```
50458 :205,201,136,208,006,072,086
50464 :032,175,203,104,096,174,048
50470 :141,002,224,005,208,115,221
50476 :201,015,208,011,162,001,130
50482 :142,098,205,032,248,197,204
50488 :076,202,197,201,003,208,175
50494 :011,169,000,141,098,205,174
50500 :032,028,198,076,202,197,033
50506 :201,026,208,016,169,000,182
50512 :141,194,002,165,056,141,011
50518 :195,002,032,066,198,076,143
50524 :202,197,201,019,208,006,157
50530 :032,124,198,076,202,197,159
50536 :201,012,208,006,032,185,236
50542 :199,076,202,197,201,002,219
50548 :208,006,238,033,208,076,117
50554 :202,197,201,011,208,006,179
50560 :238,134,002,076,202,197,209
50566 :201,006,208,006,238,032,057
50572 :208,076,202,197,201,021,021
50578 :208,011,173,024,208,073,075
50584 :002,141,024,208,076,202,037
50590 :197,162,002,032,201,255,239
50596 :173,107,205,072,168,174,039
50602 :063,003,208,006,185,000,123
50608 :207,076,182,197,104,072,246
50614 :032,210,255,104,172,019,206
50620 :003,208,011,141,085,205,073
50626 :072,032,204,255,104,032,125
50632 :106,201,032,048,200,032,051
50638 :204,255,162,002,032,198,035
50644 :255,032,228,255,201,000,159
50650 :208,003,076,008,197,174,116
50656 :063,003,208,006,041,127,160
50662 :168,185,000,206,141,085,247
50668 :205,072,032,204,255,104,084
50674 :032,106,201,076,008,197,094
50680 :160,000,152,072,185,013,062
50686 :198,141,085,205,032,248,139
50692 :201,104,168,200,192,015,116
50698 :208,238,096,013,091,066,210
50704 :085,070,070,069,082,032,168
50710 :079,080,069,078,093,013,178
50716 :160,000,152,072,185,049,134
50722 :198,141,085,205,032,248,175
50728 :201,104,168,200,192,017,154
50734 :208,238,096,013,091,066,246
50740 :085,070,070,069,082,032,204
50746 :067,076,079,083,069,068,244
50752 :093,013,160,000,152,072,042
50758 :185,087,198,141,085,205,203
50764 :032,248,201,104,168,200,005
50770 :192,017,208,238,096,013,078
50776 :091,066,085,070,070,069,027
50782 :082,032,090,069,082,079,016
50788 :069,068,093,013,160,000,247
50794 :152,072,185,236,199,141,067
50800 :085,205,032,248,201,104,219
50806 :168,200,192,014,208,238,114
50812 :160,000,152,072,185,251,176
50818 :199,141,085,205,032,248,016
50824 :201,104,168,200,192,013,246
50830 :208,238,032,228,255,240,063
50836 :251,201,089,240,007,201,113
50842 :078,208,243,076,106,199,040
50848 :169,000,141,095,205,160,162
50854 :000,152,072,185,009,200,016
50860 :141,085,205,032,248,201,060
50866 :104,168,200,192,011,208,037
50872 :238,032,204,255,160,000,049
50878 :140,101,205,032,048,200,148
50884 :032,228,255,201,000,240,128
50890 :246,201,013,240,048,201,127
50896 :020,208,011,172,101,205,157
50902 :240,233,206,101,205,076,251
50908 :239,198,201,031,144,223,232
50914 :201,091,176,219,172,101,162
50920 :205,153,238,002,238,101,145
50926 :205,141,085,205,032,248,130
50932 :201,172,101,205,192,016,107
50938 :240,197,076,193,198,172,046
50944 :101,205,140,099,205,160,142
50950 :000,185,020,200,141,085,125
50956 :205,140,100,205,032,248,174
50962 :201,172,100,205,200,192,064
50968 :014,208,236,032,228,255,229
50974 :240,251,201,084,240,007,029
50980 :201,068,240,008,076,027,144
50986 :199,162,001,076,050,199,217
50992 :162,008,169,010,160,000,045
50998 :032,186,255,173,099,205,236
51004 :162,238,160,002,032,189,075
51010 :255,169,000,141,021,208,092
51016 :169,147,032,210,255,169,030
51022 :147,141,085,205,032,248,168
51028 :201,174,095,205,208,029,228
51034 :169,000,141,021,208,174,035
51040 :194,002,172,195,002,169,062
51046 :055,032,216,255,165,055,112
51052 :141,194,002,165,056,141,039
51058 :195,002,076,142,199,169,129
51064 :000,166,055,164,056,032,081
51070 :213,255,142,194,002,140,048
51076 :195,002,162,000,134,055,168
51082 :160,038,132,056,169,001,182
51088 :141,021,208,169,000,162,077
51094 :064,157,064,003,202,016,144
51100 :250,169,255,141,085,003,035
51106 :160,003,185,181,199,141,007
51112 :085,205,152,072,032,248,194
51118 :201,104,168,136,016,240,015
51124 :096,013,075,079,013,160,104
51130 :000,140,101,205,185,034,083
51136 :200,141,085,205,032,248,079
51142 :201,238,101,205,172,101,192
51148 :205,192,014,208,234,032,065
51154 :228,255,201,000,240,249,103
