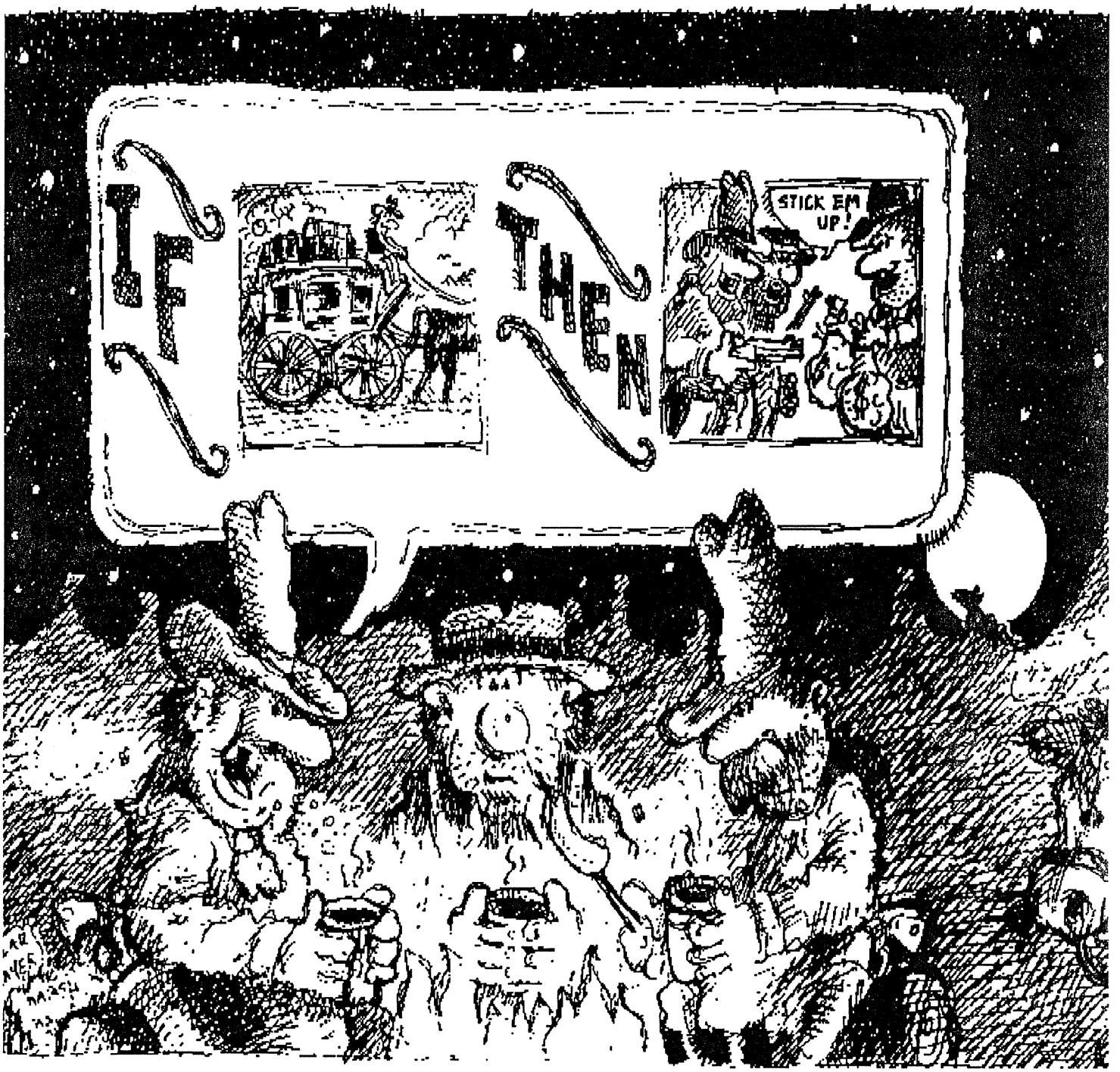
TRSTimes

Volume 6. No. 3. - May/Jun 1993 - $4.00



THE EARLY HACKERS

LITTLE ORPHAN EIGHTY

Many of our subscribers have told us that we should feature more game oriented articles and programs. Now, while I don't want TRSTimes to become an incarnation of Softside from long ago, I certainly do agree that there is nothing wrong with having fun while computing. Therefore, we will, from time to time, focus on the game playing abilities of our favorite machines.

This issue should satisfy our funsters; we feature a new adventure game for the Model 4, called 'The case at KAXL'. It is written by Doug Rogers, and it should provide hours of fun and frustration.

I am really excited about our other fun installment, a new series by Daniel Myers, called 'Beat the game'. Danny is an adventure game player extraordinaire; that is, he has solved most of the TRS-80 adventure games and he is going to share the secrets with us. He begins by going through Adventure International's 'the Curse of Crowley Manor' step by step to the conclusion. I played this game some years ago and, believe me, it is hard. I never did finsh, but I am going to take time to go through it again. Now, I know that some of you will say that it is no fun to cheat by having a solution sheet! Maybe so, but you don't have to use it. The complete solution is now available to you should you feel you need it.

The 'Crowley Manor' article is only the beginning. Danny will share the intricacies of the Zork trilogy, Deadline, Starcross, as well as many of the other famous adventures from Infocom; maybe he will even tackle a few in the equally famous Scott Adams series. I really look forward to this, and I think many of you will dust off the old disks and give the games another whirl, this time armed with the heavy ammunition.

I have waited and waited for someone to submit an article or two on Visicalc. Well, they didn't come, so to start things off, this issue features a Visicalc Reference Guide from the vast vaults of TRSTimes. Hopefully, this will be of interest to the newcomers to the TRS-80 world, and maybe it will nudge some of you expert Visicalc old-timers into sharing your tricks with the rest of us. How about a mini-tutorial? I would love for someone to teach me the finer points of spreadsheeting.

This month a strange thing happened. I was replying to a letter from a reader who had made some excellent programming suggestions. The reply had turned into an article covering some of the suggestions in detail, when the mailman delivered Chris Fara's 'Programming Tidbits' intallment for this issue. Boy, it was scary! He was, out of nowhere, covering the same subject that I was writing about. Welcome to the Twilight Zone!

Actually, evrything turned out well. Chris was covering Model III, and I was writing about Model 4. Also, my article dealt with boxes, while Chris went one step further and created windows. When you read 'Programming Tidbits' and 'Heavyweight Boxing', you'll see what I am talking about. Two people, same subject, two different approaches..... this is what makes computers fun.

DEBUG department.

I managed to mess up Chris Fara's 'Programming Tidbits' installment from our last issue (6.2). Somehow, I managed to lose several minus signs. Here are the corrections:

Page 25, column 1, last paragraph - 3rd & 4th lines read: "BASIC takes a comparison and generates a number **1**" - the number should be **-1.**

Page 25, column 1, last paragraph - 4th line reads: " In binary notation **1** is" - the number should be **-1.**

Page 25, column 1, last paragraph, 8th line reads: "its designers decided that not only **1**" - the number should be **-1.**

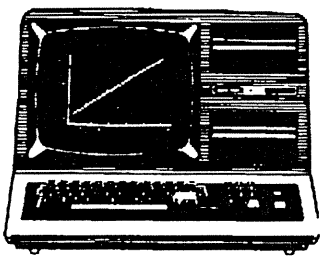
Page 25, column 2, 7th paragraph, 5th line reads: "Any value other than **1**" - the number shoulld be **-1.**

Page 25, column 2, 7th paragraph, 10th line reads: "expression with value **1**" - the number should be **-1.**

I still don't know how this happened. But no excuses, I should have caught while proofreading. Sorry.

Before closing this column, I would like to extend my thanks to all the good people who contributed to this issue, Doug Rogers, Chris Fara, Fred Bennett, Frankie Tipps, Roy Beck and Danny Myers. The TRS-80 community is lucky to have you around - without you we'd have a mighty thin issue. Thanks guys, keep on writing.

And now.....Welcome to TRSTimes 6.3.



TRSTimes magazine

Volume 6. No. 3 - May/Jun 1993

PUBLISHER-EDITOR
Lance Wolstrup
CONTRIBUTING EDITORS
Roy Beck
Dr. Allen Jacobs
Dr. Michael W. Ecker
TECHNICAL ASSISTANCE
San Gabriel Tandy Users Group
Valley TRS-80 Users Group
Valley Hackers' TRS-80 Users Group

TRSTimes magazine is published bi-monthly by TRSTimes Publications. 5721 Topanga Canyon Blvd, Suite 4. Woodland Hills, CA. 91364. (818) 716-7154.
Publication months are January, March, May, July, September and November.

1993 subscription rates (6 issues):
UNITED STATES & CANADA: $20.00 (U.S. currency)
EUROPE, CENTRAL & SOUTH AMERICA: $24.00 for surface mail or $31.00 for air mail.
(U.S. currency only)
ASIA, AUSTRALIA & NEW ZEALAND: $26.00 for surface mail or $34.00 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 2
Editorial

THE MAIL ROOM 4
Reader mail

LASER PRINTING WITH MODEL 4 5
Lance Wolstrup

THE CASE AT KAXL 6
Doug Rogers

PROGRAMMING TIDBITS 11
Chris Fara

HINTS & TIPS 14
Bennett, Tipps

ASCII REVISITED 17
Roy T. Beck

BEAT THE GAME 19
Daniel Myers

HEAVYWEIGHT BOXING 22
Lance Wolstrup

VISICALC REFERENCE GUIDE 26
TRSTimes vault

THE MAIL ROOM

PEEKING & POKING

I really appreciated the last issue of TRSTimes (6.1), especially the programming tips. Felt right at home poking on the Model IV, the one thing I missed from the Model III. Hopefully every issue will have programming tips and ideas.

Maybe you can do a future article on fast draw (lines for boxes, etc). Maybe a window type box (like the LeScript Help Screen) which could be placed anywhere on the screen, and maybe even restore the screen! A highlighted cursor Menu like the Little Brother moving cursor!

Mickey Mepham
Charles City, VA

I am glad that you like the programming aspects of TRSTimes. Peeking and poking on the Model 4 is one of my favorite pastimes. Elsewhere in this issue you will find two articles dealing with the subject you requested. Hope you enjoy them.

Ed.

GAMES

I really like the puzzles you write in TRSTimes. They are challenging, fun, and interesting for learning programming. I play them often, just to get away from it all. Write more, please!!

Anthony B. Mizzell
Chesapeake, VA

I have always enjoyed computer games, both playing them and writing them. We will feature many more games in the upcoming issues - this issue brings a new adventure game written by Doug Rogers, as well as a new series by Danny Myers calles 'Beat the Game' which will walk you through different Adventure International or Infocom adventure games. I am personally looking forward to time off to play some of the ones that have frustrated me in the past. Hope you enjoy the features.

Ed.

BOOT5 & FASTTERM

I read with interest your latest issue concerning BOOT5/CMD. You did not say that the original program is available from MISOSYS (PO Box 239, Sterling, VA 20167) on their Disk Notes 5.4 for $10.00 + $2.00 S&H. The patches to allow operation with LDOS 5.3.1 are published both in TRSTimes, Jan/Feb 1992,p.11 and the Misosys Quarterly, Spring 1992 (vl.III) p. 18. I am using the patched programs on both a 5-meg Tandy and a 20-meg Aerocomp Hard Disk to allow the last drive of each to operate in the Model 3, LDOS 5.3.1 mode. It even loads ROM into my 4P in no time at all. I leave my LS-DOS startup disk in, as the only way to get back to Model 4 mode is to hit the reset button. I recommend the program to anyone using a hard disk in both 3 and 4 modes.

We all don't have to agree on favorite programs, do we? I have been using FastTerm for years and feel Mel went too far in dropping multistroke keys (similar to DOS KSM) in favor of "Script" (similar to a /JCL file). Of my time on Dow Jones News Service last year, at least half was spent trying to recognize the DJNS log-on logo, which is:

ENTER PASSWORD
WWWWWWWWWWWWWW
MMMMMMMMMMMMMM
@@@@@@@@@@@@@@ (followed, I think, by hex 11 20 20 20 13 11)

Also, line noise often makes the whole procedure "bug out". I prefer his version 14.9 of the original FastTerm. <Clear> 1 to 4 sends my terminal type through password via Tymnet to Compuserve; <clear> 5 to 8 does the same for DJNS. Any line noise and I can just send that particular KSM (Key Stroke Multiply) again. Mel does allow DOS KSM to feed through FastTerm, but I need his convenient menu of 5 pages of 9 keys each, unfortunately replaced by "Script" in the latest versions.

John P. Jones
Fairmont, WV

You are correct - I did omit the information about Misosys. This, however, was not done purposely, as TRSTimes is of the opinion that it is in the interest of all TRS-80 owners to keep Roy Soltoff and Misosys in business. Your letter have now corrected the omission.

We all have different interests and tastes and this, of couse, is especially true when it comes to our choice of computer and what we use it for. You certainly have the right to prefer version 14.9 of FastTerm, and since it serves your needs admirably, it would make no sense at all to use a later version that does not suit you. As a programmer at heart, I like the new version so I can tinker with the script capability. Mel Patrick has written many fine programs, but I think that FastTerm (any version) is probably his best effort.

Ed.

LASER PRINTING WITH MODEL 4

by Lance Wolstrup

The hottest selling printers today are of the laser type. Many of my friends have recently added Hewlett Packard's latest LaserJet, the Series 4, to their PC clones. I have had a HP series II for several years and, I have to admit, it has also been hooked up to my 'Big Blue' compatible computer. For some reason, I just never tried to hook it up to my Model 4!

Well, this changed a few days ago. I bought a device, known as an A/B switch, which allows two computers to use the same printer. I connected the HP Series II to the output port, the Model 4 to the A input port, and the PC to the B input port. With this done, I turned on the Model 4 and the HP, turned the switch to the A port, and send a directory of drive :0 to the printer by DIR :0 (P)

Everything worked just dandy...except for two itsy-bitsy problems; the print-out didn't eject, and when I ejected the page manually, my directory was printed and overprinted all on one line; the old TRS-80 story - all kinds of carriage returns, but no line feeds!

Other printers solves this problem with a click of a dip switch, but the HP (and laser printers in general) has no dip switches; instead, all changes are handled via software. There are two ways that the 'no linefeed' problem can be solved. First, we can force the Model 4 to generate the line feeds by invoking the FORMS/FLT as follows:

```
SET *FF TO FORMS/FLT
FILTER *PR *FF
FORMS (ADDLF,CHARS=80,LINES=60,FFHARD)
```

The other method is to write a program that will make the HP printer responsible for the line feeds. To do this, the laser printer needs to receive five bytes of code:

```
27, 38, 107, 49, 71
```

This code could be transmitted from Basic, for example. The following line will do the job:

```
10 LPRINT CHR$(27);CHR$(38);CHR$(107);
CHR$(49);CHR$(71);
```

However, it is too much trouble to load Basic and execute the program. The better, and more elegant, way to send the code to the printer is to write a machine language program that can be executed directly from the DOS command line.

