINE **
2990 IF RO<>2 THEN PRINT:
PRINT "I can't use that word here.":RETURN
3000 IF C$<>"PLU" THEN PRINT:
PRINT "You can't pull that.":RETURN
3010 IF PLUG=1 THEN PRINT:
PRINT "The plug is already unplugged.":RETURN
3020 PRINT "Ok, the plug is now unplugged.":
PLUG=1:RETURN
3030 REM ** TYPE ROUTINE **
3040 IF RO<>1 THEN PRINT:
PRINT "I can't use that word here.":RETURN
3050 IF C$<>"971" THEN PRINT:
PRINT "You type in the wrong code and a loud siren
sounds. Guards rush in and take":
PRINT "you away. You DON'T return.":GOTO 3100
3060 PRINT:PRINT "Congratulations! You got the right
code. You rush outside to find that you are":
PRINT "inside a large sewer system. But, above you see
a manhole, so you climb out.":
PRINT "You rush to tell the police your story."
3070 IF TR(7)=200 THEN PRINT "You show the police
the file and you are rewarded with a hero's
reward!":SCORE=SCORE+70:GOSUB 3100:GOTO 3140
3080 PRINT "You have no proof to show the police, so
they lock you up. In exactly five":PRINT "days, the
Soviets launch their supply of MX-13's on the U.S. All":
PRINT "life as you know it, stops to exist. The human
race is never heard of again."
3090 GOTO 3140
3100 REM ** SCORE ROUTINE **
3110 REM
3120 PRINT@(0,39),"Score:";SCORE
3130 RETURN
3140 REM ** SCORING **
3150 PRINT:
PRINT "Out of 200 points, you got ";SCORE;" points."
3160 PRINT:INPUT "Would you like to play again";
F$:F$=LEFT$(F$,1)
3170 IF F$="y" OR F$="Y" THEN GOTO 3210
3180 IF F$="n" OR F$="N" THEN PRINT:
PRINT "Ok.":PRINT:PRINT:END
3190 PRINT "Please answer the question."
3200 GOTO 3160
3210 RO=17:SOP=0:SOP1=0:DOOR=0:KEE=0:
SCORE=0:GUARD=0:BOX=0:BEAKER=0:DSF=0:
DGF=0
3220 TR(1)=17:TR(2)=17:TR(3)=8:TR(4)=255:
TR(5)=8:TR(6)=5:TR(7)=255:TR(8)=16
3230 CLS:GOTO 200
3240 REM ** ROOM DESCRIPTIONS **
3250 REM
3260 PRINT@(0,0),CHR$(31);ROOM$(RO);
3270 PRINT@(0,39),"SCORE:";SCORE
3280 PRINT
3290 REM
3300 ON RO GOSUB 3320,3380,3450,3500,3550, 3600,
     3660,3700,3750,3790,3840,3890,3940,3990,4040,
```

```
      4100,4160
3310 RETURN
3320 REM ROOM ONE
3330 PRINT "Vault Room"
3340 PRINT "You are inside a very large room which is
the exit to the outside world. But"
3350 PRINT "before every person gets outside, they must
pass through a giant, metal vault"
3360 PRINT "which has a large keyboard attached to it."
3370 RETURN
3380 REM ROOM TWO
3390 PRINT "Maintenance Room"
3400 PRINT "This is a room where maintenance
equipment is stored. There are many  brooms,"
3410 PRINT "dustpans and mops around here. A dark
passage way leads to the south, while"
3420 PRINT "a lighted way leads to the east. There is a
plug on the east wall connected"
3430 PRINT "to an outlet."
3440 RETURN
3450 REM ROOM THREE
3460 PRINT "Secret Room"
3470 PRINT "You are in a secret room behind the base
laboratory. The hole in the south"
3480 PRINT "wall is smaller on this side, preventing you
from going south."
3490 RETURN
3500 REM ROOM FOUR
3510 PRINT "Hallway"
3520 PRINT "This is a non-descript hallway running
north and south."
3530 RETURN
3540 RETURN
3550 REM ROOM FIVE
3560 PRINT "Repair Closet"
3570 PRINT "This is a repair closet where hammers,
nails and screwdrivers are kept. The"
3580 PRINT "only exits is the way you came."
3590 RETURN
3600 REM ROOM SIX
3610 PRINT "Storage Room";:IF BOX=1 THEN PRINT "
(crouching in the box)"
3620 PRINT "You are in a storage room where
miscellaneous equipment is stored. There is a"
3630 PRINT "large, heavy box in the south corner. To the
east is a balcony. Also on the"
3640 PRINT "north wall, there is a button with a small
plaque beneath it."
3650 RETURN
3660 REM ROOM SEVEN
3670 PRINT "Balcony"
3680 PRINT "This is an old, crumbling balcony."
3690 RETURN
3700 REM ROOM EIGHT
3710 PRINT "Laboratory"
3720 PRINT "You are in a scientific laboratory where
great experiments are done. The usual"
3730 PRINT "array of supplies and equipment is here.
The north wall is a bit less solid":
PRINT "then the others."
3740 RETURN
3750 REM ROOM NINE
3760 PRINT "Hallway"
3770 PRINT "This is a very non-descript hallway
running east to west."
3780 RETURN
3790 REM ROOM TEN
3800 PRINT "Guard Room"
3810 PRINT "You are in a guard room, where a guard is
supposed to guard the cell to the"
3820 PRINT "south. Passageways run off in three
directions."
3830 RETURN
3840 REM ROOM ELEVEN
3850 PRINT "Elevator"
3860 PRINT "This is an elevator connecting all three
levels of the base. There is a panel"
3870 PRINT "here with three buttons labeled 1,2 and 3."
3880 RETURN
3890 REM ROOM TWELVE
3900 PRINT "Elevator"
3910 PRINT "This is an elevator connecting all three
levels of the base. There is a panel"
3920 PRINT "here with three buttons labeled 1,2 and 3."
3930 RETURN
3940 REM ROOM THIRTEEN
3950 PRINT "Elevator"
3960 PRINT "This is an elevator connecting all three
levels of the base. There is a panel"
3970 PRINT "here with three buttons labeled 1,2 and 3."
3980 RETURN
3990 REM ROOM FOURTEEN
4000 PRINT "Record Room"
4010 PRINT "This is a record room where top secret
information is kept. There is a file"
4020 PRINT "cabinet on the west wall and a camera on
the west wall near the ceiling."
4030 RETURN
4040 REM ROOM FIFTEEN
4050 PRINT "North Cell"
4060 PRINT "This is a north cell where top security
prisoners are kept. There is a"
4070 PRINT "large, wooden door on the north wall. To
the south a light can be seen."
4080 IF INSERT=1 THEN PRINT "Under the door, you
can see a newspaper."
4090 RETURN
4100 REM ROOM SIXTEEN
4110 PRINT "Missille site"
4120 PRINT "This is a large room in the shape of a
missile silo, where a very important"
4130 PRINT "missile is kept. The rounded ceiling is
partly open, awaiting the preparation"
4140 PRINT "of the MX-13, being kept in the center of
the room."
4150 RETURN
4160 REM ROOM SEVENTEEN
4170 PRINT "South Cell"
4180 PRINT "You are in a cell where prisoners are kept.
The second part of this two room"
4190 PRINT "cell is to the north while a crack in the
ceiling is letting in a large amount"
4200 PRINT "of light."
4210 RETURN
4220 END
4230 REM ** INITIALIZE **
```