51160 :201,089,240,007,201,078,008
51166 :240,138,076,209,199,162,222
51172 :001,141,095,205,032,165,099
51178 :198,096,013,091,066,085,015
51184 :070,070,069,082,032,070,121
51190 :085,076,076,093,013,013,090
51196 :083,065,086,069,032,089,164
51202 :032,079,082,032,078,063,112
51208 :013,013,070,073,076,069,066
51214 :078,065,077,069,058,013,118
51220 :013,084,065,080,069,032,107
51226 :079,082,032,068,073,083,187
51232 :075,013,013,076,079,065,097
51238 :068,032,089,032,079,082,164
51244 :032,078,063,013,174,094,242
51250 :205,240,006,162,000,141,036
51256 :021,208,096,173,054,205,045
51262 :048,015,201,040,144,024,022
51268 :169,000,141,054,205,238,107
51274 :055,205,076,092,200,238,172
51280 :054,205,206,055,205,048,085
51286 :023,169,039,141,054,205,205
51292 :173,055,205,048,013,201,019
```

```
51298 :025,144,012,206,055,205,233
51304 :032,234,232,076,113,200,223
51310 :238,055,205,169,000,141,150
51316 :051,205,173,054,205,010,046
51322 :010,141,050,205,014,050,080
51328 :205,046,051,205,024,173,064
51334 :050,205,105,024,141,000,147
51340 :208,173,051,205,105,000,114
51346 :141,016,208,173,055,205,176
51352 :010,010,010,141,086,205,102
51358 :024,173,086,205,105,050,033
51364 :141,001,208,165,162,201,018
51370 :014,144,013,133,162,173,041
51376 :033,208,041,015,141,039,141
51382 :208,076,192,200,173,134,141
51388 :002,141,039,208,165,162,137
51394 :201,028,144,004,169,000,228
51400 :133,162,172,055,205,185,088
51406 :240,236,133,209,185,056,241
51412 :205,133,210,096,172,054,058
51418 :205,177,209,201,032,208,226
51424 :003,076,105,201,162,000,003
51430 :177,209,157,169,003,072,249
51436 :024,165,210,105,212,133,061
51442 :210,177,209,157,129,003,103
51448 :056,165,210,233,212,133,233
51454 :210,136,232,224,039,240,055
51460 :011,104,201,032,208,220,012
51466 :142,053,205,076,020,201,195
51472 :104,076,105,201,174,053,217
51478 :205,200,169,032,145,209,214
51484 :200,202,208,248,173,055,090
51490 :205,201,024,208,006,032,198
51496 :234,232,206,055,205,238,186

51502 :055,205,169,000,141,054,158
51508 :205,174,053,205,168,202,035
51514 :202,189,169,003,141,083,077
51520 :205,024,165,210,105,212,217
51526 :133,210,189,129,003,141,107
51532 :134,002,072,152,072,138,134
51538 :072,032,093,202,104,170,243
51544 :104,168,104,056,165,210,127
51550 :233,212,133,210,200,202,004
51556 :016,213,206,054,205,096,122
51562 :174,098,205,208,003,076,102
51568 :248,201,160,000,141,085,179
51574 :205,166,251,142,096,205,159
51580 :166,252,142,097,205,174,136
51586 :194,002,134,251,174,195,056
51592 :002,134,252,145,251,072,224
51598 :174,096,205,134,251,174,152
51604 :097,205,134,252,056,165,033
51610 :249,237,194,002,141,088,041
51616 :205,165,250,237,195,002,190
51622 :013,088,205,208,021,162,095
51628 :002,032,201,255,169,019,082
51634 :032,210,255,032,204,255,142
51640 :032,104,198,032,066,198,046
51646 :104,096,104,201,020,240,187
51652 :011,238,194,002,208,046,127
51658 :238,195,002,076,248,201,138
51664 :141,085,205,173,194,002,240
51670 :229,055,141,088,205,173,081
51676 :195,002,229,056,013,088,035
51682 :205,240,019,072,056,173,223
51688 :194,002,233,001,141,194,229

51694 :002,173,195,002,233,000,075
51700 :141,195,002,104,173,085,176
51706 :205,174,094,205,240,072,216

51712 :162,000,142,021,208,201,222
51718 :013,240,008,162,001,142,060
51724 :244,173,076,043,202,162,144
51730 :001,142,244,173,072,120,002
51736 :162,054,134,001,032,210,105
51742 :164,032,068,168,032,125,107
51748 :164,162,055,134,001,088,128
51754 :104,174,102,205,134,251,244
51760 :174,103,205,134,252,032,180
51766 :210,002,166,251,142,102,159
51772 :205,166,252,142,103,205,109
51778 :162,000,142,244,173,096,115
51784 :173,085,205,201,032,144,144
51790 :091,201,127,144,007,201,081
51796 :160,176,003,076,170,202,103
51802 :032,065,203,172,055,205,054
51808 :185,240,236,133,209,185,004
51814 :056,205,133,210,172,054,164
51820 :205,173,084,205,240,010,001
51826 :173,083,205,009,128,145,089
51832 :209,076,129,202,173,083,224
51838 :205,145,209,024,165,210,060
51844 :105,212,133,210,173,134,075
51850 :002,145,209,056,165,210,157
51856 :233,212,133,210,173,054,135