HP/ASM, listed below, is the assembly language source code which will assemble to HP/CMD. It first checks printer port F8H to see if the printer is available. If bits 4 & 5 are set (1), then the laser printer (or any other printer) is recognized. Bits 0, 1, 2, and 3 are of no importance, so they are masked out with the AND F0H instruction. Bit 6 is set if it is determined that the printer is out of paper, while bit 7 is set if the printer is busy. Thus, if the byte value, after the mask, is 30H we can proceed. This information, incidentally, is readily available in the Model 4/4P Technical Reference Manual.

Having determined the readiness of the printer, we send the code. Note that we send two extra bytes before the aforementioned five bytes to add the line feeds. These two bytes are 27 and 69. They simply cause the HP to reset to the default values.

```
;HP/ASM
;utility to convert CR to CR+LF
;on HP laser jets & desk jets
;
          ORG   3000H
START     IN    A,(0F8H)    ;read printer status
          AND   0F0H        ;mask out lower 4 bits
          CP    30H         ;check if printer available
          JR    NZ,NOAVAIL  ;jump if not
          LD    HL,CRLF     ;point to printer data
          LD    A,14        ;@print svc
          RST   40          ;send crlf code to printer
          LD    HL,MSG1     ;point to msg1
EXIT      LD    A,10        ;@dsply svc
          RST   40          ;display msg
          RET               ;return to dos
NOAVAIL   LD    HL,MSG2     ;point to msg2
          JR    EXIT
;
CRLF      DEFB  27          ;reset
          DEFB  69          ;hp printer
;
          DEFB  27          ;this sequence
          DEFB  38          ;of code sets
          DEFB  107         ;lf=lf
          DEFB  49          ;cr=cr+lf
          DEFB  71          ;ff=ff
          DEFB  3
MSG1      DEFM  'HP printer now initialized for your'
          DEFM  ' TRS-80'
          DEFB  13
MSG2      DEFM  'HP printer is not available'
          DEFB  13
          END   START
```

The second problem, that of the sheet not being ejected from the printer, is easily solved - simply use the Model 4 library command, TOF. Yes, in his infinite wisdom, Roy Soltoff knew that someday we'd own a laser printer!

THE CASE AT KAXL

an adventure game

Model 4 - Basic

By Doug Rogers

A few minutes ago, you got a frantic phone call from your buddy Mike Mouth, the Disk Jockey. He was cut off in mid sentence. You have just arrived at the Radio Station to investigate...

If you have never been in a Radio Station before, we suggest that you look around carefully. There is a lot of equipment in the place, and some of it could be of use to you.

Note also that Radio Station managers and Program Directors are notorious memo writers and note posters. You can learn a lot (that you will need to know) about running a radio station just by keeping your eyes and ears open.

Some words that should be useful to you are: LISTEN, GET, PUT, READ, HELP, EXAMINE, USE, INVENTORY (tells what you are holding), SAVE, LOAD, WAIT, and any other VERB-NOUN combination that seems appropriate. If one combination doesn't work, try another. The program has a large vocabulary.