```
4240 CLS:PRINT "Please wait while I set the adventure"
4250 SCORE=0:RO=17:GUARD=1:FLAG=0:
KEE=0:DOOR=0
4260 A(15,7)=99:REM PROBLEM DOOR 1
4270 A(8,7)=100:REM PROBLEM DOOR 2
4280 FOR X=1 TO 17
4290 FOR Y=1 TO 6
4300 READ A(X,Y)
4310 NEXT Y
4320 NEXT X
4330 FOR X=1 TO 11
4340 READ O1$(X),O2$(X),TR(X),IN$(X)
4350 NEXT X
4360 FOR I=1 TO 17
4370 READ ROOM$(I)
4380 NEXT I
4390 RETURN
4400 REM ** ROOM DATA **
4410 DATA 0,4,0,0,0,0:REM ROOM 1
4420 DATA 0,8,3,0,0,0:REM ROOM 2
4430 DATA 0,0,0,2,0,0:REM ROOM 3
4440 DATA 1,10,5,0,0,0:REM ROOM 4
4450 DATA 0,0,0,4,0,0:REM ROOM 5
4460 DATA 0,12,7,0,0,0:REM ROOM 6
4470 DATA 0,0,0,6,0,0:REM ROOM 7
4480 DATA 0,0,9,0,0,0:REM ROOM 8
4490 DATA 0,14,10,8,0,0:REM ROOM 9
4500 DATA 4,0,11,9,0,0:REM ROOM 10
4510 DATA 0,0,0,10,0,0:REM ROOM 11
4520 DATA 6,0,0,0,0,0:REM ROOM 12
4530 DATA 0,16,0,0,0,0:REM ROOM 13
4540 DATA 9,0,0,0,0,0:REM ROOM 14
4550 DATA 0,17,0,0,0,0:REM ROOM 15
4560 DATA 13,0,0,0,0,0:REM ROOM 16
4570 DATA 15,0,0,0,0,0:REM ROOM 17
4580 DATA "BRI","There is a brick on the floor near
you.",17,"a brick"
4590 DATA "NEW","On the floor, there is an old
newspaper.",17,"a newspaper"
4600 DATA "NOT","There is a note laying nearby.",8,"a
note"
4610 DATA "KEY","There is a shining key here.",255,"a
key"
4620 DATA "BEA","There is a scientific beaker full of
liquid near you.",8,"a beaker full of liquid"
4630 DATA "SLI","There is a slip of paper lying
here.",5,"a slip"
4640 DATA "FIL","A file containg important information
is here.",255,"a file"
4650 DATA "ASP","There is a aspirin here.",255,
"an aspirin"
4660 DATA "GUA","There is an alert guard here.",10,""
4670 DATA "GUA","There is an unconscious guard
here.",255,""
4680 DATA "WOR","There is a missile worker
here.",16,""
4690 DATA Vault Room,Maintenance Room,Secret
Room,Hallway,Repair Closet
4700 DATA Storage Room,Balcony,Laboratory,
Hallway,Guard Room,Elevator
4710 DATA Elevator,Elevator,Record Room,North
Cell,Missille Site
4720 DATA South Cell
```

HINTS & TIPS

TIPS FROM KELLY
by Kelly Bates



I recently tried to install a 1.2 meg. floppy drive in my Model 4, but unfortunately it didn't work. Using TRSDOS 6, I tried to format a floppy (in external 3 position) in every way that I knew how, but the verify cycle failed on each and every track.. Another idea that won't work due to the limitation of the hardware. The good news is that the 3 1/2" drive (1.4) works great.

Here is a list of the DOSes I have converted to 3 1/2" 720K format for the 4P (will also boot in the Mod 4 desktop):

Model III mode:

- LDOS 5.1.4 (80 track)
- MULTIDOS 1.3 (80 track)
- MULTIDOS 1.6 (80 track)
- MULTIDOS 1.7 (80 track)
- NEWDOS/80 (80 track)
- SUPERDOS 1.3B (single-sided 40 track)
- DOSPLUS 3.5 (80 track)
- TRSDOS 1.3 (40 double-sided 40 track)

- CP/M (Montezuma — 710K)

Model 4 mode:

- DOSPLUS 1.0 (80 track)
- MULTIDOS 2.0 (80 track)
- TRSDOS/LS-DOS 6 (double-sided 40 track)

I have also converted the PENETRATOR game from Melbourne House. It is protected, but COPY-CAT4 will copy it onto a 40 track 3 1/2" disk. Also converted a special NEWDOS 2.0 system disk that is primarily used to boot the 4P into Model III operation. SYS22/SYS is added, which contains the Model III ROM. It is only 40 track, single-sided on the 3 1/2" disk, but now you don't need to maintain a copy of the MODELA/III file.

I will share this stuff for $8.00 per copy, but I request that you send a 5 1/4" disk with the version requested. This will insure that I am not breaking any laws. Please note that I am not able to provide manuals for the above DOSes.

If someone requests a conversion of a DOS that I don't own, I will attempt to convert the one that is sent to me

IMPORTANT RULES FOR SAFE FAX

from the TRSTimes Vault
original author unknown

Q: Do I have to be married to have safe fax?
A: Although married people fax quite often, there are many single people who fax complete strangers every day.

Q: My parents say they never had fax when they were young and were only allowed to write memos to each other until they were twenty-one. How old do you think someone should be before they can fax?
A: Faxing can be performed at any age, once you master the correct technical details.

Q: If I fax myself, will I go blind?
A: Certainly not, as far as we can see.

Q: There is a place on our street where you can go and pay for fax. Is this legal?
A: Yes. Many people have no other outlet for their fax drives and must pay a "professional" when their needs to fax become too great.

Q: Should a cover always be used for faxing?
A: Unless you are really sure of the one you're faxing, a cover sheet should be used to insure safe fax.

Q: What happens when the wrong buttons get pushed and I fax prematurely?
A: Don't panic. Many people fax prematurely when they haven't faxed in a long time. Simply start over. Most people won't mind if you try again.

Q: Sometimes when I fax, it's the other person who isn't fully ready, so that faxing doesn't work right. Is there any solution for this?
A: There is only one solution here: always be very patient and compassionate with your fax partner. Talk them through it carefully, and apply lots of fax-play before you start. Explain to them the importance of pressing the F Spot before you begin, and show them how to do this for themselves.

Q: Recently a completely unwanted fax was transmitted into my office. Does this constitute rape? And if so, can I bring charges and sue the responsible party? Would I be able to collect damages?
A: So far our police departments and our justice

system do not take fax rape too seriously as a charge, even though we all know that many people suffer greatly from this heinous crime. Concerned citizens are now taking steps to combat fax rape and faxism in all its forms, so write your Congressman and hold on to your evidence. The legal climate surrounding this offense may soon change.

Q: I have a personal and business fax. Can transmissions become mixed up?
A: Being bi-faxual can be confusing, but as long as you use a cover with each one, you won't transmit anything you're not supposed to.

BOB-WHAT ABOUT HIM?

Humor by DAVE GARR

REDMOND, Washington--January 4, 1995
In response to customer inquiries, Microsoft today clarified the naming policy for Bob(tm), its new software product designed for computer beginners. Contrary to rumors, Microsoft will not demand that all persons formerly named "Bob" immediately select new first names.

"I don't know where these rumors come from," commented Steve Balmer, Microsoft Executive Vice President for Worldwide Sales and Support. "It's ridiculous to think Microsoft would force people outside the computer industry to change their names. We won't, and our licensing policies for people within the industry will be so reasonable that the Justice Department could never question them."

Balmer said employees of other computer companies will be given the opportunity to select new names, and will also be offered a licensing option allowing them to continue using their former names at very low cost.

The new licensing program, called Microsoft TrueName(tm), offers persons who want to continue being known by the name Bob the option of doing so, with the payment of a small monthly licensing fee and upon signing a release form promising never to use OpenDoc. As an added bonus, Bob name licensees will also be authorized to display the Windows 95 logo on their bodies.

Persons choosing not to license the Bob name will be given a 60-day grace period during which they can select another related name. "We're being very lenient in our enforcement of the Bob trademark," said Bill Newkom, Microsoft's Senior Vice President of Law and Corporate Affairs. "People are still free to call themselves Robert, Robby, or even Rob. Bobby however is derivative of Microsoft's trademark and obviously can't be allowed."

Microsoft also announced today that Bob(tm) Harbold, its Executive Vice President and Chief Operating Officer, has become the first Microsoft TrueName licensee and will have the Windows 95 logo tattooed to his forehead.

It has come to our attention that Microsoft is already planning a scaled-down version for the segment of the population that is anticipated to find BOB(tm) too difficult to learn. This revelutionary new concept of utter simplicity will be named BILLY-BOB. Ed.

GAMES FOR THE MODEL I EMULATOR

by Wayne Westmoreland

The following programs were written by myself (Wayne Westmoreland) and Terry Gilman. I've listed them in the order we wrote them, however I don't remember the exact dates (old age I guess). The programs display copyright notices but I hereby release them into the public domain, do with them as you please.

TANKZONE/CMD
ARMORED/CMD
These two programs are 2 different versions of our first game, a version of the old coin-op game BATTLEZONE. The original was called TankZone 2000 (I hate that name!). We submitted it to Scott Adams' company adventure International whose marketing droids made us change the pyramids to houses, the flying saucer to a robot and the name to Armored Patrol.

DEFENDER/CMD
ELIMINAT/CMD
This was our second program, a version of William's Electronics DEFENDER, although I believe that it was the first to reach market. Once again we had to make changes and renamed it Eliminator. However we ended being sued (or at least threatened with suit) from William's Electronics. We gave them a percentage and then it was the "official" TRS-80 version of Defender.

SEADRAG/CMD
This game was our original idea, based on a suggestion from my Dad after he saw the game SCRAMBLE, however we borrowed the ending from the game PHOENIX. This game is without a doubt our favorite.

REARGARD/CMD
This was the first game we did for business instead of fun. We did it at the request of Adventure

International who wanted a TRS-80 version of a game written for the Atari 800 by Neil Larimer. This is our least favorite.

DONKKONG/CMD

We wrote this game just because we liked Nintendo's Donkey Kong and played it so much. We never marketed this game because we couldn't get permission from Nintendo, however we had so much fun writing it, we didn't care.

ZAXXON/CMD

This time we got permission first. Sega gave us a license for the game ZAXXON and then we wrote this. By this time, Tandy/Radio Shack had started accepting third party programs for sale in their stores and so we struck a deal with them instead of Adventure International. However manufacturing problems on our end limited the distribution.

I hope you enjoy them,
Wayne Westmoreland
wayne.westmoreland@srs.gov

P.S Except for Armored Patrol there aren't online instructions, basically use the arrow keys and the space bar. On Eliminator the enter key is your "smart bomb" and the clear key is "hyperspace". Shift/Break aborts all the games.

A Mod I emulator file entitled AI_GAMES/DSK has been uploaded to the TRSuretrove BBS and is available free of charge to all callers. Our thanks to Wayne Westmoreland for sharing his creativity with us. *Ed.*

TEMPCONV/BAS REVISITED

by Henry H. Herrdegen

Typing your conversion program and reading Roy's article (*from TRSTimes 7.4*), stretched my memory: "RANKINE"? Seems I heard about it somewhere. But how about "REAUMUR"? It is a system that has been widely used in the non-english speaking Europe, with freezing point 0, and boiling point 80. I still remember the thermometers of my youth in Austria: left side Celcius, right side Reaumur!

A little research turned up the following facts: Reaumur, Rene Antoine Ferchault de, french physicist, 1683 - 1757, invented the improved thermometer 1731`, used mainly in Germany. I also found a curio, a scale invented by a De Lisle (no reference to him found), supposedly used in Russia, which has the boiling point a 0 and the freezing point as +150 (it's cold in Siberia, and no negativism please!) My admittedly not up-to-date encyclopediam gives the two absolute zero temperatures as 273.10 and 459.58 (to refresh Roy's memory only — it has no effect on the calculations), and mentions a difference between "absolute" and "natural" zero. But let's not get into that can of worms.

Here is what I did to your perfectly good program:

To add the Reaumur scale, the loops in lines 14, 110 & 210 are increased; line 162 is added to the ON - GOTO section; the TM(3) expression in line 200 added, incrementing the absolutes to 4 and 5.

I could not help, but thought that -999 Celcius is a bit impossible, as tehere is no such thing as a minus degree Kelvin. Putting in the error trap line 202 necessitated another line, 204, to strip the equally impossible minus sign from the Kelvin zero result at Rankine 0 input> I also felt the old numbers could be wiped out for a new conversion, so, the added GOSUB 24 in line 240 and line 24 itself. While I was at it, the colons after the names seemed a good idea, and got added in line 16.

Change or add the following lines to the line listing from Issue 7.4.