51862 :205,201,039,208,008,174,217
51868 :191,002,208,003,032,216,040
51874 :200,238,054,205,032,048,171
51880 :200,096,173,085,205,201,104
51886 :032,176,074,201,010,208,107
51892 :003,238,055,205,201,013,127
51898 :208,005,072,032,155,203,093
51904 :104,201,014,208,010,072,033
51910 :169,002,013,024,208,141,243
51916 :024,208,104,201,017,208,198
51922 :003,238,055,205,201,018,162
51928 :208,005,162,001,142,084,050
51934 :205,201,019,208,008,162,001
51940 :000,142,054,205,142,055,058
51946 :205,201,020,208,005,072,177
51952 :032,085,203,104,201,029,126
51958 :208,003,238,054,205,201,131
51964 :141,208,005,072,032,155,097
51970 :203,104,201,142,208,010,102
51976 :072,173,024,208,041,253,011
51982 :141,024,208,104,201,145,069
51988 :208,003,206,055,205,201,130
51994 :146,208,005,162,000,142,177
52000 :084,205,201,147,208,016,125
52006 :072,032,068,229,169,000,096
52012 :141,054,205,141,055,205,077
52018 :032,048,200,104,201,157,024
52024 :208,003,206,054,205,032,252
52030 :048,200,096,173,085,205,101
52036 :072,041,128,074,141,087,099
52042 :205,104,041,063,013,087,075
52048 :205,141,083,205,096,172,214
52054 :055,205,185,240,236,133,116
52060 :209,185,056,205,133,210,066
52066 :172,054,205,240,051,056,108
52072 :169,039,237,054,205,170,210
52078 :177,209,136,145,209,200,162
52084 :024,165,210,105,212,133,197
52090 :210,177,209,136,145,209,184
52096 :056,165,210,233,212,133,113
52102 :210,200,200,202,224,255,145
52108 :208,224,169,032,160,039,204
52114 :145,209,206,054,205,032,229
52120 :048,200,096,174,168,002,072
52126 :240,003,238,055,205,169,044
```

```
52132 :000,141,054,205,141,084,021
52138 :205,032,048,200,096,173,156
52144 :094,205,240,003,076,074,100
52150 :204,174,054,205,140,090,025
52156 :205,172,055,205,140,089,030
52162 :205,162,002,032,201,255,027
52168 :169,019,032,210,255,032,149
52174 :204,255,169,216,141,229,140
52180 :203,169,000,141,228,203,132
52186 :168,141,231,203,169,176,026
52192 :141,232,203,185,255,255,215
52198 :153,255,255,056,173,228,070
52204 :203,233,232,141,088,205,058
52210 :173,229,203,233,219,013,032
52216 :088,205,240,017,238,228,240
52222 :203,238,231,203,208,223,024
52228 :238,229,203,238,232,203,067
52234 :076,227,203,169,000,141,058
52240 :021,208,169,192,141,136,115
52246 :002,173,000,221,041,252,199
52252 :141,000,221,173,024,208,027
52258 :041,015,141,024,208,173,124
52264 :033,208,141,091,205,173,123
52270 :032,208,141,092,205,173,129
52276 :134,002,141,093,205,160,019
52282 :024,185,217,000,153,000,125
52288 :180,185,000,181,153,217,212
52294 :000,136,016,241,096,173,220
52300 :094,205,208,250,169,176,154
52306 :141,119,204,169,000,141,088
52312 :118,204,168,162,002,032,006
52318 :201,255,169,017,032,210,210
52324 :255,032,204,255,141,121,084
52330 :204,169,216,141,122,204,138
52336 :120,169,054,133,001,185,006
52342 :255,255,153,255,255,056,067
52348 :173,121,204,233,233,141,205
52354 :088,205,173,122,204,233,131
52360 :219,013,088,205,240,034,167
52366 :238,118,204,238,121,204,241
52372 :208,223,238,119,204,238,098
52378 :122,204,076,117,204,160,013
52384 :024,185,217,000,153,000,227
52390 :181,185,000,180,153,217,058
52396 :000,136,016,241,169,055,021
52402 :133,001,088,173,091,205,101
52408 :141,033,208,173,093,205,013
52414 :141,134,002,173,092,205,169
52420 :141,032,208,169,004,141,123
52426 :136,002,169,001,141,021,160
52432 :208,169,003,013,002,221,056
52438 :141,002,221,169,003,013,251
52444 :000,221,141,000,221,169,204
52450 :023,141,024,208,096,173,123
52456 :167,002,174,251,002,160,220
52462 :001,032,186,255,173,169,030
52468 :002,162,172,160,002,032,006
52474 :189,255,169,000,170,160,169
52480 :206,032,213,255,032,147,117
52486 :199,096,173,167,002,174,049
52492 :251,002,160,001,032,186,132
52498 :255,173,169,002,162,172,183
52504 :160,002,032,189,255,169,063
52510 :206,133,254,169,000,133,157
52516 :253,169,253,162,255,160,008
52522 :207,032,216,255,032,147,163
52528 :199,096,000,000,000,000,087
52534 :000,000,004,004,004,004,070
52540 :004,004,004,005,005,005,087
52546 :005,005,005,006,006,006,099
52552 :006,006,006,006,007,007,110
52558 :007,007,007,013,013,013,138
```