RADIO/BAS

```
10 DIM D$(3),L$(28),L(28,3),O(60),O$(60),NO$(60),
VB$(17)
20 CLS:PRINT CHR$(15);:Q=1:PRINT CHR$(23):
PRINT@662,"THE CASE AT KAXL"
30 PRINT@1056,"Written by Doug Rogers"
40 N$="RADIO/SAV"
50 RESTORE:
X=0:L=22:Z1=0:Z=0:M=0:H=0:NO=60:NV=17:
NN=NO:AU=120:CC=0:BF=0:
FOR X= 0 TO 3:READ D$(X):NEXT:
FOR X=1 TO 28:
READ L$(X),L(X,0),L(X,1),L(X,2),L(X,3):NEXT
60 FOR X=1 TO NO:READ O$(X),O(X):NEXT:
FOR X=1 TO NV:READ VB$(X):NEXT:
FOR X=1 TO NN:READ NO$(X):NEXT:
M$="00":H$="12":ID=1:
L$="It's locked...":OP$="It's open...":CP=0
70 CLS
80 IF CC<>0 THEN 510
ELSE AU=AU-1:IF AU<-6 THEN PH=1:
IF AU<-10 THEN CC=1
90 PRINT@(0,0),"You are ";L$(L);CHR$(31):
PRINT"You see: ";:FL=0:
FOR X=1 TO NO:IF O(X)=L THEN 100 ELSE 120
100 IF POS(0)+LEN(O$(X))+3>80 THEN
PRINT STRING$(80-POS(0),32);:PRINT"        ";
110 PRINT O$(X);". ";:FL=1
120 NEXT:
IF FL=0 THEN PRINT"NOTHING";
STRING$(80-POS(0),32);
ELSE PRINT STRING$(80-POS(0),32);
130 IF SF=1 THEN SF=0:
PRINT"There's a wisp of smoke in the air...";:
IF G=0 THEN G=1:
PRINT" moving, as if someone just ran out of the
room...";STRING$(80-POS(0),32);:
ELSE PRINT STRING$(80-POS(0),32);
140 FL=0:FOR X=0 TO 3:IF L(L,X)<>0 THEN FL=1
150 NEXT:
IF FL=0 THEN 210 ELSE PRINT"You can go: ";:
FOR X=0 TO 3:IF L(L,X)<>0 THEN PRINT D$(X);
160 NEXT:PRINT STRING$(80-POS(0),32);:
M=VAL(M$):M=M+1:
IF M>59 THEN M$="00" ELSE M$=STR$(M):
IF LEN(M$)<3 THEN M$="0"+RIGHT$(M$,1)
ELSE M$=RIGHT$(M$,2)
170 H=VAL(H$):IF M>59 THEN H=H+1:ID=0:
IF H>12 THEN H=1
180 H$=RIGHT$(STR$(H),2):
PRINT"The time is "H$;":";M$;STRING$(80-POS(0),32);:
IF ID=0 AND M=5 THEN PH=1
190 IF CP=0 THEN IF PH=1 THEN PRINT"There is a
phone ringing somewhere...";STRING$(80-POS(0),32);:
IF M=13 THEN CC=1
200 IF H=4 THEN GOSUB 540
ELSE IF H=5 THEN CC=1
210 PRINT STRING$(79,"-")
220 AN$="":VB=0:NB=0:PRINT"Your command ";:
HP=POS(0):VP=ROW(0):
PRINT STRING$(80-POS(0),32):GOSUB 1890:
PRINT@(VP,HP),CHR$(14);:STORAGE=VP:
POKE &H74,PEEK(&H74) OR 32:INPUT AN$:
PRINT CHR$(15);:
IF AN$="" THEN PRINT CHR$(29);CHR$(27);:GOTO 220
230 PRINT@(VP+1,1),CHR$(31);:
IF L=22 AND (AN$="GO OUT" OR AN$="GO DOOR")
THEN CC=2:GOTO 80
240 Z1=LEN(AN$):Z=INSTR(AN$," "):
IF Z<>0 THEN 380 ELSE IF AN$="WAIT" THEN 1300
250 IF AN$="SCORE" THEN 600
260 IF LEFT$(AN$,3)="INV" THEN 340
270 IF AN$="QUIT" THEN 440
280 IF AN$="HELP" THEN
```

```
PRINT"You might try examining EVERYTHING...":
FOR X=1 TO 2000:NEXT:GOTO 80
290 IF AN$="SAVE" THEN 450
ELSE IF AN$="LOAD" THEN 480
ELSE IF AN$="LISTEN" THEN 930
300 FOR X=0 TO 3:
IF AN$<>LEFT$(D$(X),1) THEN 320
ELSE IF L(L,X)=0 THEN 330
ELSE L=L(L,X):X=3:IF G=0 THEN SF=1
310 IF (H=4 OR H=5) THEN 80
ELSE IF RND(11)=11 THEN SF=1:GOTO 80 ELSE 80
320 NEXT:
PRINT"I'm afraid I don't understand what you want...":
FOR X=1 TO 2000:NEXT:GOTO 80
330 PRINT"You can't go that way":
FOR X=1 TO 2000:NEXT:GOTO 80
340 FL=0:PRINT"YOU ARE HOLDING:":
FOR X=1 TO NO:IF O(X)<>-1 THEN 360
ELSE FL=1:
IF 80-(POS(0)+2)< LEN(O$(X)) THEN
PRINT STRING$(80-POS(0),32)
350 PRINT O$(X);". ";
360 NEXT:
IF FL=0 THEN PRINT"NOTHING AT ALL" ELSE PRINT
370 GOTO 1870
380 VB$="":NO$="":VB$=LEFT$(AN$,3):
NO$=MID$(AN$,Z+1,3):
FOR X=1 TO NV:
IF VB$(X)=VB$ THEN VB=X:X=NV ELSE NEXT:
PRINT"I don't know how to ";LEFT$(AN$,Z-1);
" something":FOR X=1 TO 2000:NEXT:GOTO 80
390 IF VB<>1 THEN FOR X=1 TO NN:
IF NO$(X)=NO$ THEN NB=X:X=NN
ELSE NEXT:PRINT"I don't know what a ";
RIGHT$(AN$,LEN(AN$)-Z);" is.":
FOR X=1 TO 2000:NEXT:GOTO 80
400 IF VB>2 THEN 430 ELSE IF VB=1 THEN 1280
410 IF NB=5 OR NB=6 OR NB=7 OR NB=9 OR
NB=12 OR NB=14 OR NB=30 OR NB=31 OR
NB=40 THEN 1320
420 IF O(NB)=-1 THEN 1320
ELSE IF O(NB)=L THEN 1320 ELSE 1430
430 ON VB GOTO 30,30,1250,1250,1270,1270,1210,
1170,1170,950,1300,890,770,710,630,590,560
440 INPUT"Want to play again ";AN$:
IF LEFT$(AN$,1)="Y" THEN 30 ELSE CLS:END
450 OPEN"O",1,N$
460 FOR X=1 TO NO:PRINT #Q,O(X):NEXT:
PRINT #Q,L,CC,AU,ID,CP,H$:PRINT #Q,M$:CLOSE
470 PRINT"GAME SAVED":
FOR X=1 TO 2000:NEXT:GOTO 80
480 OPEN"I",1,N$
490 FOR X=1 TO NO:INPUT #Q,O(X):NEXT:
INPUT #Q,L,CC,AU,ID,CP,H$,M$:CLOSE
500 GOTO 80
510 CLS:
IF CC=2 THEN PRINT"The police have just arrived..."
ELSE PRINT"The police just broke the front door in..."
520 SB=1:GOSUB 600:
IF SC<90 THEN PRINT"You keep telling them you
didn't do it, but they aren't listening.":
PRINT"After all, there's the body, and there you are
without enough evidence to the":
PRINT"contrary...":SB=0:GOTO 440
530 PRINT"They see all your evidence, and ask you to
lead them to the murderer...":CP=1:CC=0:GOTO 1870
540 IF O(40)=0 THEN O(39)=0:GF=1
550 RETURN
560 IF L<>19 THEN 570 ELSE PRINT L$:
FOR X=1 TO 2000:NEXT:GOTO 80
570 IF L<>25 THEN 580
ELSE IF O(43)<>L THEN PRINT L$:
FOR X=1 TO 2000:NEXT:GOTO 80
ELSE IF O(55)<>-1 THEN 1310
ELSE O(43)=0:O(42)=L:L(25,0)=0:GOTO 80
580 IF L<>22 THEN 1430
ELSE IF O(41)<>L THEN PRINT L$:
FOR X=1 TO 2000:NEXT:GOTO 80
ELSE IF O(55)<>-1 THEN 1310
ELSE O(41)=0:O(40)=L:GOTO 80
590 IF L<>26 THEN PRINT"There is nothing here to
climb...":FOR X=1 TO 2000:NEXT:GOTO 80
ELSE PRINT"You slip on the cold metal and fall to your
death!":GOTO 440
600 SC=0:FOR X=44 TO 48:GOSUB 620:NEXT:
X=50:GOSUB 620:FOR X=58 TO 60:GOSUB 620:NEXT:
SC=SC+BF
610 PRINT"You have found";SC;"% of the evidence.":
FOR X=1 TO 2000:NEXT:IF SB=1 THEN RETURN
ELSE 80
620 IF O(X)=-1 THEN SC=SC+10:RETURN
ELSE RETURN
630 IF L<>19 THEN 690
640 IF O(55)<>-1 THEN 1310
650 IF GF=1 THEN 680
660 IF CP=1 THEN PRINT"OK...":PRINT"Susan's inside
with the knife. She sees the cop and gives up.":
PRINT"You've solved it all!":GOTO 440
ELSE PRINT"As the door opens the murderer stabs
YOU! ";
670 PRINT"You are dead!":GOTO 440
680 PRINT"The stall is empty. She got away.":
PRINT"But with all your evidence, you've cleared
yourself. Better luck next time.":GOTO 440
690 IF L<>25 THEN 700
ELSE IF O(42)<>L THEN PRINT OP$
ELSE IF O(55)<>-1 THEN 1310
ELSE O(42)=0:O(43)=L:L(25,0)=26:GOTO 80
700 IF L<>22 THEN 1430
```

```
ELSE IF O(40)<>L THEN PRINT OP$
ELSE IF O(55)<>-1 THEN 1310
ELSE O(40)=0:O(41)=L:GOTO 80
710 IF NB<>22 THEN 720
ELSE IF O(NB)=L THEN CLS:
PRINT"Z A P !":PRINT:PRINT:
PRINT"THE HIGH VOLTAGE KILLED YOU!":GOTO 440
ELSE 1430
720 IF NB<>31 THEN 730
ELSE IF L=17 THEN PRINT OP$:
FOR X=1 TO 2000:NEXT:GOTO 80
ELSE IF L=19 THEN PRINT L$:
FOR X=1 TO 2000:NEXT:GOTO 80
730 IF NB<>40 THEN 1290
ELSE IF L=19 THEN PRINT L$
ELSE IF L<>22 THEN 750
ELSE IF O(41)=L THEN PRINT OP$
ELSE IF O(40)=L THEN PRINT L$
740 FOR X=1 TO 2000:NEXT:GOTO 80
750 IF L=25 THEN IF O(42)=L THEN PRINT L$
ELSE PRINT OP$
760 FOR X=1 TO 2000:NEXT:GOTO 80
770 IF NB=5 OR NB=6 OR NB=7 OR NB=9 OR
NB=14 OR NB=40 THEN PRINT"How?":
FOR X=1 TO 2000:NEXT:GOTO 80
780 IF O(NB)<>L AND O(NB)<>-1 THEN 1430
ELSE IF NB=3 OR NB=4 THEN 1170
ELSE IF NB<13 THEN 790
ELSE IF NB=13 THEN 810
ELSE IF NB<24 THEN 790
ELSE IF NB=24 THEN 830
ELSE IF NB<34 THEN 790
ELSE IF NB=34 THEN 840
ELSE IF NB<39 THEN 790
ELSE IF NB=39 THEN 850
790 IF NB<55 THEN PRINT"I don't think that will help...":
FOR X=1 TO 2000:NEXT:GOTO 80
800 IF NB=55 THEN 860
ELSE IF NB<57 THEN 790
ELSE IF NB=57 THEN 870
ELSE IF NB=58 THEN 880
ELSE 790
810 INPUT"Who do you wish to call ";AN$:
IF INSTR(AN$,"COP") OR INSTR(AN$,"POL") THEN 820
ELSE 790
820 PRINT"The police are on the way...":
CC=2:GOTO 1740
830 PRINT"You sure are out of practice!":GOTO 1740
840 PRINT"A-h-h-h-h-h":
FOR X=1 TO 2000:NEXT:GOTO 80
850 PRINT"That's stealing!":
FOR X=1 TO 2000:NEXT:GOTO 80
860 PRINT"I think you want to lock or unlock something":
GOTO 1740
870 PRINT"Try ";CHR$(34);"THREAD REEL";CHR$(34):
GOTO 1740
880 PRINT"OK... You look a little strange...":GOTO 1740
890 IF AX<>0 THEN PRINT"It's the Program Director
wanting to know why the stationis off the air. ";:
GOTO 920
900 IF ID=0 THEN PRINT"It's the Program Director
wanting to know why the Station ID didn't play this":
PRINT"hour. ";:GOTO 920
910 PRINT"Wrong number":
FOR X=1 TO 2000:NEXT:GOTO 80
920 PRINT"He hung up when he heard your voice...":
CC=1:GOTO 1730
930 IF AX=0 THEN PRINT"I hear music playing all over
the station":GOTO 1740
940 PRINT"The station is silent.":GOTO 1740
950 IF NB=2 THEN 1040
960 IF NB=14 THEN 1450
970 IF NB=44 THEN 1070
980 IF NB=46 THEN 1080
990 IF NB=47 THEN 1110
1000 IF NB=50 THEN 1130
1010 IF NB=51 THEN 1140
1020 IF NB=52 THEN 1150
1030 IF NB=59 THEN 1160
ELSE PRINT"I see nothing to read here.":GOTO 1740
1040 PRINT"It says:":
PRINT TAB(10)"All announcers! Remember! The Station
ID recorder is broken! We MUST":
PRINT TAB(10)"make the ID manually at the beginning
of each hour! NO EXCEPTIONS!"
1050 PRINT TAB(10)"Engineering has rigged the
machine so we can give the ID by pushing"
1060 PRINT TAB(10)"the RED button. But you'll also
then have to restart the music by":
PRINT TAB(10)"pressing the GREEN button! Don't Blow
this!":PRINT TAB(60)"Steve":GOTO 1870
1070 IF O(44)<>-1 AND O(44)<>L THEN 1430
ELSE 1560
1080 IF O(46)<>-1 AND O(46)<>L THEN 1430
ELSE IF O(47)<>-1 AND O(47)<>L THEN PRINT"It's
the top half of a letter... It says:"
ELSE PRINT"The two pieces read:"
1090 PRINT"Dear Susan,":PRINT TAB(10)"I don't know
an easy way to say this... We have meant so":
PRINT TAB(10)"much to each other. You have been a
really great kid, and":PRINT TAB(10)"I've really had a
blast being with you, but ";
1100 IF O(47)<>-1 AND O(47)<>L THEN PRINT:
PRINT TAB(60)"AND THE REST IS MISSING...":
GOTO 1870
1110 IF O(47)<>-1 AND O(47)<>L THEN 130
ELSE IF O(46)<>-1 AND O(46)<>L THEN PRINT"It's
the botton half of a letter. It says:"
1120 PRINT TAB(55);" the time has now come":
PRINT TAB(10)"when we must part. I hope you will
```

```
forgive me.":PRINT TAB(60)"Mike"
1130 IF O(50)<>-1 AND O(50)<>L THEN 1430
ELSE 1610
1140 IF O(51)<>-1 AND O(51)<>L THEN 1430
ELSE 1620
1150 IF O(52)<>-1 AND O(52)<>L THEN 1430
ELSE 1650
1160 IF O(59)<>-1 AND O(59)<>L THEN 1430
ELSE 1710
1170 IF L<>1 THEN 1750
1180 IF NB=3 THEN AX=AU:AU=0:ID=1:PRINT"The
speaker says: This is KAXL, San Francisco":GOTO 1740
1190 IF NB=4 THEN AU=AX:
PRINT"The music has started again":AX=0:GOTO 1740
1200 PRINT"Why would you want to do that???":
GOTO 1740
1210 IF L<>23 THEN 1750
1220 IF NB<>57 THEN PRINT"You can only thread
REELs on the Automation...":GOTO 1740
1230 IF O(57)<>-1 THEN PRINT"What are you going
to thread onto the machine? You don't have any tape...":
GOTO 1730
1240 O(57)=0:AU=181:GOTO 1300
1250 IF NB<44 THEN 1290 ELSE IF O(NB)=-1 THEN
PRINT"You already have the ";O$(NB)
ELSE IF O(NB)<>L THEN 1430
ELSE O(NB)=-1:GOTO 1300
1260 GOTO 1740
1270 IF O(NB)<>-1 THEN
PRINT"You don't have the ";O$(NB):GOTO 1740
ELSE O(NB)=L:GOTO 1740
1280 FOR X=0 TO 3:
IF NO$=LEFT$(D$(X),3) THEN AN$=LEFT$(NO$,1):
X=3:GOTO 300
ELSE NEXT:PRINT"HUH?":
FOR X=1 TO 2000:NEXT:GOTO 80
1290 PRINT"It is beyond my power to do that":
FOR X=1 TO 2000:NEXT:GOTO 80
1300 PRINT"OK":FOR X=1 TO 2000:NEXT:GOTO 80
1310 PRINT"I can't do that...yet...":
FOR X=1 TO 2000:NEXT:GOTO 80
1320 ON NB GOTO 1340,1350,1360,1360,1400,1410,
1420,1370,1380,1390,30,1440,1360,1450,1480,1490,30,
30,30,1500,30,1370,30,1510,30,30,1360,1360,30,1520,
1530,30,30,1360,30,30,1540,1550,1360,1360,30,30,30
1330 ON NB-43 GOTO 1560,1570,1580,1590,1600,1360,
1610,1620,1650,1660,1670,1680,1360,1690,1700,1710,
1720
1340 PRINT"It's Mike all right... and he's got a knife
wound in the back!":BF=10:
IF O(60)<>-1 THEN O(60)=L:GOTO 1730
ELSE 1730
1350 PRINT"There's something there all right...":
FOR X=1 TO 2000:NEXT:GOTO 80
1360 PRINT"I see nothing unusual":
FOR X=1 TO 2000:NEXT:GOTO 80
1370 PRINT"There's too many switches and controls
here for me to understand...":PRINT"leave it alone!":
GOTO 1730
1380 IF L=3 OR L=4 OR L=12 THEN 1360 ELSE 1430
1390 IF O(45)<>-1 THEN O(45)=L:GOTO 1350
ELSE 1360
1400 IF L=1 OR L=7 OR L=12 OR L=13 THEN 1360
ELSE 1430
1410 IF L=1 OR L=8 THEN 1360
ELSE IF L=7 THEN IF O(48)<>-1 THEN
PRINT"There's a record there...":O(48)=L:
GOSUB 1740:GOTO 80
ELSE 1360
ELSE 1430
1420 IF L=1 OR L=7 OR L=2 THEN 1370 ELSE 1850
1430 PRINT"I don't see it here...":
FOR X=1 TO 2000:NEXT:GOTO 80
1440 IF L=4 THEN 1360
ELSE IF L=14 THEN IF O(51)<>-1 THEN O(51)=L:
GOTO 1350
ELSE 1360
ELSE 1430
1450 IF L=4 THEN PRINT"It says:":
PRINT TAB(10)"Welcome to KAXL":
PRINT TAB(10)"Please see receptionist for tour":
GOTO 1730
1460 IF L=22 THEN PRINT"It says:":
PRINT TAB(10)"NO VISITORS AFTER 5 PM":GOTO 1730
ELSE IF L=23 THEN PRINT"Electric sign says:":
PRINT TAB(10);AU;"minutes until tape runout":
GOTO 1730
1470 GOTO 1430
1480 IF O(47)<>-1 THEN O(47)=L:GOTO 1350
ELSE 1360
1490 IF O(46)<>-1 THEN O(46)=L:GOTO 1350
ELSE 1360
1500 PRINT"There must be thousands of old hits here!":
FOR X=1 TO 3000:NEXT:
IF O(49)<>-1 THEN PRINT"H-m-m-m...";:
FOR X=1 TO 2000:NEXT:
PRINT" and one empty sleeve... Seems a record is
gone.":O(49)=L:GOTO 1730
ELSE 80
1510 PRINT"Looks kinda old... but I'm sure it still works.":
GOTO 1730
1520 IF L=17 THEN 1360 ELSE IF L=19 THEN
PRINT"There's a light red stain here...":
GOTO 1740 ELSE 1430
1530 IF L=17 THEN PRINT OP$:GOTO 80
ELSE IF L=19 THEN PRINT L$:
FOR X=1 TO 2000:NEXT:GOTO 80
ELSE 1430
1540 PRINT"It's a LONG WAY UP>>>>>>>>!":
GOTO 1740
```

```
1550 PRINT"It's too high to climb... and there's no
gate...":GOTO 1730
1560 PRINT"The notepad says:":
PRINT TAB(10)"@ 11 remember to open door for Susan":
PRINT TAB(10)"get record":
PRINT TAB(10)"call cleaners @ 8 am":
PRINT TAB(20)"Maybe it means something to you, but I
don't understand...":GOTO 1870
1570 PRINT"It's a ";CHR$(34);"Virginia Slims";CHR$(34);
" and there's lipstick on it.":GOTO 1730
1580 PRINT"It's the bottom half of a torn sheet.":
GOTO 1740
1590 PRINT"It's the top half of a torn sheet.":GOTO 1740
1600 PRINT"It's broken... it says:";CHR$(34);"MISTY";
CHR$(34);" on the label...":GOTO 1730
1610 PRINT"It's the transmitter log. Mike's last entry was
at 11:00":PRINT"That could help establish time of
death...":GOTO 1730
1620 PRINT"It's a memo from the manager"
1630 PRINT TAB(10)"All Staff!":
PRINT TAB(20)"This business with visitors after business
hours is":
PRINT TAB(20)"going to STOP! It's DANGEROUS to let
people in this":
PRINT TAB(20)"place! I'm going to can the next guy who
pulls this."
1640 PRINT TAB(60)"Steve":GOTO 1870
1650 PRINT"It's a ratings book that shows the station to
be number 1":GOTO 1730
1660 PRINT"They're too big to go in your radio...":
GOTO 1740
1670 PRINT"I don't see a thing that's useful...":
GOTO 1740
1680 PRINT"It's a man's keyring with several keys...":
GOTO 1740
1690 PRINT"There's a three-hour reel of recorded music
here.":GOTO 1740
1700 PRINT"It's just a cheap lipstick... it's been used
some...":GOTO 1740
1710 PRINT"It's pretty mushy...":PRINT"Let's just say
that some girl named Susan was really hot":
PRINT"for our boy Mike...":GOTO 1730
1720 PRINT"It looks like a bead off a girl's necklace or
earring..."
1730 FOR X=1 TO 5000:NEXT:GOTO 80
1740 FOR X=1 TO 3000:NEXT:GOTO 80
1750 PRINT"You can't do that from HERE...":GOTO 1740
1760 DATA "NORTH ","EAST ","SOUTH ","WEST "
1770 DATA at the Control Board,0,0,0,2,
in the Control Room,11,1,10,3,
in the staff lounge,28,2,0,27,
in the Lobby,10,14,22,15,
in the News Room,27,10,0,6,
at a TELETYPE MACHINE in the News Room,0,5,0,0,
in the Production Room,0,0,0,11
1780 DATA in the Record Library,0,0,15,0,
in the Transmitter Room,25,16,11,18,
in a dark passage,2,13,4,5,
in a dark passage,9,7,2,23,
in a Large Studio,13,0,0,0,
in a Small Studio,0,0,12,10,
in the Manager's Office,0,0,0,4
1790 DATA in the Program Director's office,8,4,0,0,
in the Supply Room,0,0,0,9,
in the Men's Room,0,27,0,21,
in the Workshop,0,9,0,0,
in the Ladies' Room,0,0,27,0,
in the stall,0,0,0,0,
in the stall,0,17,0,0,
at the Front Door,4,0,0,0
1800 DATA at the Automation System,0,11,0,24,
in the Reel Storage area,0,23,0,0,
at the Back Door,0,0,9,0,
at the Tower,0,0,25,0,
in a dark passage,19,3,5,17,
at the staff Noteboxes in the Lounge,0,0,3,0
1810 DATA a body (DEAD I think!),1,bulletin board,1,
Red switch,1,Green switch,1,Microphone,1,Turntables,1,
Control Board,2,Equipment,2,Couch,3,Ash tray,3,
Couch,4,Desk,4,Phone,4,sign,4,typewriter,5,
wastebasket,5,Control Board,7,Turntables,7
1820 DATA Microphone,7,Shelves loaded with records,8,
Turn table for audition of records,8,Transmitter,9,
Microphones,12,Piano,12,a Large (rumpled) Couch,12,
Microphones,13,Table,13,Chairs,13,Desk,14,Lavatory,17,
Stall,17,Lavatory,19,Stall,19
1830 DATA john,21,sign,22,sign,23,tower,26,
a heavy wire fence,26,a strange car,22,locked door,0,
Unlocked Door,22,Locked door,25,unlocked door,0,
Notepad,1,Cigarette butt,0,torn half of a letter,0,
paper,0,record,0,empty record sleve,0
1840 DATA Transmitter Log sheet,9,Memo,0,
Ratings book,15,Tubes,16,misc. parts,16,keys,4,
test equipment,18,reel of recorded tape,24,lipstick,27,
lovenote,28,bead,0
1850 DATA "GO ",EXA,GET,TAK,DRO,PUT,THR,PUS,
PRE,REA,SIT,ANS,USE,OPE,UNL,CLI,LOC
1860 DATA BOD,BUL,RED,GRE,MIC,TUR,CON,EQU,
COU,ASH,,DES,PHO,SIG,TYP,WAS,,,,SHE,,TRA,,PIA,,,
TAB,CHA,,LAV,STA,,,JOH,,,TOW,FEN,CAR,DOO,,,,NOT,
BUT,LET,PAP,REC,SLE,LOG,MEM,RAT,TUB,PAR,KEY,
TES,REE,LIP,LOV,BEA
1870 PRINT"Press <enter>";
1880 IF INKEY$=""" THEN 1880 ELSE 90
1890 IF VP=>STORAGE THEN RETURN
1900 TEMP=VP+1
1910 PRINT@(TEMP,0),STRING$(80,32)
1920 IF TEMP=STORAGE THEN RETURN
ELSE TEMP=TEMP+1:GOTO 1910
```

PROGRAMMING TIDBITS

MOD-III POP-UP WINDOWS

Yes, sir, ma'am, Model III. Readers' demand for more coverage of that fine machine, duly noted in a recent editorial, prompted today's idea. It's a little challenge to the alleged superiority of MSDOS machines which for several years now have been bombarding the world with cute pop-up menus and help windows. But the programming of those gimmicks is a pain and eats up oodles of memory and disk space. Yet it's ridiculously simple in Mod-III. The whole process of displaying and hiding a little window anywhere on the screen boils down to swapping a part of video memory with the contents of a window. With the help of the following miniature machine subroutine (42 bytes) it happens instantly, even in BASIC (if you are not comfortable with assembly language then send me 5 bucks for a disk with this routine, c/o Microdex, 1212 N. Sawtelle, Tucson AZ 85716; specify TRSDOS/LDOS).