```
14 FOR X=1 TO 5:READ NM$(X):NEXT
16 DATA "Celsius:","Fahrenheit:","Reaumur:",
"Kelvin:","Rankine:"
24 V=9:H=45:A$=CHR$(30):FOR X=1 TO 4:
V=V+1:NEXT:RETURN
110 V=8:FOR X=1 TO 5:A$=NM$(X):GOSUB 23:
V=V+1:NEXT
140 ON V-7 GOTO 160,160,162,170,180
162 TM(1)=VAL(IN$)*1.15:GOTO 200
200 TM(2)=TM(1)*9/5+32:TM(3)=TM(1)/1.25:
TM(4)=TM(1)+273.11:TM(5)=TM(2)+459.6
202 IF TM(4)<-.1 THEN
A$="IMPOSSIBLE TEMPERATURE!!":
V=14:GOSUB 21:GOTO 220
204 IF TM(4)<0 THEN TM(4)=0
210 V=8:H=45:FOR X=1 TO 5:
PRINT@SW*V+H,USING"######.#";TM(X);:
V=V+1:NEXT
240 IF I$="y" OR I$="Y" THEN V=14:
A$=CHR$(31):GOSUB 20:GOSUB 24:GOTO 120
ELSE IF I$="N" OR I$="n" THEN CLS: END
ELSE 220
```

I learn a great deal dissecting your programs, and hope you are not miffed by my fooling around with them.

On the contrary, I am happy that people find the programs interesting enough to customize. I enjoy seeing the add-on bells and whistles and, by the way, I also learn a little something. Incidentally, line 22 can be deleted. It is my standard code for right-justifying text, but TEMPCONV/BAS does not use it. *Ed.*

C PROGRAMMING TUTORIAL

Part 6

by J.F.R. "Frank" Slinkman

The Microsoft BASIC for the TRS-80 offers two methods of file I/O and management: a serial byte stream and "random" I/O of fixed block lengths up to a maximum of 256 bytes per block.

In BASIC, there are three ways to open files:

OPEN"I",n,filename/ext:d

(where "n" is the buffer number) to read a serial byte stream;

OPEN"O",n,"filename/ext:d"

to write a serial byte stream; and

OPEN"R",n,"filename/ext:d"[,block_size]

(with an associated FIELD command) to read or write to a file consisting of blocks of bytes, with the block size fixed unless you close the file and reopen it specifying another block size.

The C language supports these two file types, but with much more versatility and power. It also supports a third type of file -- a "directory" file, which is essentially a block file, with records as described by the struct in the header file dirent/h.

Also, C treats external devices, such as the keyboard, monitor and printer, as byte stream files.

There are three files which are always open, namely "stdin," "stdout," and "stderr." On most microcomputers, stdin is the keyboard, and both stdout and stderr are the monitor screen.

Please note that the listing in part 5 from TRSTimes 8.1 contains an omission: On the bottom of page 26 in column 2

return

should be

return count

This scheme gives the user the ability to use I/O redirection to substitute other devices for stdin and stdout while keeping error messages separate from program output.

Devices like the keyboard and printer are called "character special" or "tty" devices because they do not have full I/O capabilities. You can't write to the keyboard, for example, or read stored data from a printer.

So far, the only way we've sent any data to the printer is by using standard I/O redirection.

But if we open the printer as a file, we can print anything the printer is capable of printing.

First, we would declare a variable to be of a type which is called a "file pointer" (pointer to object of type "FILE") or "file designator," as follows:

```
FILE    *printer;
```

Of course, the variable doesn't have to be "printer." It could be "fred," "laser_jet," "pr," or any other legal variable name you care to use.

Then we would open the file using the standard fopen() function, as follows:

```
printer = fopen( "*pr", "w" );
```

We use "*pr" here because that is the name TRS/LS-DOS uses for the printer device. If you are using some other type of system, you need to use that system's device name (e.g.: "LPT1:" for MeSs-DOS). The printer file is opened in the "w" (write only) mode for reasons which should be obvious.

Once the printer file is opened, there are a number of standard functions can be used to send data to it.

For example, "putc('X',printer);" will cause the character "X" to be printed.

The standard fprintf() function can be used to send formatted data to the printer (or any other file stream, for that matter) in exactly the same way we have used printf() in the past to send formatted output to the monitor display.

For example:

```
fprintf( printer, "%s\n", "Hello!" );
```

will cause the string "Hello!" to be printed, and the printer to advance to the next line.

In fact, every time we have used printf() in the past, the system has actually been executing fprintf() similarly to the following:

```
PRINTF POP    AF          ;p/u RET address
       LD     HL,STDOUT   ;p/u file ptr
       PUSH   HL          ;put ptr on stack
       PUSH   AF          ;restore RET addr
       JP     FPRINTF     ;go output data
```

When a routine calls a function, it pushes the arguments in reverse order before making the call. This lets the called function see the arguments in the correct order.

In the fprintf() example above, the first thing pushed is a pointer to the string "Hello!" Next a pointer to the control string is pushed, and then the file pointer. The call to FPRINTF causes the return address to be pushed.

Thus fprintf() will see the return address at SP+0; the file pointer at SP+2; the control string pointer at SP+4, and the pointer to "Hello!" at SP+6.

Hopefully, you now understand how the assembler code above would convert

```
    printf( "control", data );
```

to

```
    fprintf( stdout, "control", data );
```

Each of the three file types (stream, block and directory) is treated differently, and has its own set of associated functions.

However, stream and block access aren't totally discrete as they are in BASIC. There are functions which enable you to access a file opened as a data stream as though it was a block file, and vice versa.

Most of you BASIC programmers are going to be shocked to learn just how limited you've been when it comes to file access and management.

In BASIC, for example, you can only open a file for stream output one way. If you're creating a file, you have to start writing at the first byte. If you open an existing file, you can only append new data at the end.

You can also do those things in C if you want, but you can also move to any desired point in the stream, read existing data, and/or replace old data virtually at will.

In fact, C offers SIX different ways to handle a data stream file, not just the one offered by BASIC.

The following program writes a ASCII string to a disk file and then reads it back in reverse order -- something that can only be simulated in BASIC:

```
/*   prog10.c  */

#include     <stdio.h>
#option      REDIRECT     OFF
#option      FIXBUFS      ON
#option      MAXFILES     1

char         filename[] = { "prog10/dat:1" };

main( argc, argv )
int argc;  char **argv;
{
    static FILE      *fp;
    static int       size;

    if ( argc != 2 )
    {        puts( "usage:  prog10 string" );
             exit(1);  }

    if ( !( fp = fopen( filename, "w+" ) ) )
    {        perror( "fopen()" );  exit(1);        }

    size = strlen( argv[1] );
    if (fwrite( argv[1],sizeof(char),size,fp ) != size)
    {        puts( "fwrite() error" );  exit(1);    }

    while ( size )
    {        fseek( fp, (long)--size, 0 );
             putc( getc( fp ), stdout );  }