Program 3: Plus/Term ML Portion For VIC

Refer to the "MLX" article before entering this listing.

```
6144 :032,023,032,076,200,024,131
6150 :169,016,141,136,002,169,127
6156 :194,141,005,144,169,140,037
6162 :141,252,002,169,032,141,243
6168 :253,002,169,171,141,254,246
6174 :002,169,032,141,255,002,119
6180 :169,001,141,102,033,032,002
6186 :120,028,169,147,141,093,228
6192 :033,032,252,029,169,000,051
6198 :141,091,033,133,055,169,164
6204 :064,133,056,169,035,133,138
6210 :044,169,000,141,191,002,101
6216 :141,107,033,141,000,035,017
6222 :169,000,141,063,003,032,230
6228 :117,024,169,000,141,092,115
6234 :033,141,093,033,141,094,113
6240 :033,169,000,141,019,003,205
6246 :165,055,141,194,002,165,056
6252 :056,141,195,002,076,200,010
6258 :024,006,016,160,255,152,215
6264 :192,219,176,024,192,193,092
6270 :144,006,056,233,128,076,001
6276 :150,024,192,065,144,012,207
6282 :192,096,176,006,024,105,225
6288 :032,076,150,024,169,000,083
6294 :153,000,019,136,192,255,137
6300 :208,217,160,255,152,192,060
6306 :128,176,024,192,096,144,154
6312 :006,056,233,032,076,191,250
6318 :024,192,065,144,012,192,035
6324 :091,176,008,024,105,128,200
6330 :076,191,024,169,000,153,031
6336 :000,018,136,192,255,208,233
6342 :217,096,032,052,028,032,143
6348 :204,255,174,141,002,224,180
6354 :004,240,017,224,005,240,172
6360 :055,032,228,255,201,000,219
6366 :240,003,076,197,025,076,071
6372 :243,025,164,197,192,128,153
6378 :176,226,185,094,236,056,183
6384 :233,064,048,218,240,216,235
6390 :160,000,162,000,136,208,144
6396 :253,202,208,250,076,197,158
6402 :025,160,000,162,000,136,229
6408 :208,253,202,208,250,076,181
6414 :243,025,166,197,224,041,142
6420 :208,008,072,032,160,026,014
6426 :104,076,194,025,166,197,020
6432 :224,021,208,008,072,032,085
6438 :185,027,104,076,194,025,137
6444 :166,197,224,035,208,019,125
6450 :072,024,173,015,144,105,071
6456 :016,141,015,144,009,008,133
6462 :141,015,144,104,076,194,224
6468 :025,166,197,224,044,208,164
6474 :018,072,173,134,002,041,002
6480 :007,024,105,001,041,007,009
6486 :141,134,002,104,076,194,225
6492 :025,166,197,224,042,208,186
6498 :030,072,024,173,109,033,027
6504 :041,007,024,105,001,141,167
6510 :109,033,173,015,144,041,113
6516 :248,013,109,033,009,008,024
6522 :141,015,144,104,076,194,028
```

```
6528 :025,166,197,224,052,208,232
6534 :013,072,162,001,142,107,119
6540 :033,032,028,026,104,076,183
6546 :194,025,166,197,224,034,218
6552 :208,013,072,162,000,142,237
6558 :107,033,032,064,026,104,012
6564 :076,194,025,166,197,224,022
6570 :033,208,018,072,162,000,151
6576 :142,194,002,166,056,142,110
6582 :195,002,032,102,026,104,131
6588 :076,194,025,076,206,024,021
6594 :076,003,025,072,201,136,195
6600 :208,005,104,032,152,031,220
6606 :096,162,002,032,201,255,186
6612 :174,063,003,208,009,104,005
6618 :072,168,185,000,019,076,226
6624 :228,025,104,072,032,210,127
6630 :255,104,172,019,003,208,223
6636 :006,141,093,033,032,109,138
6642 :029,032,052,028,032,204,107
6648 :255,162,002,032,198,255,128
6654 :032,228,255,201,000,208,154
6660 :003,076,200,024,174,063,032
6666 :003,208,006,041,127,168,051
6672 :185,000,018,141,093,033,230
6678 :032,109,029,076,200,024,236
6684 :160,000,152,072,185,049,134
6690 :026,141,093,033,032,252,099
6696 :029,104,168,200,192,015,236
6702 :208,238,096,013,091,066,246