On entry to the routine Z-80 registers must be set up like this:
HL points to window data, stored as a continuous "string" of window rows.
B = number of window rows.
C = number of columns in each row.
D = row (0-15) of screen, where the upper-left corner of the window will be "anchored".
E = column (0-63) of anchor point.

First some preliminaries: translate the row and column of the anchor point into an absolute video memory address, and rearrange a couple of other registers.

```
;-------------------------------------------------------------------------
POPUP   xor     A       ;A=0
        srl     d
        rra
        srl     d
        rra             ;now DA=D*64
        add     a,e     ;add column
        ld      e,a     ;DE=anchor offset
        ld      a,60    ;video page #
        add     a,d     ;add to offset
        ld      d,a     ;DE=anchor address
        ld      a,b     ;A=window rows
        ld      b,0     ;BC=window columns
;-------------------------------------------------------------------------
```

There is a bit of creative math here. We need to multiply the anchor row times 64, add the column to it, and then add this offset to the address of screen memory. Sounds like a lot to do, but watch this. We put the row into the "high" register D to represent multiples of 256. Two bitwise shifts to the right divide it by 4. It's the same as times 64, only much easier. The column is always less than 64, so we can simply add it to the "low" byte of the result. Video memory starts at 15360 (3C00'hex) which is a multiple of 256 ("page boundary"). So we only need to add its page number 60 (3C'hex) to the "high" byte of the total. Neat, huh? Now continue with the main part of the routine.

```
;-------------------------------------------------------------------------
POPY    push    af      ;stash rows
        push    bc      ;and columns
POPX    ld      a,(de)  ;A=video char
        ldi             ;data to video
        dec     hl
        ld      (hl),a  ;store video char
        inc     hl
        inc     c       ;test C
        dec     c       ;more columns?
        jr      nz,POPX ;yes
        ex      de,hl   ;else next row
        ld      bc,64   ;video width
        add     hl,bc   ;add it
        pop     bc      ;columns
        sbc     hl,bc   ;subtract it
        ex      de,hl   ;DE=>next row
        pop     af
        dec     a       ;more rows?
        jr      nz,POPY ;yes
        RET             ;else done
;-------------------------------------------------------------------------
```

The POPX loop picks a character from video memory and holds it in register A, while LDI copies a character of window text to video. Then we store the previous video character in the window string (since LDI already incremented HL to the next position in that string, we must DEC it before storing A, and then INC it again). The LDI automatically decrements the count of window columns BC. Since it's never greater than 63, only the register C matters and so we run the loop until C=0. After one entire row is swapped, we add 64 (full screen width) to the video pointer, and subtract the width of the window from it, so that the next row of the window will align with the previous. This goes on until the count of rows in register A gets down to zero. After all rows are swapped, the window data string holds what was on the screen before. Next time the same routine can be used to hide the window by swapping that previous screen contents with the window.

To use this scheme from BASIC, some additional code is needed up front. We will send to the routine all data for the window in one string. The first 4 bytes of that string will be the row and column of the anchor point, and the

number of window rows and columns. The rest will be the displayable window text. The USR call puts the "VARPTR" of the string into register DE (the VARPTR points to a string identifier; the first byte is the string length, the next 2 bytes are the address of the actual string text). The following front-end code extracts the first 4 bytes from the string and sets up registers needed by the POPUP routine:

```
;-------------------------------------------------------------------------
          ORG    64000     ;DE=VARPTR
          ex     de,hl     ;HL=VARPTR
          inc    hl
          ld     e,(hl)
          inc    hl
          ld     d,(hl)
          ex     de,hl     ;HL=>string
          ld     d,(hl)    ;D=anchor row
          inc    hl
          ld     e,(hl)    ;E=anchor column
          inc    hl
          ld     b,(hl)    ;B=window rows
          inc    hl
          ld     c,(hl)    ;C=window cols
          inc    hl        ;HL=>window text
POPUP     ..... continue here
          END
;-------------------------------------------------------------------------
```

Don't forget the END at the end of the listing and assemble it as POPUP/BIN. To test it, enter BASIC, protecting memory from 64000 up:

```
TRSDOS 1.3      BASIC -M:63999
LDOS 5          BASIC (M=63999)
```

Then try this demo program:

```
10 clear 1000: defint A-Z
12 cmd "L", "POPUP/BIN"
14 def usr = 64000 - 65536
16 for x=1 to 146: print "TRS-80 ";: next

21 w1$=chr$(151)+string$(10,131)+chr$(171)
22 w2$=chr$(149)+" TRSTimes "+chr$(170)
23 w3$=chr$(181)+string$(10,176)+chr$(186)
24 w4$=chr$(149)+string$(10,32)+chr$(170)
25 w5$=chr$(149)+" Magazine "+chr$(170)
40 ww$=w1$+w2$+w3$+w4$+w5$+w3$
42 wy=6: wx=12: w$=chr$(wy)+chr$(wx)+ww$
44 vy=5: vx=26

50 z$=chr$(vy)+chr$(vx) + w$
60 if inkey$="" then goto 60
65 x$ = usr (z$)
70 if inkey$="" then goto 70
75 x$ = usr (z$)
80 vy=vy+1: if vy > (16-wy) then vy=0
85 vx=vx+2: if vx > (64-wx) then vx=0
90 goto 50
```

Lines 10 through 16 load our subroutine, define USR address, and print a background on the screen.

Lines 21 through 25 build substrings for the rows of the window. In line 40 the rows are collected into one window text string WW$. In line 42 the number of window rows WY and columns WX are converted into a 2-byte string and combined with the window text into a window data string W$. In line 44 variables VY and VX are the row and column of the "anchor" point. Their values can be changed to pop the window anywhere on the screen.

Line 50 builds the string Z$ to be sent to our routine. Anchor coordinates are converted into a 2-byte string and window data string W$ is added to it.

Now press any key to activate the USR call and pop up the window. Press a key again to restore the screen. Hold down any key for a while to appreciate the speed with which the window pops all over the screen. Delete line 60 for a sort of animation ("dynamic dragging" they call it). In line 42 change WY=6 to WY=3 to pop only a part of the window. And so on. Press BREAK to quit the demo.

I can already hear the gears in your mind, turning over the possibilities of our little gizmo. Of course, BASIC strings are limited to 255 characters, so the windows can't be larger than some 15 rows by 16 columns, or 10x25, etc. But that's more than plenty for pop-up menus, help messages, graphic borders, happy faces and such (in machine programs the windows can be any size, up to the full screen). The number of characters in each row must be exactly the same as specified in the variable WX. The variable WY must not be greater than the number of rows stored in the window string. But it may be smaller, in which case only the inital WY rows will pop up on the screen (perhaps we want to hide items not applicable at some stages in the program). In any case beware: to prevent program crash, the window size must be co-ordinated with the "anchor" point so as to never allow window overlap beyond video limits.

The machine subroutine could be also assembled as a self-protecting "memory module" (yes, even under TRSDOS 1.3). By intentional design it has no "fixed" addresses, so that only the simplest "loader" is needed. If you are interested, look up the explanation of memory modules for assembly and BASIC programming, in Volume II of "Z-80 Tutor" (available from Microdex, see ad elsewhere in this issue).

With some modifications the idea works just as well in Mod-4. But in video pranks Model III shines over many other machines, because of the simplicity of its "memory mapped" screen.

*** DR. PATCH ***

BRAND-NEW UTILITY FOR TRS-80 MODEL 4 AND LS-DOS 6.3.1

A 'MUST HAVE' FOR ALL LS-DOS 6.3.1 OWNERS.

DR. PATCH MODIFIES LS-DOS 6.3.1 TO DO THINGS THAT WERE NEVER BEFORE POSSIBLE.

COMPLETELY SELF-CONTAINED - MENU-DRIVEN FOR MAXIMUM USER CONVENIENCE.

FAST & SAFE - EACH MODIFICATION IS EASILY REVERSED TO NORMAL DOS OPERATION.

DISABLE PASSWORD CHECK IN FORMAT/CMD
FORMAT DOUBLE-SIDED AS DEFAULT
FORMAT 80 TRACKS AS DEFAULT
DISABLE VERIFY AFTER FORMAT
CHANGE 'DIR' TO 'D'
CHANGE 'CAT' TO 'C'
VIEW DIR/CAT WITH (I) PARAMETER AS DEFAULT
VIEW DIR/CAT WITH (S,I) PARAMETERS AS DEFAULT
CHANGE 'REMOVE' TO 'DEL'
CHANGE 'RENAME' TO 'REN'
CHANGE 'MEMORY' TO 'MEM'
CHANGE 'DEVICE' TO 'DEV'
DISABLE THE BOOT 'DATE' PROMPT
DISABLE THE BOOT 'TIME' PROMPT
DISABLE FILE PASSWORD PROTECTION
ENABLE EXTENDED ERROR MESSAGES

DISABLE PASSWORD CHECK IN BACKUP/CMD
BACKUP WITH (I) PARAMETER AS DEFAULT
BACKUP WITH VERIFY DISABLED
DISABLE BACKUP 'LIMIT' PROTECTION
DISABLE PASSWORD CHECK IN PURGE
PURGE WITH (I) PARAMETER AS DEFAULT
PURGE WITH (S,I) PARAMETERS AS DEFAULT
PURGE WITH (Q=N) PARAMETER AS DEFAULT
IMPLEMENT THE DOS 'KILL' COMMAND
CHANGE DOS PROMPT TO CUSTOM PROMPT
TURN 'AUTO BREAK DISABLE' OFF
TURN 'SYSGEN' MESSAGE OFF
BOOT WITH NON-BLINKING CURSOR
BOOT WITH CUSTOM CURSOR
BOOT WITH CLOCK ON
BOOT WITH FAST KEY-REPEAT

DR. PATCH IS THE ONLY PROGRAM OF ITS TYPE EVER WRITTEN FOR THE TRS-80 MODEL 4 AND LS-DOS 6.3.1. IT IS DISTRIBUTED EXCLUSIVELY BY TRSTIMES MAGAZINE ON A STANDARD LS-DOS 6.3.1 DATA DISKETTE, ALONG WITH WRITTEN DOCUMENTATION.

*** DR. PATCH *** $14.95

SHIPPING & HANDLING: U.S & CANADA - NONE
ELSEWHERE - ADD $4.00
(U.S CURRENCY ONLY, PLEASE)

TRSTimes magazine - dept. DP
5721 Topanga Canyon Blvd. #4
Woodland Hills, CA 91367

DON'T LET YOUR LS-DOS 6.3.1 BE WITHOUT IT!

HINTS & TIPS

DMP-200 SETUP
by Fred Bennett

The Radio Shack DMP-200 printer has been a workhorse for me. Even after all these years, it has remained a good and reliable machine, capable of many different ways of printing my data.

To ease the setup of the machine, I use the program listed below. It is written for the DMP-200 but it should work for most all of the Tandy DMP line.

```
10 CLS:PRINT"DMP-200 PRINTER DRIVER"
20 TE$="AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRr
SsTtUuVvWwXxYyZz1234567890!#$%&'():*-=@;+/?"
30 PRINT:PRINT"Do you want Data Processing or Word
Processing mode (D/W) ";
40 A$=INKEY$
50 IF A$="D" OR A$="d" THEN A$="Data Processing
mode":GOTO 290
60 IF A$="W" OR A$="w" THEN A$="Word Processing
mode":GOTO 300
70 GOTO 40
80 PRINT:PRINT"Standard, Correspondence or
Proportional Font Style (S, C, P) ";
90 B$=INKEY$
100 IF B$="S" OR B$="s" THEN B$="Standard
font":GOTO 310
110 IF B$="C" OR B$="c" THEN B$="Correspondence
font":GOTO 320
120 IF B$="P" OR B$="p" THEN B$="Proportional
font":GOTO 330
130 GOTO 90
140 PRINT:PRINT"Select pitch width:":
PRINT"1. Normal (10 cpi)":
PRINT"2. Compressed (12 cpi)":
PRINT"3. Condensed (16.7 cpi)"
150 C$=INKEY$
160 IF C$="1" THEN C$="10 cpi":GOTO 340
170 IF C$="2" THEN C$="12 cpi":GOTO 350
180 IF C$="3" THEN C$="16.7 cpi":GOTO 360
190 GOTO 150
200 PRINT:PRINT"Press any key to continue ";
210 X$=INKEY$:IF X$="" THEN 210
220 CLS:PRINT"Ready printer"
230 PRINT:PRINT"Request test function (Y/N) ";
240 D$=INKEY$
250 IF D$="Y" OR D$="y" THEN 370
260 IF D$="N" OR D$="n" THEN 280
270 GOTO 240
280 END
290 LPRINT CHR$(19);:
PRINT" Data Processing mode selected":GOTO 80
300 LPRINT CHR$(20);:
PRINT"Word Processing mode selected":GOTO 80
310 LPRINT CHR$(27);CHR$(19);:
PRINT"Normal font selected":GOTO 140
320 LPRINT CHR$(27);CHR$(18);:
PRINT"Correspondence font selected":GOTO 200
330 LPRINT CHR$(27);CHR$(17);:
PRINT"Proportional font selected":GOTO 200
340 LPRINT CHR$(27);CHR$(19);:
PRINT"Normal 10 pitch":GOTO 200
350 LPRINT CHR$(27);CHR$(23);:
PRINT"Compressed 12 pitch":GOTO 200
360 LPRINT CHR$(27);CHR$(20);:
PRINT"Condensed 16.7 pitch":GOTO 200
370 LPRINT TE$:
LPRINT"Selected options are: ";A$;", ";B$;", ";C$
380 PRINT:PRINT"Change options (Y/N) ";
390 E$=INKEY$
400 IF E$="Y" OR E$="y" THEN 10
410 IF E$="N" OR E$="n" THEN 280
420 GOTO 390
```

A LITTLE COMPUTER MATH
Model I/III & 4
by Frank Tipps

Here is a series of drills to help you learn the four logical and mathematical functions that your computer performs: binary OR, binary AND, binary addition, and hexadecimal addition.

Binary OR and AND are logical operations that compare two values bit for bit. The outcome of OR is true (i.e. equal to one) if at least one of the inputs is true. This means that OR is false (i.e. equal to zero) only when both inputs are false. In contrast, the outcome of AND is true only if both inputs are true.