    putc( '\n', stdout );
    fclose( fp );
    remove( filename );
}
```

After we use some non-standard Pro-MC options to reduce the final program size, we declare and initialize the variable "filename" as a char array large enough to hold the file name we'll be using.

Inside main(), we declare the static variable "fp" to be of type pointer to "FILE," and the static variable "size" to be of type int.

Now we check the command line to ensure there are two arguments. The first, of course, will be the program name, and the second will be the string we wish to store on disk, then read back and display backwards.

Note a slight programming style change with this statement. The closing brace isn't on a line by

itself but is on the same line as the last statement.

Personally, I think it's clearer to put the braces on a separate lines, but this way saves space. Of course, the compiler doesn't care, since it ignores all white space characters anyway.

Look at "exit(1);". This causes immediate program termination and return to TRS/LS-DOS Ready. The argument is non-zero to indicate an abort. A zero indicates successful completion.

This convention is valid across all platforms. On the TRS-80, a non-zero value is useful because if the program is invoked from a JCL file, a non-zero return value will cause the JCL to terminate.

Next we open the file "prog10/dat:1" in the "w+" mode. If you'll refer to the docs for fopen(), you'll see where this allows us to both read and write to the file. Also, if the file exists, it will be truncated (all data erased). If the file does not exist, it will be created.

If you want the file to exist somewhere other than Drive 1, you need to change the drive number in this string. Alternatively, as an "exercise for the reader," you could eliminate this initialization of "filename," and have main() take the file name and/or drive number as command line arguments, and build the "filename" string using command line data.

If fopen() returns a NULL pointer, the file could not be opened. If you'll refer to the docs, you'll see where the associated error number will be stored in the system's global "errno" variable; so perror() can be used to report the specific error.

Next we use the standard strlen() function to get the length of the string in argv[1] and assign it to "size."

Now fwrite() is used to write the string to the file. If you'll look at the docs for fwrite(), you'll see that, if successful, this function returns the number of data items (chars, in this case) actually written.

If there's an error, however, fwrite() doesn't put the error number in "errno;" so we can't use perror(). Instead, we have to make up our own error message.

The null character at the end of the string will not be written to the file because we told fwrite() to write "size" characters, and strlen() doesn't count a string's terminating null.

We could have performed a virtually identical write using:

```
fprintf( fp, "%s", argv[1] );
```

If you choose this method (and I hope you'll try both ways), the string's terminating null character will be written to disk. You'll also have to change the error check, since fprintf() would return one (one item printed) if successful, and EOF (-1) if an error occurred.

Now we set up a "while" loop governed by the "size" variable.

The location of bytes in a file is "base zero," just as with arrays. Thus the first byte in a file is byte #0, and the 15th byte is byte #14. This is why "size" is pre-decremented before it is sent to fseek().

If you'll look at the fseek() docs, you'll see how easy it is to step to any portion of the file within the range of a long int (plus or minus 2 gigabytes) either from the start, the current position or the end of the file.

What this loop does is point to the file bytes in reverse order (as determined by the value of "size") via fseek(), read the pointed-to characters via getc(), and display them on the monitor screen via putc().

Note the pre-decremented value of "size" is cast to a long to satisfy the requirements of fseek(). Had we failed to do this, fseek() (and therefore the program) would fail.

The resulting disk reads will be slow because fseek() in combination with an update mode causes the buffer to be flushed and the entire disk sector read anew for each byte read.

After the loop is exited, an ending newline character is sent to the screen via putc().

Finally, the file is closed and then removed via fclose() and remove(), respectively. If you'll look at the docs, you'll see files must be closed before they can be remove()'d or unlink()'ed.

Now compile prog10.c, and invoke it as follows:

```
prog10 sdrawkcab si siht
```

Hey! It didn't work! You got the "usage" message. What happened?!!!

You input FOUR command line arguments, dummy, not two; so the "argc!=2" check aborted the program.

Rats! You mean this silly program can only work with one word? Let's find out. Try entering this:

prog10 "pu evig I ro krow retteb siht"

Ah. Much better. And you learned you can use quote marks to delimit an ASCII string on the command line.

Basically, there are four things you can do to a byte stream file: read, write, append and/or update. "Append" means new writes go to the end of the file. "Update" means you can both read and write to the file.

As I mentioned previously, there are six different ways ("modes") a byte steam type file can be opened.

"r" read only;

"w" write only -- if file exists, truncate it (erase all data and start writing at byte #0); or create new file for writing;

"a" append -- open existing file for appending, or create new file for writing;

"r+" open for update;

"w+" truncate existing file or create new file for update; and

"a+" open or create for update (appending).

The "r" mode is the equivalent of OPEN"I" in BASIC. The "a" mode corresponds to OPEN"O". The "w" mode is the same as if, in BASIC, you KILLed the file, and then created it anew via OPEN"O".

You cannot use "r" or "r+" to create a file. Obviously, a file has to be created and written to before it can be read.

Also, there are some restrictions on how a file is accessed in an "update" mode. You may not read immediately after a write without first calling fseek() or rewind() to flush the file buffer and update the drive control table. If a read of any kind returns EOF, you must call cleareof() before attempting another read operation.

You may not write immediately after a read without an intervening fseek() or rewind().

O.K. Enough about byte stream type files for now. Let's look at how directory files are accessed.

Prog11.c below is taken from a larger program which I wrote out of frustration over how BACKUP copies individual files in the wrong order.

While David Goben's FBACKUP does process files in the correct sequence, it's buggy and not reliable enough to use when important data is involved.

But FBACKUP did provide the inspiration behind my BYDATE utility, of which the following code, which handles the reading of directories, is part.

```
/*   prog11.c */

#include     <stdio.h>
#include     <dirent.h>
#include     <stat.h>
#include     <time.h>
#include     <ustat.h>

#option      INLIB

#define      CTRL_S 0x13
#define      RECD   struct record

voidreport();

RECD         {        long     size;
                      long     date;
                      char     name[15];      }
    *array;
/*=*=*=*=*=*=*=*/
main( argc, argv )
int argc;    char **argv;
{   register int      count;

    if ( argc != 2 || argv[1][0] != ':' )
    {       puts( "usage: prog11 :d" ); exit(EOF); }

            {       struct ustat      ubuf;

                if ( ustat( argv[1][1] - '0', &ubuf ) )
                {     perror( "ustat()" ); exit(EOF); }
            }

    if ( !( array = calloc( 256, sizeof(RECD) ) ) )
    {puts( "Not enough memory" ); exit(EOF); }

    count = get_dat( get_names( argv[1] ) );
    realloc( array, count * sizeof(RECD) );
    report( count );
    free( array );
}
/*=*=*=*=*=*=*=*/
int get_names( drv_nam )
char *drv_nam;
{   RECD             *this;
```

```
        DIR             *dirp;
        struct dirent   *dptr;

        if ( !( dirp = opendir( drv_nam ) ) )
        {       puts( "opendir() error" ); exit(1); }

        this = array;

        while ( dptr = readdir( dirp ) )
        {       if ( !( strfind( dptr->d_name, "/SYS", 0 )))
                {       strcpy(  this->name,  dptr->d_name );
                        ++this; } }

        closedir( dirp );
        return this - array;
    }
    /*=*=*=*=*=*=*/
    int get_dat( count )
    int  count;
    {   RECD                    *this, *limit;
        struct stat             sbuf;
        register int            kill = 0;

        this = array;
        limit = array + count;

        while ( this < limit )
        {       if ( !( stat( this->name, &sbuf ) ) )
                {       this->size = sbuf.st_size;
                        this->date = sbuf.st_mtime; }
                else
                {       this->date = 0x7fffffffL;
                        ++kill; }
                ++this;
        }
        qsort( array, count, sizeof(RECD), compare );
        return count - kill;
    }
    /*=*=*=*=*=*=*/
    int compare( a, b )
    RECD        *a, *b;
    {   if ( a->date > b->date )
                return 1;
        else if ( a->date < b->date )
                return -1;
        else
                return strcmp( a->name, b->name );
    }
    /*=*=*=*=*=*=*/
    void report( count )
    int count;
    {   struct tm               *tbuf;
        RECD                    *this, *limit;
        register int        c;
        long                    tl_bytes = 0L;