6708 :085,070,070,069,082,032,204
6714 :079,080,069,078,093,013,214
6720 :160,000,152,072,185,085,206
6726 :026,141,093,033,032,252,135
6732 :029,104,168,200,192,017,018
6738 :208,238,096,013,091,066,026
6744 :085,070,070,069,082,032,240
6750 :067,076,079,083,069,068,024
6756 :093,013,160,000,152,072,078
6762 :185,123,026,141,093,033,195
6768 :032,252,029,104,168,200,129
6774 :192,017,208,238,096,013,114
6780 :091,066,085,070,070,069,063
6786 :082,032,090,069,082,079,052
6792 :069,068,093,013,160,000,027
6798 :152,072,185,240,027,141,191
6804 :093,033,032,252,029,104,179
6810 :168,200,192,014,208,238,150
6816 :160,000,152,072,185,255,216
6822 :027,141,093,033,032,252,232
6828 :029,104,168,200,192,013,110
6834 :208,238,032,228,255,240,099
6840 :251,201,089,240,007,201,149
6846 :078,208,243,076,132,027,186
6852 :169,000,141,106,033,160,037
6858 :000,152,072,185,027,028,154
6864 :141,093,033,032,252,029,020
6870 :104,168,200,192,011,208,073
6876 :238,032,204,255,160,000,085
6882 :140,104,033,032,228,255,250
6888 :201,000,240,249,201,013,112
6894 :240,048,201,020,208,011,198
6900 :172,104,033,240,236,206,211
6906 :104,033,076,016,027,201,195
6912 :031,144,226,201,091,176,101
6918 :222,172,104,033,153,050,228
6924 :033,238,104,033,141,093,142
6930 :033,032,252,029,172,104,128
6936 :033,192,016,240,200,076,013
6942 :229,026,172,104,033,140,222
6948 :105,033,160,000,185,013,020
6954 :028,141,093,033,140,103,068
6960 :033,032,252,029,172,103,157
6966 :033,200,192,014,208,236,169
6972 :032,204,255,032,228,255,042
6978 :240,248,201,084,240,007,062
6984 :201,068,240,008,076,060,213
6990 :027,162,001,076,086,027,201
6996 :162,008,169,010,160,000,081
7002 :032,186,255,173,105,033,106
7008 :162,050,160,033,032,189,210
7014 :255,169,147,141,093,033,172
7020 :032,252,029,169,147,032,001
7026 :210,255,174,106,033,208,076
7032 :022,174,194,002,172,195,111
7038 :002,169,055,032,216,255,087
7044 :165,055,141,194,002,165,086
7050 :056,141,195,002,096,169,029
7056 :000,166,055,164,056,032,105
7062 :213,255,142,194,002,140,072
7068 :195,002,162,000,134,055,192
7074 :160,063,132,056,160,003,224
7080 :185,236,027,141,093,033,115
7086 :152,072,032,252,029,104,047
7092 :168,136,016,240,096,160,228
7098 :000,140,104,033,185,038,174
7104 :028,141,093,033,032,252,003
7110 :029,238,104,033,172,104,110
7116 :033,192,014,208,234,032,149
7122 :228,255,201,000,240,249,103
7128 :201,089,240,007,201,078,008
7134 :240,011,076,209,027,162,179
7140 :001,141,106,033,032,201,230
7146 :026,096,013,075,079,013,024
7152 :013,091,066,085,070,070,123
7158 :069,082,032,070,085,076,148
7164 :076,093,013,013,083,065,083
7170 :086,069,032,089,032,079,133
7176 :082,032,078,063,013,013,033
7182 :084,065,080,069,032,079,167
7188 :082,032,068,073,083,075,177
7194 :013,013,070,073,076,069,084
7200 :078,065,077,069,058,013,136
7206 :013,076,079,065,068,032,115
7212 :089,032,079,082,032,078,180
7218 :063,013,173,214,032,048,081
7224 :015,201,022,144,024,169,119
7230 :000,141,214,032,238,215,134
7236 :032,076,085,028,238,214,229
7242 :032,206,215,032,048,023,118
7248 :169,021,141,214,032,173,062
7254 :215,032,048,013,201,023,106
7260 :144,012,206,215,032,032,221
7266 :117,233,076,106,028,238,128
7272 :215,032,172,215,032,185,187
7278 :240,032,133,209,185,216,101
7284 :032,133,210,096,172,214,205
7290 :032,177,209,073,128,145,118