Binary addition is easy to learn, but because you work with only zero and one, you have to make carries much more frequently in binary addition than in decimal addition. Two simple rules make carrying out to the next [lace easy:

1 + 1 = 0 with a carry out of 1
1 + 1 + 1 = 1 with a carry out of 1

As in the decimal system, you work binary addition from right to left.

Hexadecimal notation uses the decimal digits zero through nine and letters A through F. Adding hexadecimal numbers is just a little more difficult than adding binary numbers.

Note that the listings to BINOR/BAS, BINAND/BAS and BINADD/BAS are nearly identical. Simply type in BINOR/BAS and use it to create BINAND/BAS. Change lines 40, 220, and 400. To create BINADD/BAS, change lines 40, 50, 220, and 400.

Finally, as listed, the programs will run correctly on the Model 4. To run on Models I and III, change line 20 in all programs to read: 20 SW=64

BINOR/BAS

```
5 'BINOR/BAS
10 DIM A1(7),A2(7),AN(7)
20 SW=80
30 CLS
40 PRINT"BINARY OR"
50 H1=RND(256)-1:H2=RND(256)-1
60 HX=H1
70 FOR B=7 TO 0 STEP-1
80 GOSUB 470
90 A1(B)=INT(HX/B1)
100 HX=HX-(A1(B)*B1)
110 NEXT
120 HX=H2
130 FOR B=7 TO 0 STEP-1
140 GOSUB 470
150 A2(B)=INT(HX/B1)
160 HX=HX-INT(A2(B)*B1)
170 NEXT
180 PRINT@SW*2+5,A1(7);
190 FOR I=6 TO 0 STEP-1
200 PRINT A1(I);
210 NEXT
220 PRINT@SW*3,"OR";
230 PRINT@SW*4,STRING$(32,"-");
240 PRINT@SW*3+5,A2(7);
250 FOR I=6 TO 0 STEP-1
260 PRINT A2(I);
270 NEXT
280 FOR I=0 TO 7
290 PRINT@SW*5+27-3*I,"";
300 A$=INKEY$:IF A$="" THEN 300
310 IF (A$<>"0") AND (A$<>"1") THEN 300
320 AN(I)=VAL(A$)
330 PRINT A$;
340 NEXT
350 A=0
360 FOR B=0 TO 7
370 GOSUB 470
380 A=A+(AN(B)*B1)
390 NEXT
400 IF A<>(H1 OR H2) THEN 430
410 PRINT@SW*7,"CORRECT"
420 FOR I=1 TO 3000:NEXT:GOTO 30
430 PRINT@SW*7,"WRONG, TRY AGAIN!"
440 FOR I=1 TO 3000:NEXT
450 PRINT@SW*5,CHR$(31)
460 GOTO 280
470 IF B=0 THEN B1=1:RETURN
480 B1=1
490 FOR J=1 TO B:B1=2*B1:NEXT
500 RETURN
```

BINAND/BAS

```
5 'BINAND/BAS
10 DIM A1(7),A2(7),AN(7)
20 SW=80
30 CLS
40 PRINT"BINARY AND"
50 H1=RND(256)-1:H2=RND(256)-1
60 HX=H1
70 FOR B=7 TO 0 STEP-1
80 GOSUB 470
90 A1(B)=INT(HX/B1)
100 HX=HX-(A1(B)*B1)
110 NEXT
120 HX=H2
130 FOR B=7 TO 0 STEP-1
140 GOSUB 470
150 A2(B)=INT(HX/B1)
160 HX=HX-INT(A2(B)*B1)
170 NEXT
180 PRINT@SW*2+5,A1(7);
190 FOR I=6 TO 0 STEP-1
200 PRINT A1(I);
210 NEXT
220 PRINT@SW*3,"AND";
230 PRINT@SW*4,STRING$(32,"-");
240 PRINT@SW*3+5,A2(7);
250 FOR I=6 TO 0 STEP-1
260 PRINT A2(I);
270 NEXT
280 FOR I=0 TO 7
290 PRINT@SW*5+27-3*I,"";
300 A$=INKEY$:IF A$="" THEN 300
310 IF (A$<>"0") AND (A$<>"1") THEN 300
320 AN(I)=VAL(A$)
330 PRINT A$;
340 NEXT
350 A=0
360 FOR B=0 TO 7
370 GOSUB 470
380 A=A+(AN(B)*B1)
390 NEXT
400 IF A<>(H1 AND H2) THEN 430
410 PRINT@SW*7,"CORRECT"
420 FOR I=1 TO 3000:NEXT:GOTO 30
430 PRINT@SW*7,"WRONG, TRY AGAIN!"
```

```
440 FOR I=1 TO 3000:NEXT
450 PRINT@SW*5,CHR$(31)
460 GOTO 280
470 IF B=0 THEN B1=1:RETURN
480 B1=1
490 FOR J=1 TO B:B1=2*B1:NEXT
500 RETURN
```

BINADD/BAS

```
5 'BINADD/BAS
10 DIM A1(7),A2(7),AN(7)
20 SW=80
30 CLS
40 PRINT"BINARY ADDITION"
50 H1=RND(128)-1:H2=RND(128)-1
60 HX=H1
70 FOR B=7 TO 0 STEP-1
80 GOSUB 470
90 A1(B)=INT(HX/B1)
100 HX=HX-(A1(B)*B1)
110 NEXT
120 HX=H2
130 FOR B=7 TO 0 STEP-1
140 GOSUB 470
150 A2(B)=INT(HX/B1)
160 HX=HX-INT(A2(B)*B1)
170 NEXT
180 PRINT@SW*2+5,A1(7);
190 FOR I=6 TO 0 STEP-1
200 PRINT A1(I);
210 NEXT
220 PRINT@SW*3,"+";
230 PRINT@SW*4,STRING$(32,"-");
240 PRINT@SW*3+5,A2(7);
250 FOR I=6 TO 0 STEP-1
260 PRINT A2(I);
270 NEXT
280 FOR I=0 TO 7
290 PRINT@SW*5+27-3*I,"";
300 A$=INKEY$:IF A$="" THEN 300
310 IF (A$<>"0") AND (A$<>"1") THEN 300
320 AN(I)=VAL(A$)
330 PRINT A$;
340 NEXT
350 A=0
360 FOR B=0 TO 7
370 GOSUB 470
380 A=A+(AN(B)*B1)
390 NEXT
400 IF A<>(H1+H2) THEN 430
410 PRINT@SW*7,"CORRECT"
420 FOR I=1 TO 3000:NEXT:GOTO 30
430 PRINT@SW*7,"WRONG, TRY AGAIN!"
```


```
440 FOR I=1 TO 3000:NEXT
450 PRINT@SW*5,CHR$(31)
460 GOTO 280
470 IF B=0 THEN B1=1:RETURN
480 B1=1
490 FOR J=1 TO B:B1=2*B1:NEXT
500 RETURN
```


HEXADD/BAS

```
5 'HEXADD/BAS
10 CLS
20 SW=80
30 PRINT"HEX-ERCISE"
40 PRINT"BYTE ADDITION"
50 B1=RND(128)-1:B2=RND(128)-1
60 M1=INT(B1/16):L1=B1-(M1*16)
70 M2=INT(B2/16):L2=B2-(M2*16)
80 IF L1<10 THEN PRINT@SW*4+14,STR$(L1);:
GOTO 100
90 PRINT@SW*4+15,CHR$(L1+55)
100 IF M1<10 THEN PRINT@SW*4+13,STR$(M1);:
GOTO 120
110 PRINT@SW*4+13),CHR$(M1+55);
120 PRINT@SW*5+10,"+ ";
130 IF L2<10 THEN PRINT@SW*5+14,STR$(L2);:
GOTO 150
140 PRINT@SW*5+15,CHR$(L2+55);
150 IF M2<10 THEN PRINT@SW*5+13,STR$(M2);:
GOTO 180
160 PRINT@SW*5+13,CHR$(M2+55);
170 PRINT@SW*5+10,"+ ";
180 PRINT@SW*6+12,"----";
190 PRINT@SW*7+15,CHR$(14);
200 A$=INKEY$:IF A$="" THEN 200
210 PRINT@SW*7+15,A$;
220 PRINT@SW*7+14,"";
230 B$=INKEY$:IF B$="" THEN 230
240 PRINT@SW*7+14,B$;
250 PRINT CHR$(15);
260 IF (A$<"0") OR (A$>"F") THEN 370
270 IF (B$<"0") OR (B$>"F") THEN 370
280 IF A$<="9" THEN A=VAL(A$)
290 IF A$=>"A" THEN A=ASC(A$)-55
300 IF B$<="9" THEN B=VAL(B$):GOTO 330
310 IF B$=>"A" THEN B=ASC(B$)-55:GOTO 330
320 GOTO 370
330 AN=A+(16*B)
340 IF AN<>(B1+B2) THEN 370
350 PRINT@SW*9+10,"CORRECT";
360 FOR I=1 TO 3000:NEXT:GOTO 10
370 PRINT@SW*9+10,"WRONG! TRY AGAIN";
380 FOR X=1 TO 3000:NEXT:PRINT@SW*7,CHR$(31)
390 GOTO 190
```

ASCII REVISITED

by Roy T. Beck

I know the ASCII code has been around a long time, but there are always newcomers to whom even an old tale can be new. If you are an oldtimer who knows all about ASCII, then bypass this story and go on to the next one. If you're not sure about ASCII, stick around, and I'll try to add a few insights.

First off, what is the meaning of ASCII? It stands for American Standard Code for Information Interchange and is pronounced "ASS-KEY". The standard was created a good many years ago, and fortunately it has been implemented widely, (although not universally) among computer and peripheral manufacturers. A notable holdout is IBM in certain of their mainframe equipment. They use EBCDIC in some of their equipment. Still another code was used in punched cards, which are now obsolete. Luckily for us, all printer manufacturers, all MODEM manufacturers, and all personal computer manufacturers have agreed to the use of ASCII as their basic interchange language. I say basic, because there are extensions of ASCII which are not widely implemented, and this brings about all sorts of difficulties.

Note that ASCII, as defined in the standard, ia a 128 byte code, with only 7 bits per character. Fundamentally, there are 96 "printable" characters and 32 "control codes" in ASCII. Incidentally, 20h, the space character is classed as a printable character, because it takes up space on the printed page. The code values are distributed as follows:

A-Z	26 chars
a-z	26 chars
0-9	10 chars
space code	1 char
delete code (7Fh)	1 char
punctuation & symbols	32 chars
Printable values	96 chars
Control codes	32 chars
Complete ASCII range	128 chars
Vendor's option	128 chars
Maximum number of values in 8 bit byte	256 chars

The code was established in the days of the teletype, and so included many control characters required by the "modern" teletype. I say modern because it used a 7 level character code, whereas the oldest teletype used a 5 level code. That older code was known as the "Baudot" code. 5 levels and 7 levels as used with respect to teletypes is the same as 5 bit and 7 bit bytes in computer parlance. As computers and MODEMs came along, and teletype code was sent over telephone lines and radio waves, the need for error checking and/or correcting came along with it, as the noise inherent in such circuits caused errors in the received code. The computer practice of using 8 bit bytes allowed the 7 bit ASCII code to be supplemented with an eighth bit for parity checking. This bit could be used to make the sum of all the bits in a byte total to an odd or an even number. If the total was made to come out even, you had even parity. If the total was made odd, you had an odd parity system. Modem practice today allows for odd, even, or no parity, using 7 or 8 bit bytes. If 8 bits is used and no parity checking is implemented, you have "no" or "none" parity. Of course, if only 7 significant bits are used, the parity may be set as odd, even or "none", (N). Most BBS' today use 8-N-1 as their protocol, and this means 8 bit bytes, no parity, and 1 stop bit. The "stop bit" is another relic of the teletype era. Its purpose was to allow the mechanical teletype to figure out when a byte ended. There was also a "start bit" used to resynchronize the system at the beginning of each byte. The teletype system only synchronized the beginning of each byte. It depended upon accurate adjustment of the motor speed at each station to keep all the bits in synchronism. The motor speed had to be set within a tolerance, rather like our floppy disk crises.

It is also necessary to point out that the 32 control characters hi-1Fh are not restricted to the ASCII definitions. Tandy, and probably others, use some of these val-

ues to control the screen cursor when sent to the video driver. Thus a value which clears a line on the screen may produce some other response if it is sent to a printer or to some other computer. In fact this variation in the meaning of the control codes is part, but not all of the reason why you must have especially written printer drivers for most large programs, especially on word processor programs.

A lot of thought went into the design of the ASCII code, most of whose features are quite useful today. Some are more of the historical curiosity type, such as the use of character 127 (7Fh) as the delete code. It's basis is the paper tape era. The only way to correct an error in paper tape was to backup and repunch the erroneous holes with a character with all bits set, which of course was the value 7Fh. Some software today still uses 7Fh as its delete code.

The first 32 decimal values in the code are reserved for "control" codes, most of which were uniquely related to the teletype mechanical functions. All bits unset, 00h, could result from a disconnected line, and so this value today is a null value, nothing happens. The next 31 values do mechanical things in teletypes. The most popular values for computer purposes are 10 and 13, 0Ah and 0Dh, respectively. These are the linefeed (LF) and carriage return or ENTER (CR) characters. Top of form is signalled by TOF, 0Ch. Not all printers can respond to 0CH, but fortunately HP laserjets do, among others. Another useful item is the BELL, 07h. On a teletype, this was a physical bell, which could be dinged to get the remote operator's attention. Today this is usually a beep produced electronically in our printers and often our computers, as well. Another valuable item is the backspace, BS, 08h. I believe every printer can respond to this one. Some terminal programs depend upon XON and XOFF to start and stop data flow from a remote station. The values here are 11h and 13h, respectively. Finally, most printers have adjustable parameters selected with "escape codes". The decimal value 27 or 1Bh is ESC.

The numerals 0 through 9 have the ASCII codes 48 through 57. What is significant about these? In hex notation, they are 30h - 39h. If you strip off bits 5 and 6, the value remaining is the binary value, all ready for use in arithmetic functions. The reverse is equally true. Set bits 5 and 6, and you have converted the binary value to the printable ASCII values, already to go to the printer or video.

The alphabet comes in upper case and lower case values. 41h - 5Ah are the upper case values for the alphabet. Add 20h to the upper case value of a letter and presto you have the lower case value. And how do you add 20h? Very simple, just turn on bit 5 of the upper case letter value. Thus it is easy to make text input all upper case or all lower case when required by simply setting or unsetting bit 5.

The remaining 128 values which an 8 bit byte can take are assigned by various manufacturers at their own proprietary discretion. They are frequently used for graphics characters, but don't look for consistency, you will be disappointed.