        this = array;
        limit = array + count;

        while ( this < limit )
                printf( "%s\t", this->name );
                if ( strlen( this->name ) < 8 )
                        putchar( '\t' );
                tbuf = localtime( &this->date );
                printf( "%02d/%02d/%02d  %02d:%02d",
                                tbuf->tm_mon + 1,
                                tbuf->tm_mday,
                                tbuf->tm_year,
                                tbuf->tm_hour,
                                tbuf->tm_min );
                printf( "%9ld bytes\n", this->size );
                tl_bytes += this->size;
                ++this;
                if ( inkey() == CTRL_S )
                        do c = inkey();
                        while ( !c || c == CTRL_S );
        }
        printf( "\n%3d files, %ld total bytes\n",
                                count, tl_bytes );
    }
```

Prog11.c makes fairly extensive use of allocated memory; so it's probably a good time to explore that subject a little deeper.

Pro-MC provides a number of functions to enable your programs to allocate, deallocate and use RAM for any purpose.

The principal ones are:

alloc(size) -- this NON-STANDARD function reserves "size" (up to 65,535) bytes of RAM.

malloc(size) -- reserves a zeroed block "size" (up to 65,535) bytes in size and, while the function name is standard, what it does is NON-STANDARD.

calloc(number, size) -- reserves a zeroed block of RAM large enough to hold "number" of elements of "size" bytes each. The product of "number" and "size" must not exceed 65,535. For example, calloc(50, sizeof(int)) would null out and reserve 100 bytes of RAM -- enough to hold 50 two-byte short ints.

realloc(ptr, size) can be used to change (increase or decrease) the number of bytes in the block pointed to by "ptr." The 65,535 limit applies.

freemem() is a NON-STANDARD function which returns the size in bytes of the largest block of memory which can be allocated by your program.

free(ptr) releases (deallocates) memory previously allocated by alloc(), malloc(), calloc() or realloc(), and pointed to by "ptr."

There are a couple of others [brk() and sbrk()] but I don't recommend them.

In standard C, malloc() doesn't zero the allocated memory block, and alloc() doesn't even exist.

Alloc() is one of Pro-MC's holdovers from the old "Small C" days (before full C compilers for the Z80 microprocessor or the TRS-80 microcomputer were available) to let Small C code be upward compatible.

Thus, if you're using a MeSs-DOS or other computer to compile these example programs, you would use malloc() whenever you see alloc() used in this series. Also, be aware that calloc() is the only STANDARD way to obtain a zeroed memory block.

Prog11.c makes much more use of standard header files than we've ever done before. If you'll look at the docs for each, you'll see:

dirent.h contains definitions for the directory I/O functions;

stat.h defines a struct for storing certain information about a file [see docs for fstat()];

time.h defines a struct for storing data about the time (e.g., month, day, minute, etc.); and

ustat.h defines a struct for storing data about a disk drive (e.g., free space, number of directory slots in use, etc.).

It would be a good idea to list each of these header files out to your printer so you can see what data they contain, and how the prog11.c accesses and uses that data.

After some defines and a prototype for report(), we declare "array" to be a pointer-to-struct of type "record." This type of struct contains members which can store the size of a file in bytes, the UNIX time the file was created or last updated, and a 15-char array to hold the file name.

Notice no memory has been allocated to hold the actual data. The only thing that has been declared is a pointer to a struct of type "record." This declaration tells the compiler how to handle the data which exists at the RAM address which will be assigned to "array" later in the program.

Inside main(), we first check the command line to make sure there are two arguments, and the second argument starts with a colon character. If either of these tests are FALSE, the user is informed via the "usage" message.

Now we declare and use a variable which is unique to a specific block of code within a function.

Braces define the block in which "ubuf" exists. No code outside this block -- not even other code in main() -- can access "ubuf," which is created on the stack when the block is entered and which ceases to exist after the last line of code in the block is executed.

The reason "ubuf" is created this way instead of as part of main() is because the original program needed all the memory it could get for a large disk buffer. Because we're still in main(), this is the only way the memory used by "ubuf" can be released after "ubuf" is no longer needed.

Yes, I could have used alloc(), but this is simpler and requires less code.

What actually happens is that the code in this block first reserves enough stack space to hold a struct of type "ustat." Then it passes the drive number and address of "ubuf" to the ustat() function, and checks ustat()'s return code.

If there is an error (e.g., the specified drive doesn't exist), ustat() returns EOF (-1). This non-zero value causes the "if" statement to evaluate to TRUE; so the error will be reported and the program aborted with an error message.

If there is no error, then we've come to the end of the code block; so "ubuf" simply ceases to exist.

The purpose of this call to ustat() is to make sure the drive specified on the command line is valid and ready.

Next, calloc() is used to allocate a block of zeroed RAM large enough to hold 256 structs of type "record." If the requested RAM cannot be allocated, the program is aborted with an error message.

Why 256? That's the maximum number of records TRS/LS-DOS can have in a directory file. Other platforms may require a larger number.

If calloc() is successful, it returns a pointer to the allocated block which is then loaded into the global variable "array."

It's extremely important for you to understand that "array" now points to the first element of an array of 256 structs of type "record." There's no data in the structs yet, but they're there.

If you don't understand this, go back and look at the way "array" was declared, and at how "array" received it's value (namely a pointer to a block of allocated RAM).

In actuality, not all the elements in this array will be used. Let's follow the call to get_names() to find out why.

The argument passed to get_names() is a pointer to the ASCII string containing the drive number.

Inside get_names() we declare "this" to be a pointer to struct of type "record;" "dirp" to be a directory file pointer,; and "dptr" to be a pointer to struct of type "dirent."

Now the directory file on the specified drive is opened. If it cannot be opened, opendir() will return the NULL pointer; which will cause program termination with an error message.

Then "this" is pointed to the first element of the "array" array, and each directory record is read in sequence.

In each case, the file name from the directory record is checked via the standard strfind() function to see if contains the string "/SYS." If so, the record is skipped.

If this is a non-/SYS file, it's name is copied to the "name" member of the "array" element pointed to by "this," and "this" is incremented.

In other words, /SYS files are ignored.

When all directory records have been read, the readdir() function returns the NULL pointer, which causes the "while" loop to be exited, and the directory file is closed.

The next statement -- "return this - array" -- utilizes the automatic scaling of variables in C.

Remember, both "this" and "array" are pointers to struct of type "record." Thus, after "array" is subtracted from "this," the initial result will be the number of bytes of RAM taken up by the structs of type "record" which were actually used.

But this will automatically be divided by "sizeof (struct record);" so the return value will be the number of non-/SYS file names read from the directory.

For example, suppose 20 file names were stored. Because each struct of type "record" is 23 bytes, "this" will contain a RAM address which is 20 x 23 = 460 higher than the RAM address stored in "array."

Because of scaling, this difference (460) will automatically be divided by 23 [sizeof(struct record)], yielding a final result of 20, the value which will be returned by the get_names() function.

Thus, the get_names() function does two things: it stores all non-/SYS file names in the "name" members of elements of the "array" array of structs of type "record;" and it returns the number of the file names it has stored.