7296 :209,024,165,210,105,132,205
7302 :133,210,165,209,105,000,188
7308 :133,209,173,102,033,240,006
7314 :022,177,209,041,015,141,239
7320 :101,033,169,000,141,102,186
7326 :033,173,134,002,041,015,044
7332 :145,209,076,179,028,173,206
7338 :101,033,145,209,169,001,060
7344 :141,102,033,056,165,210,115
7350 :233,132,133,210,233,000,099
7356 :133,209,032,052,028,096,226
7362 :172,214,032,177,209,201,175
```

```
7368 :032,208,012,238,214,032,168
7374 :032,052,028,032,120,028,242
7380 :104,104,096,162,000,177,087
7386 :209,157,010,033,072,024,211
7392 :165,210,105,132,133,210,155
7398 :177,209,157,050,033,056,144
7404 :165,210,233,132,133,210,039
7410 :136,232,224,022,240,011,083
7416 :104,201,032,208,220,142,131
7422 :213,032,076,016,029,104,212
7428 :238,214,032,032,052,028,088
7434 :032,120,028,104,104,096,238
7440 :174,213,032,200,169,032,068
7446 :145,209,200,202,208,248,210
7452 :173,215,032,201,023,208,112
7458 :006,032,117,233,206,215,075
7464 :032,238,215,032,169,000,214
7470 :141,214,032,032,052,028,033
7476 :174,213,032,160,000,202,065
7482 :202,189,010,033,141,090,211
7488 :033,024,165,210,105,132,221
7494 :133,210,189,050,033,141,058
7500 :134,002,056,165,210,233,108
7506 :132,133,210,072,152,072.085
7512 :138,072,032,017,030,104,225
7518 :170,104,168,104,200,202,018
7524 :016,213,206,214,032,032,045
7530 :052,028,096,174,107,033,084
7536 :208,003,076,252,029,160,072
7542 :000,141,093,033,166,253,036
7548 :142,064,003,166,254,142,127
7554 :065,003,174,194,002,134,190
7560 :253,174,195,002,134,254,124
7566 :145,253,072,174,064,003,085
7572 :134,253,174,065,003,134,143
7578 :254,056,165,249,237,194,029
7584 :002,141,096,033,165,250,079
7590 :237,195,002,013,096,033,230
7596 :208,021,162,002,032,201,030
7602 :255,169,019,032,210,255,094
7608 :032,204,255,032,140,026,105
7614 :032,102,026,104,096,104,142
7620 :201,020,240,011,238,194,076
7626 :002,208,047,238,195,002,126
7632 :076,252,029,141,093,033,064
7638 :056,173,194,002,229,055,155
7644 :141,096,033,173,195,002,092
7650 :229,056,013,096,033,240,125
7656 :019,072,056,173,194,002,236
7662 :233,001,141,194,002,173,214
7668 :195,002,233,000,141,195,242
7674 :002,104,173,093,033,201,088
7680 :032,144,115,201,127,144,251
7686 :007,201,160,176,003,076,117
7692 :118,030,032,041,031,032,040
7698 :120,028,172,215,032,185,002
7704 :240,032,133,209,185,216,015
7710 :032,133,210,172,214,032,055
7716 :173,091,033,240,010,173,244
7722 :090,033,009,128,145,209,144
7728 :076,056,030,173,090,033,250
7734 :145,209,024,165,210,105,144
7740 :132,133,210,173,134,002,076
7746 :145,209,056,165,210,233,060
7752 :132,133,210,173,214,032,198
7758 :201,021,208,026,174,191,131
7764 :002,208,021,032,194,028,057
7770 :238,214,032,032,052,028,174
7776 :032,120,028,169,001,141,075
7782 :102,033,032,120,028,096,001
7788 :238,214,032,032,052,028,192
7794 :032,120,028,096,032,120,030
7800 :028,173,093,033,201,032,168
7806 :176,084,201,013,208,005,045
7812 :072,032,132,031,104,201,192
7818 :014,208,010,072,169,002,101
7824 :013,005,144,141,005,144,084
7830 :104,201,017,208,008,072,248
7836 :238,215,032,032,052,028,241
7842 :104,201,018,208,007,072,004
7848 :169,001,141,091,033,104,195
7854 :201,019,208,013,072,169,088
7860 :000,141,214,032,141,215,155
7866 :032,032,052,028,104,201,123
7872 :020,208,005,072,032,062,079