Miscellaneous Items

Users of XMODEM, a file transfer function developed by Ward Christianson, know it includes a parity checking scheme referred to by its two options, checksum and CRC. Checksum was originally the only provision in XMODEM for parity checking. This scheme added up all the bits of all the 128 characters which could be transmitted in a data block, and sent this sum as a two byte value to the receiving end. The receiving end also added up all the bits in a block, and then compared its checksum with that received from the sending end. If they agreed, XMODEM then went on to the next block. Later, Ward added the CRC capability to XMODEM which is less likely to allow a bad block to be passed through a noisy circuit as a good one. The letters stand for Cyclical Redundancy Check, but I don't know the algorithm employed. Suffice it to say that it is statistically better than "checksum", and it is generally used now by all users of XMODEM. Current versions of XMODEM allow the sending end to select CRC or checksum at the operator's option, and the receiving end will be told of the choice and will use the corresponding algorithm. The default is usually CRC.

Before you challenge me, yes, I know XMODEM has been extended to use larger blocks of code. 128 was the original value because XMODEM was originally written in the days of CP/M, and CP/M uses 128 byte blocks. There are other options today including YMODEM and others which use larger blocks for faster transfer. Since this is only a footnote to ASCII, I will not go any further with XMODEM.

BEAT THE GAME

by Daniel Myers

THE CURSE OF CROWLEY MANOR

THE CURSE OF CROWLEY MANOR is the second in the series of "Other Ventures" from Adventure International. Like the others, there is a minimum of description (and clues) in the rooms through which you travel. This walkthrough will examine all the locations and objects, even if they aren't instrumental in solving the game. That way, you'll get the whole flavour of the adventure.

You start in a large office with some furniture, and a closed door to the east. You might as well have a look around, because you can't do anything (including leave the room) until the telephone rings, and you answer it. So, type <I> for inventory and have a LOOK AT the objects that you are carrying.

You can't EXAMINE anything, you can only LOOK. LOOK AT THE CALENDAR and NAMEPLATE on the desk, and have a LOOK OUT the window. Now you know who and where you are! You're Inspector Black of Scotland Yard in London, on the night of April 2, 1913.

Finally, the telephone rings and you ANSWER it. Officer Strade tells you to get to CROWLEY MANOR, because there has been a murder. Now you can OPEN THE DOOR to the east, and when you do, you find yourself on a landing, with a hallway heading south. GO SOUTH <S>, and you will end up on a brick street in front of Scotland Yard.

If you LOOK AT THE STREET, you will discover that a hansom cab is parked there, and the driver is smiling at you. This looks promising, so GO TO THE CAB. (Note that in some versions of the game, you cannot "GO TO THE CAB", you must "CLIMB IN"). If you TALK, the friendly driver will ask you where you are going. Why, CROWLEY MANOR, of course! (You will find that throughout the game, typing TALK or LISTEN will elicit information that will help you to solve the mystery.)

LOOK AT THE DRIVER, and you will find a small VIAL. GET THE VIAL...it's HOLY WATER and will come in handy, later. While the cab is driving through the streets of London, have a LOOK as you pass some familiar landmarks, and eventually pull into the driveway that leads into CROWLEY MANOR.

GET OUT of the cab, and the driver will tell you that he has taken 10 of your 50 shillings. You are on the porch, and if you LOOK, you will see Police Inspector Harbour. TALK, and he will tell you that the body is in the kitchen and Officer Strade is inside. CLIMB THE STEPS and you will end up in the plush entry hall, which contains a large, locked cabinet.

From now on, you should type LOOK when you enter a room, and it will list the contents. When you type LOOK AT (a specific object), you will get some of the details that are included in this walkthrough.

Now's the time to get rid of some of the baggage that you don't need: Drop the I.D. CARD, and the 40 SHILLINGS. If you were to quit or get killed, you would be returned to your office with all possessions intact. To get back to CROWLEY MANOR, you would need your money with you, but since this won't happen, you don't need to lug it around.

GO WEST <W> and then <S>. This is the elegant music room, lined with old portraits. They stare at you with fiendish, inhuman eyes. You find an old victrola and a piano. LOOK AT THE VICTROLA, and you will see the crank. Try turning the crank. Hmmm...it's stuck! Have a LOOK AT THE CRANK and you will find a GOLD KEY. You don't need the GOLD KEY to solve the adventure, but take it anyway. Now the crank is unstuck and you can TURN it. LISTEN.

Ahhh...MOZART!! Go <N> twice, into the dimly lit room. A small figure huddles in the dark, but you can't see much. The figure is on the floor. TALK to him. It is DAVONN, and NO MAN has murdered his master.

GO EAST <E> into the study. It's full of old books and a desk. There is a silver book on the desk, which you can READ (Note that some versions of the game require you to "READ" twice). You learn that "the demon" is trapped in the house. Brrrrr!!! Open the desk, and get the CRYSTAL BALL. Go <W> (you will find Davonn has been cruelly murdered!), <S>, and <E>.

If you want, unlock the large locked cabinet with the GOLD KEY. Officer Strade falls out! Now you have a personal stake in finding the demon of CROWLEY MANOR!! Drop the GOLD KEY and GO EAST. You are at the south end of a long hall. If you LOOK, you will find a statue

which turns out to be a WHITE ELEPHANT. It is, so forget about it.

GO NORTH again. You are in the centre of the hall. A tremendous force thrusts you against the wall, and there is a hideous smell. (This only happens if you LOOK. Without doing so, it would be a rather dull adventure indeed!) GO NORTH again to the north end, and CLIMB THE STAIRWAY there.

You are in the kitchen. Blood splatters the walls and floor, but there is no body! If you LOOK again, you will discover a brown, slimey GROWTH on the floor. GET THE GROWTH and GO WEST, into the exquisite dining room.

There is a china cabinet which you cannot open, and a marble table here. LOOK AT THE OAK TABLE. You must do this, to discover the delicious looking FOOD. DROP THE GROWTH and LOOK. It slides over to the plate, devours the food, and begins to grow. LOOK again. The GROWTH shoots under the china cabinet, which falls over with a crash. What have we here?! A LETTER OPENER and a HAND AXE! Get Them both. Return to the parlor: <E> <S> <S> <S> <W> <W>.

Here you will find a Rosewood chest, screwed shut. But you have a LETTER OPENER! UNSCREW THE SCREWS, and OPEN THE CHEST. You will find a CRUCIFIX and an OLD NOTE. Get Them. When you read the note, you will see that it says "5271". Remember this, and drop the OLD NOTE, and the LETTER OPENER. Go back to the dining room <E> <E> <N> <N> <CLIMB STAIRS> <W>. GO NORTH to the food pantry. A closed door is to the east. OPEN THE DOOR, and you will be in a short E/W hall. GO EAST to the small storage room.

If you LISTEN, you will hear a noise behind the plywood wall. CHOP A HOLE IN THE WALL WITH THE AXE. CLIMB THROUGH THE HOLE. You are in a darkened room with scientific instruments. LOOK, and a tremendous voice will boom out: "YOU HAVE NOT THE POWER TO FACE ME YET. BE WARNED..." Wow, the plot thickens!

COLLECTING THE WEAPONS

LOOK AT THE SCIENTIFIC INSTRUMENTS. There is an ancient book here. READ THE BOOK, and you will learn that "GAFALA ALONE CAN HELP." Who is GAFALA? You'll GO WEST to the musty room. There is a door here, with a numeral lock.

Luckily, you have sacraments (CRUCIFIX and HOLY WATER), or an icy creature would appear and kill you! SET THE DIAL TO 5271 (the number on the OLD NOTE). The door will click. OPEN THE DOOR.

You are in a damp, brick walled room. There is a horrible stench. LOOK (several times): A powerful force slams you against the wall. Something screams. You are thrown flat on the floor. A voice bellows, "SOON YOU'LL BE MINE!!" The room cools and fills with smoke. Your crucifix is red hot. An evil smelling smoke drifts through the room. It forms frightening shapes.

This is getting exciting, isn't it?!

CLIMB THE STAIRWAY that lies to the east. You are at the south end of a N/S hall. Sound familiar? GO NORTH twice, then <W> through the doorway. You are in a stark room of red brick. This appears to be a dead end! Time to call in the reinforcements! Type GAFALA. A wall falls, and a voice shouts: "I AM HERE!" LOOK! The west wall has collapsed, revealing a stairway. CLIMB THE STAIRWAY.

You are in a brilliant crystal room. A towering figure of white stands in the centre. A glowing mist surrounds the figure. TALK TO GAFALA. You learn that he is the evil demon's brother, and that two paths leave the room to the north and the south. It's a good thing that you had GAFALA to help you, because there's no indication of any exits from this room!

GO SOUTH to the east end of an E/W hall. GO WEST to the crypt. If you LOOK, you will discover that THE HORROR IS HERE - THE DEMON! It screams: "WHERE IS YOUR WEAPON, MAGGOT?" Well...it seems that you need to arm yourself!

Whatever you do, don't LOOK again, or you will end up back in the darkened room with DAVONN. This wouldn't be the end of the world, but you would have to retrace your steps.

GO WEST to the dim, shimmering room. A ghostly figure sits at a piano. It resembles the portraits back in the music room! TALK, and it will tell you that it knows one composer and will play if you name him. Know any composers? How about MOZART! Sure enough, the figure plays and a stairway appears to the north. CLIMB THE STAIRWAY.

Oh oh...another apparent dead end! This is a large deserted room with nothing in view. Luckily, you DO have five senses. LISTEN...and you will hear a noise above your head. LOOK UP, and then CLIMB THE ROPE you see hanging from the ceiling.

You are in a damp, musty passage with a rope hanging through a hole in the floor. Why not have a LOOK? Sure enough, a GOLD SHIELD appears. GET THE GOLD

SHIELD. If you look again, you will notice a light to the east. GO EAST.

You are in a great silver room, with a circular depression on the floor. I'll bet you're carrying something that would fit that perfectly! DROP THE CRYSTAL BALL and LOOK. You see a vision of a sword and a beautiful fountain. LOOK again, and a magic SWORD will appear.

GET THE SWORD and return to the crystal room where GAFALA is: <W> <CLIMB DOWN ROPE> <S> <E> (If you like, have a look when you get to the crypt, and you will find a SILVER CLUB and half decayed corpses on the floor. Don't bother with either!) Just for fun, GO NORTH, LOOK and GO SOUTH. Keep going to the crystal room: <E> <N>.

CHASING THE DEMON

Now, take the other path NORTH <twice> and you will find yourself in a wide, dark, smelly pit. A low wailing moan permeates the darkness. LISTEN. BUT IS IT CLEANSED?" Hmmm...something else to keep in mind! CLIMB THE STAIRWAY to the north.

You are in a stone room strewn with bones. A huge, ugly rat is here. Unfortunately , you can't kill the rat until it removes a chunk of flesh from your arm ! Don't do anything but LOOK. It will oblige you, and you can KILL THE RAT WITH THE REVOLVER. If you try to use any other weapon, it will rip out your throat! OPEN THE DOOR to the west, and you will be in a smelly dirt pit with fungus growing on the walls. A stairway to the north is blocked by wooden beams. CHOP THE BEAMS WITH THE AXE, and then you can CLIMB THE STAIRS.

You are in a very eerie, black walled chamber. Strange, magical runes are drawn on the floor. A ghost-like image of GAFALA is here. TALK TO GAFALA, and he will give you some important advice: "YOU HAVE BUT ONE MOMENT TO STRIKE. STRIKE ONLY THEN."

Now comes a rather annoying part of the game. There is no indication of any other exit from the room, except the golden doorway to the north. If you GO SOUTH, you will end up back in the pit.

Yet, you must GO EAST!! Here you will find a brilliant green room with a beautiful fountain. It flows with green liquid and energy waves. CLEANSE THE SWORD in the fountain. You will be rewarded with a surge of power and a low moan from the north. If you hadn't gone east and cleansed your sword, than all would be lost!! Rather a significant thing to leave hidden away, isn't it? (Note that in the graphic versions of the game (for another computer), the doorway to the east is visible in the picture)

Return to the eerie chamber and through the golden doorway: <W> <N>. You end up in a gigantic cavern, stretching out of sight to the east and west, and filled with a sickening odor. Just head straight NORTH to the Gates of Hell. IT IS UPON YOU!

LOOK (several times): The demon hurls you through the air. Your body slams against a wall. Your face is red hot from the heat of its breath. It howls, "THANK YOU FOR DROPPING IN, FOOL!" It slams you into a small passage and laughs, "YOU'VE BEEN DUPED! YOU CAN'T KILL ME!" Your body shakes with convulsions as the demon tries to crush you, but you are shielded! It touches your white hot sword and screams with pain. (Good thing you cleansed it, or you wouldn't have gotten even this far!) It belches a sheet of flame which engulfs your body. You're getting a little perturbed, but you're biding your time!

The demon stares into your face. "JOIN ME...SPAWN OF DIRT, AND SHARE THE POWER THAT IS MINE." The time has come!!! Without further ado, KILL THE DEMON WITH THE SWORD.

Your CRT explodes with light!! Congratulations! You have defeated the demon of CROWLEY MANOR!

HEAVYWEIGHT BOXING

Programming Technique

Model 4 - Basic

by Lance Wolstrup

Basic is a wonderful programming language. It is easy to learn and most tasks can be accomplished with a minimum of effort. However, as Mickey Mepham of Charles City, VA points out in his letter from this issue's mail page, a little help can be useful when dealing with screen graphics.

Because the Model 4 screen is not memory mapped, as is both Model I and III, Basic handles the display in an unsatisfactorily slow manner. To be sure, the screen updates text or graphics with reasonable speed moving horizontally. But, when updating graphics vertically, a comparison to 'turtle'-graphics is almost appropriate.

This lack of speed is not a detriment to the Basic programming language, though. When handling a task inadequately, Basic usually has an alternative method of solving the problem. This is certainly true in this case. Since Basic handles graphics too slowly, we will use Basic's ability to incorporate machine language routines.

Now, whenever I mix Basic and machine language, I use a method called 'string-packing' to place the machine code in memory. This eliminates the need to reserve memory space for the routine. But, on the other hand, it also necessitates that the code is completely relocatable.

The routine should be flexible enough to allow the calling Basic program to pass four parameters - the vertical and horizontal position of the cursor, as well as the length and height of the rectangle. This allows the graphic to be placed anywhere on the screen and be of any size.

The machine code listing below does just that. It begins by storing the desired cursor position in register HL (vertical position in H, and horizontal position in L). The listing shows the position to be 0000H. This will be changed when the calling Basic program passes the cursor location parameter by the way of POKEs. The routine then uses the @VDCTL SuperVisor Call to move the cursor to the desired position.

Next, register BC is loaded with the desired length and height of the rectangle (B is the length, C is the height). Again, the value set forth in the listing is 0000H. This will also be changed when the calling Basic program passes the length and height parameters.

At this point the cursor is where we wish for the top left corner of the rectangle to be, so we can now display chr$(151), followed by a loop to display a series of chr$(131)'s, which is followed by chr$(171), the top right corner.

The top of the box is now complete, so we move the cursor back to the top left corner, move it down one line, beginning a loop that displays left and right sides of the box.