Since every directory must contain at least two /SYS files (BOOT/SYS and DIR/SYS), we will never use all 256 elements of "array." The most we could possibly ever use is 254.

Now, back in main(), get_name()'s return value is passed to get_dat(); so let's follow that call.

Here we declare two pointers to struct of type record, a struct of type "stat" and we declare and initialize the integer "kill."

"This" is then initialized to point to the first element of "array," and "limit" is made to point to the first byte of the "array" element AFTER the last element actually used.

Thus, the "while" loop will step through only the elements of "array" which are actually being used.

In this loop, stat() is used to obtain information about each file. Each time stat() is successful the "size" and "date" members of the struct pointed to by "this" will be assigned data which stat() put in "sbuf" -- namely the file size and last modification time and date.

If stat() returns an error, the file name is invalid; so the highest possible date and time is loaded into "this->date" and the "kill" counter incremented.

After all the data has been obtained, the array is sorted. If you'll look at compare(), you'll see the sorting is done by time/date in ascending order, and alphabetically within identical time/dates.

This causes any invalid array entries (in the extremely unlikely event there are any) to be sorted to the top.

Back in get_dat(), the number of valid entries is returned and, back in main(), this value is loaded into the variable "count."

Now the actual amount of RAM used by valid "array" entries is calculated by multiplying the number of valid entries times sizeof(struct record).

This value, along with the pointer to the block, is passed to realloc(), which reduces the amount of allocated RAM to what is actually needed, thereby freeing the RAM not needed or used by "array." Again, this was done because the original program needed all the free memory it could get.

Now we call report(), which provides a listing, in date and time order, of all non-/SYS files on the specified drive.

Again, we use "this" and "limit" to govern a "while" loop, and declare "tbuf" to be a struct of type "tm" and declare and initialize a long int variable named "tl_bytes."

The first thing done in this loop is to send the file name to stdout (the monitor screen). If the name is too short, an additional tab character is also sent to make the listing line up properly.

Next, the last-modification date from the "array" element pointed to by "this" is sent to localtime().

This standard function breaks down the UNIX time stored in "this->date" to the year, month, day, hour, minute, day of the week, etc., and stores this data in struct of type "tm" pointed to by "tbuf."

Now data from "tbuf" is displayed in a formatted manner, namely "MM/DD/YY HH:MM." You chaps in Europe, Australia and most other places may wish to change the date format to the proper "DD/MM/YY" by swapping the two references to "tm_mday" and "tm_mon." If you live in the former Soviet bloc, you may wish to put the references to the "YY/MM/DD" order you're more familiar with.

Next the number of bytes in the file is displayed and accumulated in "tl_bytes," and "this" incremented to point to the next element of"array."

Now we use the NON-STANDARD inkey() function to see if any keys are being pressed. If so, the key value returned by inkey() is checked to see if it's the <CTRL><S> combination, which is the standard "stop" combination.

TRS-80 users are accustomed to using <SHIFT><@>; so if you want you can change CTRL_S to PAUSE, and replace the #define line for CTRL_S to "#define PAUSE 0x60." Personally, I find <CTRL><S> easier to use on my non-clustered-arrow-keys keyboard, but to each his own.

Anyway, if <CTRL><S> is detected, the following "do" loop keeps scanning the keyboard until some key other than the <CTRL><S> combination is detected.

After all files have been listed, a final report line containing the number of files and their total number of bytes is displayed before the function returns to main().

Finally, back in main(), the calloc()'ed and then realloc()'ed block of memory pointed to by "array" is released, and return made to TRS/LS-DOS Ready.

Type in and compile prog11.c now, and see how it runs. Save the source file, because next time the full bydate.c program will be presented to show how block disk I/O is performed; and some of the code in prog11.c will be reused.

Now I am now issuing a call for "help" from those readers following this series.

If you have a short or medium size BASIC program you like a lot, and want to have it converted to C to run faster and/or better, please send it to me. These programs must be something you've written yourself, or otherwise be clearly in the public domain -- not copyrighted commercial programs.

Also, if you have an idea for a program -- something you want your Model 4 to do that it can't do now -- perhaps something you've seen other computers do that you'd like to be able to do on the Model 4 -- please write me with the idea.

This will not only help me produce articles to help you and others learn C, but could also result in useful new programs to help keep the Model 4 alive. After all, if it's something you want, the chances are others will want it, too.

A Few Trouble Shooting Efforts

by Roy T. Beck

Some of you may know that I have some degree of expertise on TRS hard drives. Let me tell you about a case where a little knowledge was dangerous.

Recently, (Allen S.) a TRS Model 4 user acquired some hard drives, and wanted to get them operating. He contacted Stan Slater, who gave my phone number to Allen. In the course of time, we got together by phone, and I attempted to help him. Initially, he wanted to operate his 5 Meg master drive. I advised him on this, especially how to connect the infamous "three wires" RS put into their master drives. Those three wires were soldered directly to points on the pc board of the bubble, and the locations varied with the make and model of the bubble. He got the master going OK, and then wanted to connect up his slave drive. This is where my tunnel vision came into action. Allen said "slave drive", and I immediately assumed he was referring to the slave drive which RS sold to accompany their 5 Meg master hard drive. That slave contained a power supply, a bubble, a fan, a small pc board containing one and only one chip, and some lights on the front. It also had a trick circuit consisting of a relay with a 12 V dc coil which was energized by 12 volts from the master drive. This relay then turned on the 110 V ac for the power supply in the slave, thus avoiding the need to turn on a separate power switch on the slave.

Just to further complicate matters, RS added "four wires" which were soldered to points on the pc board of the bubble in the slave, three of which corresponded to the "three wires" in the master, and the fourth wire was involved with the 12 V dc relay. RS didn't bother to match the colors of the wires in the slave with the master, so corresponding functions had different colored wires in the master and the slave box.

In attempting to assist Allen, I told him to look for the little pc board and the "four wires". He swore up and down that there was no such pc board in his slave, and I concluded someone must have removed the little board and disposed of it. At the time of our communications about the slave, I was located in a hotel room in Richmond, California, and had no access to my notes, manuals, etc. This also complicated communications between us.

Eventually, it occurred to me to ask him to verify the Cat No on the bottom of the slave drive to be sure it was the one that was designed to work with the 5 Meg drive. SURPRISE! The catalog number, (25-1041) was totally wrong for what I expected, but fortunately I did recognize it for what it was. At some time in the past, RS built up some outboard hard drive packages for use with their 1000 series machines, but they maintained enough compatibility that these same outboard drives would also function with Models I, III and 4/4P machines. The outboard unit consisted of two small boxes which stacked on each other. Each box was of the height and width to accommodate a half height hard drive, and contained a little power supply in the back end of each one. One box contained only a hard drive controller, and was identified as Catalog number 26-1138, and the other housed only a half height bubble, the 25-1041. It required one of each box to make a functioning package, and, while I have worked with these units, it never occurred to me to think of the box containing the bubble as a "slave". But that was what Allen had acquired, and not knowing otherwise, he referred to the unit as a slave, and expected to make it work as a slave to the old 5 Meg box.