7878 :031,104,201,029,208,008,011
7884 :072,238,214,032,032,052,076
7890 :028,104,201,141,208,005,129
7896 :072,032,132,031,104,201,020
7902 :142,208,010,072,169,240,039
7908 :045,005,144,141,005,144,200
7914 :104,201,145,208,008,072,204
7920 :206,215,032,032,052,028,037
7926 :104,201,146,208,007,072,216
7932 :169,000,141,091,033,104,022
7938 :201,147,208,016,072,032,166
7944 :095,229,169,000,141,214,088
7950 :032,141,215,032,032,052,006
7956 :028,104,201,157,208,008,214
7962 :072,206,214,032,032,052,122
7968 :028,104,032,052,028,032,052
7974 :120,028,096,173,093,033,069
7980 :041,128,074,141,095,033,044
7986 :173,093,033,041,063,013,210
7992 :095,033,141,090,033,096,032
7998 :172,215,032,185,240,032,170
8004 :133,209,185,216,032,133,208
8010 :210,172,214,032,240,051,225
8016 :056,169,021,237,214,032,041
8022 :170,177,209,136,145,209,108
8028 :200,024,165,210,105,132,160
8034 :133,210,177,209,136,145,084
8040 :209,056,165,210,233,132,085
8046 :133,210,200,200,202,224,255
8052 :255,208,224,169,032,160,140
8058 :021,145,209,206,214,032,181
8064 :032,052,028,096,174,168,166
8070 :002,240,003,238,215,032,096
8076 :169,000,141,214,032,141,069
8082 :091,033,032,052,028,096,222
8088 :174,214,032,142,098,033,077
8094 :172,215,032,140,097,033,079
8100 :162,002,032,201,255,169,217
8106 :019,032,210,255,032,204,154
8112 :255,169,148,141,199,031,095
8118 :169,000,141,198,031,141,094
8124 :201,031,169,022,141,202,186
8130 :031,160,000,185,255,255,056
8136 :153,255,255,056,173,201,013
8142 :031,233,250,141,096,033,222
8148 :173,202,031,233,023,013,119
8154 :096,033,240,017,238,198,016
8160 :031,238,201,031,208,223,132
8166 :238,199,031,238,202,031,145
8172 :076,197,031,169,020,141,102
8178 :136,002,173,015,144,141,085
8184 :099,033,169,210,141,005,137
8190 :144,173,134,002,141,100,180
8196 :033,160,025,185,217,000,112
8202 :153,000,034,185,050,034,210
```

```
8208 :153,217,000,136,016,241,011
8214 :096,169,022,141,060,032,030
8220 :169,000,141,059,032,162,079
8226 :002,032,201,255,169,017,198
8232 :032,210,255,032,204,255,004
8238 :169,000,141,062,032,169,107
8244 :148,141,063,032,160,000,084
8250 :185,255,255,153,255,255,136
8256 :056,173,059,032,233,250,099
8262 :141,096,033,173,060,032,093
8268 :233,023,013,096,033,240,202
8274 :017,238,059,032,238,062,216
8280 :032,208,223,238,060,032,113
8286 :238,063,032,076,058,032,081
8292 :173,099,033,141,015,144,193
8298 :173,100,033,141,134,002,177
8304 :169,016,141,136,002,169,233
8310 :194,141,005,144,160,025,019
8316 :185,217,000,153,050,034,251
8322 :185,000,034,153,217,000,207
8328 :136,016,241,096,173,167,197
8334 :002,174,251,002,160,001,220
8340 :032,186,255,173,169,002,197
8346 :162,172,160,002,032,189,103
8352 :255,169,000,162,000,160,138
8358 :024,032,213,255,096,173,191
8364 :167,002,174,251,002,160,160
8370 :001,032,186,255,173,169,226
8376 :002,162,172,160,002,032,202
8382 :189,255,169,024,133,254,190
8388 :169,000,133,253,169,253,149
8394 :162,255,160,025,032,216,028
8400 :255,096,000,000,000,000,047
8406 :000,000,016,016,016,016,022
8412 :016,016,016,016,016,016,060
8418 :016,016,017,017,017,017,070
8424 :017,017,017,017,017,017,078
8430 :017,017,000,022,044,066,148
8436 :088,110,132,154,176,198,078
8442 :220,242,008,030,052,074,108
8448 :096,118,140,162,184,206,138
8454 :228,250,013,013,013,013,024
```