Finally, the cursor is moved to the left side of the box, moved down one line, and we display chr$(181), which is the bottom left of the box. A loop then displays a series of chr$(176)'s, followed by chr$(186), the bottom right of the box. Having done its job, the routine then returns to Basic.

The machine language routine is 95 bytes, which is rather lengthy. You will notice that several pieces of the code repeat. In a normal machine language program, I would have written the repeating code in subroutines, thus shortening the program significantly. It is not possible to do that here, because we will store the entire routine in a Basic string and, since strings move around in memory, we must only use code that will execute no matter where in memory it is located. Subroutines within such a routine is a no-no, as they become absolute addresses.

```
        LD      HL,0000H  ;cursor position will be
                          ;poked here
        LD      A,15      ;@vidctl
        LD      B,3       ;function 3 - move cursor
                          ;to position specified by HL
        RST     40
        LD      BC,0000H  ;box length in B;
                          ;box height in C; will be
                          ;poked from basic
        PUSH    BC        ;save length & height
        LD      C,151     ;chr$(151) - top left of box
        LD      A,2       ;@dsp
        RST     40        ;display chr
        LD      C,131     ;chr$(131) - top of box
LOOP1   LD      A,2       ;@dsp
        RST     40        ;display chr
        DJNZ    LOOP1     ;loop until B=0
        LD      C,171     ;chr$(171) - top right of box
        LD      A,2       ;@dsp
        RST     40        ;display chr
        POP     BC        ;restore length & height
        PUSH    BC        ;save it again
        LD      D,B       ;copy length to D
        LD      B,C       ;copy height to C
LOOP2   PUSH    BC        ;save height
```

```
        PUSH    DE      ;save length
        INC     H       ;increment vertical cursor
                        ;position
        LD      A,15    ;@vdctl
        LD      B,3     ;function 3 - move cursor
                        ;to position specified by HL
        RST     40
        LD      C,149   ;chr$(149) - left side of box
        LD      A,2     ;@dsp
        RST     40      ;display chr
        POP     DE      ;restore length
        POP     BC      ;restore height
        PUSH    BC      ;save height
        PUSH    HL      ;save cursor position
        PUSH    DE      ;save length
        LD      A,L     ;copy horizontal cursor
                        ;position to A
        ADD     A,D     ;add length
        LD      L,A     ;copy result back to L
        INC     L       ;next horizontal cursor
                        ;position - cursor is now
                        ;at right side of box
        LD      A,15    ;@vdctl
        LD      B,3     ;function 3 - move cursor to
                        ;position specified by HL
        RST     40
        LD      C,170   ;chr$(170) - right side of
                        ;box
        LD      A,2     ;@dsp
        RST     40      ;display chr
        POP     DE      ;restore length
        POP     HL      ;restore cursor position
        POP     BC      ;restore  height
        DJNZ    LOOP2   ;loop until B=0
        INC     H       ;increment vertical cursor
                        ;position
        LD      A,15    ;@vdctl
        LD      B,3     ;function 3 - move cursor to
                        ;position specified by HL
        RST     40
        POP     BC      ;restore length
        LD      C,181   ;chr$(181) - bottom left of
                        ;box
        LD      A,2     ;@dsp
        RST     40      ;display chr
        LD      C,176   ;chr$(176) - bottom of box
LOOP3   LD      A,2     ;@dsp
        RST     40      ;display chr
        DJNZ    LOOP3   ;loop until B=0
        LD      C,186   ;chr$(186) - bottom right
                        ;of box
        LD      A,2     ;@dsp
        RST     40      ;display chr
        RET             ;return to basic
```

OK., now that we have written the machine language routine, we need to translate it into code that Basic can understand which, of course, is DATA statements. When the machine code is assembled, LD HL,0000H is turned into 21 00 00; LD A,15 becomes 3E 0F, and LD B,3 changes to 06 03, etc.

All these numbers are hexadecimal values. To make it easier to use for us to use them in data statements, we will translate the values from hexadecimal to decimal numbers. Thus, 21 00 00 become 33,0,0. The hex numbers 3E 0F become 62,15, etc.

The translation is now:

Hex	Decimal
21 00 00	33,0,0
3E 0F	62,15
06 03	6,3
EF	239
01 00 00	1,0,0
C5	197
0E 97	14,151
3E 02	62,2
EF	239
0E 83	14,131
3E 02	62,2
EF	239
10 FB	16,251
0E AB	14,171
3E 02	62,2
EF	239
C1	193
C5	197
50	80
41	65
C5	197
D5	213
24	36
3E 0F	62,15
06 03	6,3
EF	239
04 95	14,149
3E 02	62,2
EF	239
D1	209
C1	193
C5	197
E5	229
D5	213
7D	125
82	130
6F	111
2C	44
3E 0F	62,15
06 03	6,3
EF	239
0E AA	14,170
3E 02	62,2
EF	239
D1	209
E1	225

C1	193
10 DB	16,219
24	36
3E 0F	62,15
06 03	6,3
EF	239
C1	193
0E B5	14,181
3E 02	62,2
EF	239
0E B0	14,176
3E 02	62,2
EF	239
10 FB	16,251
0E BA	14,186
3E 02	62,2
EF	239
C9	201

So far, so good! We now have the values to store in DATA statements; the only thing we need is a program that will use the data. BOXDEMO/BAS (see listing below) will demonstrate how to use the data to create the machine language routine, as well as how to execute the routine from within Basic.

Line 10 sets up variable Z$ to hold 95 spaces (Z$=STRING$(95,32). This string will eventually hold our 95 byte machine language routine. Line 10 branches off to the subroutine in line 70 where variable Z is computed to hold the actual address of Z$. Coming back from the subroutine, line 10 performs the loop FOR X=0 TO 94:READ ML: POKE Z+X,ML:NEXT. This loop reads the data (our machine language routine) from lines 15 and 16 into Z$. Now, whenever we want to execute the routine, all we have to do is find the current address of Z$ and then CALL that address. The subroutine in lines 70-72 does just that.

BOXDEMO/BAS emulates the drop-down, PC-type of menu found in FastTerm. You will notice that, for the most part, it behaves just as if you were in FastTerm. You may call on a submenu by using the <CLEAR><letter> combination and, when already in a menu, you may use the <LEFT-ARROW> or <RIGHT-ARROW> to move to other menus. The real FastTerm menu is written in 100% machine language, so everything there happens instantly. This demo is written in Basic so, while our machine language routine displays the boxes with the speed of lightning, the text inside the boxes (not handled by an ML routine) display with the normal slow speed of Basic. But that is a problem we will tackle in a future article!

BOXDEMO/BAS

```
10 DEFINT A-Y:DIM MN(9,11),MN$(9,11):
Z$=STRING$(95,32):PRINT CHR$(15):GOSUB 70:
FOR X=0 TO 94:READ ML:POKE Z+X,ML:NEXT
11 FOR X=1 TO 9:READ HE(X):NEXT:
FOR X=1 TO 9:READ LE(X):NEXT:
FOR X=1 TO 9:READ TB(X):NEXT:
FOR X=1 TO 9:FOR Y=1 TO HE(X):
READ MN$(X,Y):
NEXT:NEXT
14 MN$=" Echo  Uart  Buffer  Dialer  Scripts  Library
Transfer  Status  Who "+CHR$(31):GOTO 100
15 DATA 33,0,0,62,15,6,3,239,1,0,0,197,14,151,62,2,
239,14,131,62,2,239,16,251,14,171,62,2,239,193,197,80,
65,197,213,36,62,15,6,3,239,14,149,62,2,239,209,193,197,
229,213,125,130,111,44,62,15,6,3,239,14,170
16 DATA 62,2,239,209,225,193,16,219,36,62,15,6,3,239,
193,14,181,62,2,239,14,176,62,2,239,16,251,14,186,62,2,
239,201
17 DATA 3,6,7,8,6,1,11,6,2,16,22,23,54,43,35,25,12,16,0,
9,16,24,34,43,53,65,62,"<R>emote OFF",
"<L>ocal OFF","<P>rinter","<B>aud Rate : 2400",
"<W>ord Len  : 8","<S>top Bits : 1",
"<P>arity   : NONE","<M>ask Bit8 : OFF",
"Video Scroll: ON"
18 DATA"[ ] <O>pen Buffer","[*] <C>lose Buffer",
"  <S>ave Buffer","  <V>iew Buffer",
"Buffer Size [ 0K]","Memory Free [ 73K]","File: ....."
19 DATA"<A>rea : 818  California,San Fernando Val
02:25:45",
"<E>nter Phone number:        <B>uild # Entry:",
"<T>ime redial    : 30     <L>imit Redial : 50",
"<R>edial #       :        Redial Attempts: 0"
20 DATA"Prefix Dial Code #1: ATDT 1\",
"Prefix Dial Code #2:",,
"1> #1; 213 664-5056\TRSuretrove BBS\"
21 DATA"<S>ave Directory/Scrips to Disk",
" Script Size [.....]     Status [.....]",
"<C>lock ON/OFF  <Z>ero Clock",
"<E>nter Script Key#:",,"1> TRSure/scr\"
22 DATA"<C>md: ....."
23 DATA"<Y>modem BATCH Receive",
"<U>pload ASCII File","<D>ownload ASCII File",
"<O>utput File (1Kxmdm)","<I>nput File (1Kxmdm)",
"<S>end File (Xmodem)","<R>eceive File (Xmodem)"
24 DATA"File: ....."","Block: ...  Data:",
"Total Blocks: ....","Total Size:"
25 DATA"Bell : ON","CR+LF: OFF","Open : 18",
"Close: 20","  LF: OFF","AutoS: OFF"
26 DATA" This Demo by","Lance Wolstrup"
70 Z=PEEK(VARPTR(Z$)+2)*256+
PEEK(VARPTR(Z$)+1):IF Z<0 THEN Z=Z+65536!
71 IF F THEN POKE Z+1,H:POKE Z+2,V:
POKE Z+9,HE-2:POKE Z+10,LE-2:CALL Z
72 RETURN
100 F=1:MN=0:CLS
110 PRINT@0,MN$
120 I$=INKEY$:IF I$="" THEN 120
130 IF I$=CHR$(9) THEN IF MN=9 THEN MN=1:
```

```
GOTO 200 ELSE MN=MN+1:GOTO 200
140 IF I$=CHR$(8) THEN IF MN=1 THEN MN=9:
GOTO 200 ELSE MN=MN-1:GOTO 200
145 IF I$=CHR$(129) THEN IF TP=0 THEN MN=1:
GOTO 200 ELSE MN=TP:GOTO 200
150 IF I$=CHR$(197) THEN MN=1:GOTO 200
151 IF I$=CHR$(213) THEN MN=2:GOTO 200
152 IF I$=CHR$(194) THEN MN=3:GOTO 200
153 IF I$=CHR$(196) THEN MN=4:GOTO 200
154 IF I$=CHR$(205) THEN MN=5:GOTO 200
155 IF I$=CHR$(204) THEN MN=6:GOTO 200
156 IF I$=CHR$(212) THEN MN=7:GOTO 200
157 IF I$=CHR$(193) THEN MN=8:GOTO 200
158 IF I$=CHR$(215) THEN MN=9:GOTO 200
159 TP=MN:MN=0:GOTO 110
200 TP=MN:V=0:PRINT@V,MN$:V=0:H=TB(MN):
LE=LE(MN)+2:HE=HE(MN)+2:GOSUB 70:H=H+2:
FOR V=1 TO HE(MN):PRINT@(V,H),MN$(MN,V);:NEXT
210 GOTO 120
```

VISICALC REFERENCE GUIDE

from the TRSTimes vault

We have had numerous request for articles about the grand-daddy of all spreadsheets - VISICALC. While the interest for this material was high, the submissions were low; as a matter of fact, we had absolutely no submissions covering this subject.

Well, never let it be said that TRSTimes does not cater to the whims of the readers; so, aiming to please, I went down to the TRSTimes vault and made an extensive search through the archives. With the efficient and speedy TRSTimes filing system, it did not take long before I found what I was looking for.

(*Actually, my wife yelled at me to clean up the garage and, in the process of obeying the higher authority, I accidentally ran across a set of disks with long-forgotten files*).

Looking through the goodies, I found a 'quick reference' text-file covering VISICALC. I feel this will be of interest to many of the readers, especially the new arrivals to the TRS-80 world; so, without further ado, TRSTimes presents the 'quick reference' guide to our favorite spreadsheet. I hope you will find it useful.

MOVING THE CURSOR

ARROW KEYS	The arrow keys move the cursor left, right, up or down.
;	If two windows, the semicolon moves the cursor from one window to the other.
>	Go To command. Type the coordinates of the entry where you want the cursor to go; end with <ENTER>.
<SHIFT><CLEAR>	Clears the cursor.
<SHIFT><0>	Switches the cursor between blinking and steady.

THE CLEAR KEY

<CLEAR>	is used to recover from simple typing mistakes. It usually erases the last thing you typed. If you press CLEAR enough times, it will abort what you are doing and return VisiCalc to a blank prompt line.

THE BREAK KEY

<BREAK>	This is equivalent to pressing CLEAR enough times to return VisiCalc to a blank prompt line. It will also abort most operations, such as file input and printing.

SETTING A LABEL ENTRY

Label entries start with the letters (A-Z) or with a quote character ("). Terminate entering a label entry by pressing an arrow or ENTER. Correct errors by pressing the CLEAR key. The prompt line will say LABEL while a label entry is being typed.

SETTING A VALUE ENTRY

A value entry displays the calculated value of the expression stored at the entry. Expressions consist of numbers, coordinates of other value entries (value references), functions (such as @SUM), arithmetic operators (+ - * / [ ) and/or parentheses. Exponent ([ ) is typed by pressing <SHIFT><@>. Expressions are evaluated strictly from left to right except as modified by parentheses. You must start and expression with a +, a digit (0-9), or one of the symbols @ - (. or #. The prompt line will say VALUE while an expression is being typed. Terminate entering an expression by pressing an arrow key or ENTER. Errors can be corrected by pressing the CLEAR key. Examples of expressions are:

12.34	A normal number
.1234E2	A number in scientific notation
2+2	An arithmetic expression
+B4	A value reference
2*B4	An expression with a value reference
2*(3+4)	An expression with parentheses

If you press <!> while entering an expression, VisiCalc will calculate the value of the expression so far and replace the expression on the edit line with the number which results from the calculation.

VALUE REFERENCES

An expression at one entry can refer to the value of another entry, and the value of such an expression can

be automatically recalculated when the value of the other entry changes. Value references are allowed in expressions wherever numbers are allowed. A value reference is made by either typing the coordinate of the desired entry (such as B5), or by "pointing" to the entry with the cursor (in this case, the coordinate will be "typed" automatically by VisiCalc). If an expression starts with a value reference, it must be preceded by a + character.

In order to insert the current value of another entry into an expression as a number, which will be unaffected by later changes to the other entry, type a value reference followed by the character # (e.g., B5#). If # is used by itself, it will be replaced by the current value of the expression stored in the entry you are changing.