Now, knowing what Allen has, I can assist him in making the "slave" drive actually function as a slave to his 5 Meg unit. I have not done this before, but having enough manuals, wiring diagrams and whatnot about, I believe he can indeed make the small outboard box function correctly as a slave to the 5 Meg box.

Once I backed up and took a broader view of the problem, all came clear, and I expect success the next time Allen and I talk. As a matter of fact, I am out of town at the moment, being located this week in Mendocino County, northern California.

A little later: Well, I heard from Allen again, and wouldn't you know, he still had a problem! When I next spoke to him, he had smoked a resistor in his hard drive controller box. It all goes back to Radio Shack's infamous three wires monkey business, in which they soldered wires directly to points on the bubbles in both their master and slave drive boxes. Their arrangements were OK and logical as long as only the original Tandon drives

were used in the original boxes. Their circuits worked because they were able to make use of unused pins on the 34/C and 20/C cables to the bubbles. Allen got in trouble because he had installed a 20 meg bubble of unknown ancestry in the small slave box. Unfortunately, the R/S design applied 12 V dc to a line which was evidently grounded in the strange bubble, leading to serious overheating of the 15 ohm resistor which was in series with the 12 V. That worked fine in the old R/S slave box, but Allen wasn't using one of the old R/S slave boxes, he was using the 25-1041 box which was intended to work with the 26-1138 hard drive controller, and that "slave" box was not equipped with the 12 volt relay. Probably the line with 12 volts on it was grounded in the 25-1041. To solve Allen's problem, I had him disconnect the 15 ohm resistor in the master which fed the 12 V, as his setup didn't require 12 V. Allen and his wife attempted to unsolder the 15 ohm resistor, but between them they broke it in two. Oh well, the purpose was accomplished, if crudely!

This step allowed Allen to operate the 25-1041 box as a slave to his Model 4 and its master HD box. But now the "slave box" reported it was write protected, and could not accept formatting. Oh, yeah, the three wires again! Allen was using a master box with the old, large HDC board in it, and without the proper connection of the three wires in the slave, it believes the slave bubble is write protected. I thought a bit and told Allen how to install a resistor in the master which would pull the write protect line for the slave high, which is the non-write protected condition. I also had him cut a trace so no 5V dc would be fed back into the strange bubble in the "slave" box. Of course, these instructions required me to tell him where to solder the resistor. I had selected two pins on a buffer chip, because the 5V was available on pin 20, and pin 15 is the write protect circuit for the slave drive. The only problem with my instructions was that Allen, it turned out, did not know how to count pins on a chip. GRRR. Anyway after a while, I got him to draw a sketch of a chip and locate the two pins on the sketch. I hope he soldered the resistor to the correct pins. Fortunately, TTL logic is fairly forgiving, and I don't think he can hurt anything. We'll see.

My Experience as a Beta Tester

Recently I had the pleasure of being a beta tester for a hardware product, and it was most interesting. Actually, the beta testing aspect of it was sort of inadvertent; the sponsor of the product actually wanted me to simply review the new instruction manual for his product, which was in the process of being rewritten. In order to do the review, he of course gave me a complete hardware package to work with, as I had to understand how the equipment functioned in order to give the manual a careful once over.

The device is a printer sharing network which allows several computers to share access with several printers. In fact, the network can be expanded to a total of 32 computers plus printers, although the system would probably suffer from severe queuing delays if that many machines were actually connected together. Functionally, the system resembles a SCSI bus, in that all the devices share a common bus, are each identified with an unique ID, and can talk with any other device on the bus. A significant difference is that the bus consists of ordinary 4 wire telephone cable with RJ-ll modular plugs on each end of each piece of bus. The system can have a total length of 1200 feet, and can move data at 30kbytes per second.

Because the system can talk between devices, and the computers are devices, the system can readily move files between computers, as well as allow any computer to use any printer. A typical office application might include printers dedicated to specific tasks. For example, one printer might do nothing but address envelopes, another might be a dot matrix assigned to printing multi-part invoices, and a third might use letterhead paper for correspondence. I am sure you could think of other dedicated applications. Within the 32 device limit, a number of computers in different offices could all share the printers as needed.

To continue with my adventure. I took the basic hardware and one expansion package with me when I went to northern California last week, and my total of functioning equipment consisted of a 486 machine and (believe it or not), an Epson MX-80 F/T, the oldest Epson in existence! My first task was to read the draft of the new manual which had been supplied to me. I went over it with a red pencil, and did some significant marking-up. In the process, I acquired a fair knowledge of what the system was supposed to do. I carefully avoided reading the old manual which came with the system, as I wanted to read the new manual from the point of view of a new, ignorant purchaser of the system.

After reading the new manual, I ventured to connect up the hardware, following the instructions in the new manual. I then powered up the system, and immediately knew I was in trouble! Each module of the system is equipped with an LED,

whose blinking pattern corresponds to a status. Only one module would blink at me at all, and its signal indicated it wasn't seeing any communication with the remaining parts of the system, which in this case was simply a length of wire and the module attached to the computer. Oh, dear....

Since the 486 and the Epson MX-80 had been talking (slowly) to each other before, I reconnected the MX-80 via its parallel cable to verify it and the 486 still functioned. Yup, all OK in that mode.

Since the hardware loaned to me in effect had spare parts, I swapped computer modules, I swapped cables, and I swapped printer modules since he had loaned me an extra printer module. No soap, no communications between any two modules.

At this point, I gave up and called the customer service tech and got a very savvy man on the line. He coached me through swapping of bits and pieces again, to verify the trouble. He then asked me if I had any more cables available. It happened, there was one more in the extra printer package which I had not tried. We tried that, and Yeah!, everything came to life. To shorten the story, the trouble was that the two modular plugs on each of the first two cables were reversed relative to each other. Think of it as pin #1 of one plug was connected to pin #4 of the other plug, and vice versa. In effect, it was connected up as a sort of a null modem, whereas the hardware requires pin 1 to be that all through the system, with #2, #3, and #4 wired in similar fashion. By luck, the third cable had been constructed correctly, and I could make the system work. In fact, I am using it here at the moment to drive my Laser 4. Now that everything in the hardware is working satisfactorily, I can get back to editing the new manual.

I have a rule which I apply in all troubleshooting. Almost always, there is one, lone cause of the trouble, which when identified, logically explains all the observed symptoms and troubles. Find that one glitch, and you usually have the problem licked. Of course, in some cases the original trouble causes progressive failure in the system. This makes the trouble shooting more difficult, but the principle still applies. Look for that one, original trouble, and you stand a good chance of solving your problems.