FUNCTIONS

@SUM (list)	Calculates the sum of the values in (list). See LISTS, below.
@MIN (list)	Calculates the minimum value in (list).
@MAX (list)	Calculates the maximum value in (list).
COUNT(list)	Results in the number of non-blank entries in (list). Maximum number of entries in the list is 255.
@AVERAGE (list)	Calculates the average of the non-blank values in (list). Maximum number of entries in the list is 255.
NPV(dr,range)	Calculates the next present value of the cash flow in (range), discounted at the rate specified by expression (dr). The first entry in the range is the cash flow at the end of the first period; the second entry is the cash flow at the end of the second period, etc. See ENTRY RANGES, below.
@LOOKUP (v,range)	Compares the value of (v) to the values of successive entries in (range), and selects a corresponding value from the first column or row immediately to the right or below the entries in (range), as the result of the function. The values in (range) are normally in ascending order, and the result is the value corresponding to the last entry in (range) that is less than or equal (v) before an entry greater than (v) is found. If the first entry in (range) is greater than (v), the result of the function is NA
@NA	Results in a "Not Available" value that makes all expressions using the value display as ERROR.
@ERROR	Results in an "Error" value that makes all expressions using the value display as ERROR.
@PI	Results in 3.1415926536.
@ABS (v)	Results ts in the absolute value of (v).
@INT (v)	Results in the integer portion of (v).
@EXP (v)	Returns e (2.71828...) to the v power.
@SQRT (v)	Results in the square root of v.
@LN	Returns the natural log (base e) of a number.
@LOG (v)	Results in the logarithm (base 10) of v.
@SIN (v) @ASIN (v) @COS (v) @ACOS (v) @TAN (v) @ATAN (v)	Returns the appropriate trigonometric functions of the value. Calculations are done in radians.

EXAMPLES OF FUNCTIONS

@SIN(C7*@PI/180)

@SUM(B4...B15)

@MIN(100,F4...F11,@SUM(B4 . . . B15))

@MAX(0,F4-F5)

@NPV(.15,B4 . . . F4)

ENTRY RANGES

An entry range consists of a number of entries that are next to each other in a row or column, such as B2, B3, and B4, or B2, C2, D2, and E2. You enter an entry range

by specifying the coordinate of the first entry in the range, then typing an ellipsis (...-you need only type the first period; VisiCalc will fill in the others), and then specifying the last entry. For example, the entry ranges just mentioned would be B2...B4 and B2...E2, respectively.. Coordinates are specified by either typing them, or "pointing" to the desired entry with the cursor.

LISTS

A list consists of a series of expression and ranges separated by commas. See the examples of lists in Examples of Functions, above.

COMMANDS

/B — Sets an entry to blank. Doesn't take effect unless you follow it with an arrow or ENTER. Does not affect /F formats set at the entry.

/C — Clears the sheet, setting all entries to blank, resetting formats, windows, titles, etc., but doesn't reset any remembered file name qualifiers. VisiCalc will wait for you to type a Y to confirm that you indeed want to clear the sheet.

/D — Deletes the row (/DR) or column (/DC) on which the cursor lies.

/F — Sets the display format of an entry to one of the following formats:

- /FG General
- /FI Integer
- /F$ Dollar and cents
- /FL Left justified (used for values)
- /FR Right justified (used for labels)
- /F* Graph
- /FD Default. Resets an entry to use the global default format instead of an explicit format set with a /F command.

/G — Global commands. These apply to the entire sheet or window.

- /GC Sets the column width. Requests a number greater than 2; end with ENTER. The column width can be changed on a per window basis.
- /GF Sets the global default format that determines the display format of all entries without explicit format settings set with a /F command. Requests one of the same display formats used by the /F command. Note that /GFD is the same as /GFG. The global default format is a per window setting.
- /GO Sets the order of recalculation to be down the columns starting at entry A1 (/GOC), or across the rows starting at entry A1 (/GOR).
- /GR Sets the recalculation to be automatic (/GRA) or manual (/GRM). You can always cause a manual recalculation of all entries by pressing the <!> key.

/I — Inserts a row (/IR) or a column (/IC) just above or to the left of the row or column on which the cursor lies.

/M — Moves an entire row or column to a new position. Prompts you to move the cursor from the row or column which you want to move to the destination row or column just before which the entries moved should reappear. End with ENTER.

/P — Print command. See Printing, below.

/R — Replicate command. See Replicate, below.

/S — Storage command.
See Storage Commands, below.

/T — Sets a horizontal title area (/TH), a vertical title area (/TV), sets both a horizontal and vertical title area (/TB), or resets the window to have no title areas (/TN).

/V — Displays VisiCalc's version number on the prompt line. The version number will disappear as soon as you type something else.

/W — Window control. Splits the screen into two windows at the cursor position. (/WH for horizontal, /WV for vertical), or returns the screen to one window (/W1). Windows may be synchronized (/WS) or returned to unsynchronized (/WU).

/- — Repeating label. Requests the contents of a label entry; end with an arrow or ENTER. The contents of the label will be repeated over and over to fill the entry, no matter what the column width.

PRINTING

The /P command lets you output to the printer.

1) Position the cursor at the upper left corner of the rectangle of entries that you wish to print and type /P.

2) VisiCalc will prompt for the type of device on which you print. You respond with P for a printer, R for RS-232, or F to specify a filename (see File Names below.)

3) If you press + at this point a carriage return will be output immediately. Lines printed by VisiCalc are usually terminated by RETURN. To have the RETURN followed by a LINE FEED character, type & at this point. To suppress the LINE FEED, type a minus (-) at this point.

4) If you want to output setup characters type ", then type the characters, and end with ENTER.

5) Move the cursor to (or type the coordinates of) the lower right corner of the rectangle of entries to be printed out, and press ENTER.

You may stop printing at any time by pressing BREAK.

REPLICATE

The replicate command makes copies of entries.

1) Position the cursor on the first entry that you wish to replicate, and type /R.

2) VisiCalc will ask for the coordinates of the source (what you want to replicate). If you are just replicating the current entry, press ENTER. If you want to replicate a range of entries, type an ellipsis and provide a coordinate to complete the entry range specifying the source, ending with ENTER.

3) VisiCalc will ask for the coordinates of the target (where you want the copies to go). This may be a single coordinate or an entry range. End the target with ENTER. If you are replicating a source range of entries, the first source entry will be replicated into the entire target range, and succeeding entries will be replicated into correspondingly succeeding target ranges.

4) If the expression being replicated contains value references, VisiCalc will ask you, for each value reference, whether it should not be modified (respond by typing <N>), or should always refer to the entry in the same relative position (type <R>).

FILE NAMES

Some of the VisiCalc commands prompt for a file name. You may repond in one of two ways. In the first way, you type a file name (with optional qualifiers, defined below) followed by ENTER. VisiCalc will use the file name that you type. In the second way, you do not type a full file name. You respond with a blank line, or optionally an extension (preceded by a slash) and/or specification for which disk drive you are referring to. You then press the right arrow key. VisiCalc will display the name of the first file with that extension (if provided) that it finds on the diskette. If that is the name of file that you wish to use, press ENTER; otherwise, press the right arrow key and VisiCalc will show you the name of each successive file on the diskette. When you have found the file name that you want to use, press ENTER. You may edit the file name before executing the command by typing additional characters to add to the name, and/or using the CLEAR key to erase part of it.

The following qualifiers are allowed:

1) extension (preceded by a slash)
2) password (preceded by a period)
3) disk drive number (preceded by a colon)
4) device specification (:P for print, :R for RS-232).

Note that Visicalc provides extensions by default:

/VC	for saved sheets (/SS and /SL commands)
/PRF	for printing a file (/PF command)
/DIF	for DIF files (/S# commands)

STORAGE COMMANDS

The /S commands let you save and load the current entries using a diskette, and exit from VisiCalc. The storage commands may be aborted by pressing BREAK.

/SS	Save all entries, titles, and window settings in a file. Prompts for a file name.
/SL	Load the contents of all entries that were saved in a file. This command does not blank out all entries before doing the load; if that is desired, use the /C command first. Prompts for a file name.
/SD	Deletes the specified file on the diskette. Prompts for a file name, then asks you to type a <Y> to confirm.
/S#S	Saves data in the Data Interchange Format. Prompts for a file name, then requests the lower right coordinate of the rectangle of entries to be saved. Move the cursor or type the coordinate, and then press ENTER. Finally, it asks whether the data is to be saved by rows (<R> or ENTER) or by columns (<C>).
/S#L	Loads data in the Data Interchange Format. Prompts for a file name, then asks whether the data is to be loaded by rows (<R> or ENTER) or by columns (<C>).
/SQ	Quits out of VisiCalc and returns to the operating system. Type <Y> to confirm.

ATTENTION TRSDOS 1.3. USERS!

ANNOUNCING "SYSTEM 1.5.", THE MOST COMPREHENSIVE 1.3. UPGRADE EVER OFFERED!

MORE SPEED!! MORE POWER!! MORE PUNCH!!

While maintaining 100% compatibility to TRSDOS 1.3., this DOS upgrade advances TRSDOS 1.3. into the 90's! SYSTEM 1.5. supports 16k-32k bank data storage and 4MGHZ clock speed (4/4P/4D). DOUBLE SIDED DRIVES ARE NOW 100% UTILIZED! (all models).

CONFIG=Y/N	CREATES CONFIG BOOT UP	FILEDATE=Y/N	DATE BOOT UP PROMPT ON or OFF
TIME=Y/N	TIME BOOT UP PROMPT ON or OFF	CURSOR='XX'	DEFINE BOOT UP CURSOR CHAR
BLINK=Y/N	SET CURSOR BOOT UP DEFAULT	CAPS=Y/N	SET KEY CAPS BOOT UP DEFAULT
LINE='XX'	SET *PR LINES BOOT UP DEFAULT	WP=d.Y/N (WP)	WRITE PROTECT ANY or ALL DRIVES
ALIVE=Y/N	GRAPHIC MONITOR ON or OFF	TRACE=Y/N	TURN SP MONITOR ON or OFF
TRON=Y/N	ADD an IMPROVED TRON	MEMORY=Y/N	BASIC FREE MEMORY DISPLAY MONITOR
TYPE=B/H/Y/N	HIGH/BANK TYPE AHEAD ON or OFF	FAST	4 MGHZ SPEED (MODEL 4'S)
SLOW	2 MGHZ SPEED (MODEL III'S)	BASIC2	ENTER ROM BASIC (NON-DISK)
CPY (parm,parm)	COPY/LIST/CAT LDOS TYPE DISKS	SYSRES=H/B/'XX'	MOVE/SYS OVERLAY(s) TO HI/BANK MEM
SYSRES=Y/N	DISABLE/ENABLE SYSRES OPTION	MACRO	DEFINE ANY KEY TO MACRO
SPOOL=H/B.SIZE	SPOOL is HIGH or BANK MEMORY	SPOOL=D.SIZE='XX'	LINK MEM SPOOLING TO DISK FILE
SPOOL=N	TEMPORARILY DISABLE SPOOLER	SPOOL=Y	REACTIVATE DISABLED SPOOLER
SPOOL=RESET	RESET (NIL) SPOOL BUFFER	SPOOL=OPEN	OPENS, REACTIVATES DISK SPOOLING
SPOOL=CLOSE	CLOSES SPOOL DISK FILE	FILTER *PR.ADLF=Y/N	ADD LINE FEEDS BEFORE PRINTING 0DH
FILTER *PR.IGLF	IGNORES 'EXTRA' LINE FEEDS	FILTER *PR.HARD=Y/N	SEND 0CH to PRINTER (FASTEST TOF)
FILTER *PR.FILTER	ADDS 256 BYTE PRINTER FILTER	FILTER *PR.ORIG	TRANSLATE PRINTER BYTE TO CHNG
FILTER *PR.FIND	TRANSLATE PRINTER BYTE TO CHNG	FILTER *PR.RESET	RESET PRINTER FILTER TABLE
FILTER *PR.LINES	DEFINE NUMBER LINES PER PAGE	FILTER *PR.WIDTH	DEFINE PRINTER LINE WIDTH
FILTER *PR.TMARG	ADDS TOP MARGIN to PRINTOUTS	FILTER *PR.BMARG	ADDS BOTTOM MARGIN to PRINTOUT
FILTER *PR.PAGE	NUMBER PAGES, SET PAGE NUMBER	FILTER *PR.ROUTE	SETS PRINTER ROUTING ON or OFF
FILTER *PR.TOF	MOVES PAPER TO TOP OF FORM	FILTER *PR.NEWPG	SET DCB LINE COUNT TO 1
FILTER *KI.ECHO	ECHO KEYS to the PRINTER	FILTER *KI.MACRO	TURN MACRO KEYS ON or OFF
ATTRIB:d.PASSWORD	CHANGE MASTER PASSWORD	DEVICE	DISPLAYS CURRENT CONFIG INFO

All parms above are installed using the new LIBRARY command SYSTEM (parm,parm). Other new LIB options include DBSIDE (enables double sided drive by treating the "other side" as a new independent drive, drives 0-7 supported) and SWAP (swap drive code table #s). Dump (CONFIG) all current high and/or bank memory data/routines and other current config to a disk data file. If your type ahead is active, you can (optional) store text in the type buffer, which is saved. During a boot, the config file is loaded back into high/bank memory and interrupts are recognized. After executing any active auto command, any stored type ahead data will be output. FANTASTIC! Convert your QWERTY keyboard to a DVORAK! Route printer output to the screen or your RS-232. Macro any key,even F1, F2 or F3. Load *01-*15 overlay(s) into high/bank memory for a memory only DOS! Enter data faster with the 256 byte type ahead option. Run 4MGHZ error free as clock, disk I/O routines are properly corrected! Spool printing to high/bank memory. Link spooling to disk (spooling updates DCB upon entering storage). Install up to 4 different debugging monitors. Print MS-DOS text files, ignoring those unwanted line feeds. Copy, Lprint, List or CATalog DOSPLUS, LS-DOS, LDOS or TRSDOS 6.x.x. files and disks. Add top/bottom margins and/or page numbers to your hard copy. Rename/Redate disks. Use special printer codes eg: LPRINT CHR$(3); toggles printer output to the ROUTE device. Special keyboard codes add even more versatility. This upgrade improves date file stamping MM/DD/YY instead of just MM/YY. Adds optional verify on/off formatting, enables users to examine *01-*15, DIR, and BOOT sectors using DEBUG, and corrects all known TRSDOS 1.3. DOS errors. Upgrade includes LIBDVR, a /CMD driver that enables LIBRARY commands, such as DIR, COPY, DEBUG, FREE, PURGE, or even small /CMD programs to be used within a running Basic program, without variable or data loss.

By special arrangement with GRL Software,

SYSTEM 1.5. is now distributed exclusively by TRSTimes magazine.

ORDER YOUR COPY TODAY!

Send $39.95 (U.S. funds) to:

TRSTimes - SYSTEM 1.5.

5721 Topanga Canyon Blvd., Suite 4

Woodland Hills, CA. 91367