Program 4: VIC/64 Tokenizer (Disk Only)

Refer to the "MLX" article before entering this listing.

```
828 :032,237,255,224,022,240,046
834 :012,032,253,174,032,158,215
840 :173,032,130,183,076,095,249
846 :003,032,253,206,032,158,250
852 :205,032,130,215,166,034,098
858 :164,035,032,189,255,166,163
864 :034,164,035,032,189,255,037
870 :169,032,162,008,160,008,129
876 :032,186,255,032,192,255,036
882 :169,125,141,036,003,169,245
888 :003,141,037,003,096,008,152
894 :138,072,152,072,169,008,225
900 :032,180,255,169,104,032,136
906 :150,255,032,165,255,141,112
912 :203,003,032,171,255,165,205
918 :144,240,026,169,032,032,025
924 :195,255,032,138,255,169,176
930 :008,032,177,255,169,232,011
936 :032,147,255,032,174,255,039
942 :169,013,141,203,003,173,108
948 :204,003,208,010,173,203,213
954 :003,201,013,240,003,032,166
960 :210,255,104,168,104,170,179
966 :040,173,203,003,096,000,201
```

THE BEGINNER'S PAGE

Tom R. Halfhill, Editor

IF-THEN Intelligence

At one time or another you've probably seen the term *artificial intelligence*. It refers, of course, to computer intelligence—the ability of a machine to reproduce (or, if you prefer, simulate) some of the thought processes of a human being.

We're not going to reopen here the philosophical debate about whether computers are really intelligent, or if they ever will be intelligent (or for that matter, if humans are intelligent). Scientists still can't agree on exactly how the human brain works, much less whether it can be duplicated in silicon circuitry.

But we do know how computers work. Although computers aren't (yet) capable of independent thought or action, they certainly appear intelligent at times. They can play chess at the grandmaster level, forecast tomorrow's weather as well as anybody else, help plan the economy of a household or a nation, create wonderfully abstract art, and even simulate the responses of a psychoanalyst closely enough to fool many laypeople. How can a mass of wires and silicon chips seem to be so smart? What sets computers apart from all other machines?

Programmability alone isn't the answer. There were programmable machines long before computers came along. One example is the centuries-old music box. A melody is programmed into the box by punching little bumps onto the surface of a revolving drum; as the drum turns, the bumps pluck a series of tiny metal prongs tuned to different notes. Of course, a music box is capable of playing only one melody drum, or "program." A more sophisticated example is the player piano, with its interchangeable paper rolls that operate on the same principle.

Still, there's something missing from a programmable player piano that keeps it from qualifying as a true computer. Even some of today's programmable calculators lack the essential element of computer intelligence. They, too, can be programmed to carry out a series of steps, but they can't imitate the decision-making power of a real brain.

What do computers have that all these other machines don't? *Conditional logic.* Although some other devices are capable of conditional logic on a very primitive level, no machine can do it as flexibly as a computer.

Michael Jackson Vs. Beethoven

Here's how conditional logic works. Let's say you send a friend to a record store with a $10 bill and these instructions: "If the store has Michael Jackson's latest album, then buy it for me. Otherwise, buy the Cleveland Symphony Orchestra's new recording of Beethoven's Fifth."

Now, you've done more than simply programmed your friend to visit the store and buy you a record. You've given him the power to make a decision in your absence, and also the information he needs to make the decision. Depending on whether a certain *condition* is met (if the store has Michael Jackson's latest album or not), your friend will act on either of the two alternatives (he'll buy you the Jackson record or the Beethoven record). Even if your friend has the brains of a hamster, he'll appear semi-intelligent to the record store clerk as he flips through the bins and picks the correct album.

All computer programming languages have commands that let you tell the computer to make the same sort of decisions. In BASIC, the most common command for conditional logic is the IF-THEN statement. It takes this form:

IF *condition is met* **THEN** *perform this action.*

Computers don't understand English, of course, so the italicized words above must be replaced with terms the computer *can* understand. Usually the conditional part of the statement involves a comparison between variables and numbers. And often the resulting action will be a second command which sends the computer to another section of the program. Let's try some actual examples.

Conversational Computing

At one time or another you've probably used a computer program which carries on a conversation with you, depending on how you respond to certain questions. The machine appears almost human, and conditional logic is the key.

Let's say you're an insurance salesman who is writing a program designed to analyze a client's

life insurance needs. One of the first questions the program needs to ask is the person's age. This can be done with a simple PRINT statement. Clear the computer's memory by turning it off, then on again, and type the following line exactly as it appears (remember that to enter a program line into memory, you must press the RETURN or ENTER key after typing the line):

```
10 PRINT "What is your age";
```

When the program runs, the PRINT statement will print the text between the quotes on the screen. Next, type this line:

```
20 INPUT A
```

When the program runs, this simple statement does three things. First, it prints a question mark after the text in the PRINT statement. Second, it makes the computer pause until someone types in a number on the keyboard. Finally, it takes the number and stores it in a memory location which can be referenced with the variable name A.

Now it's time for some conditional logic. Enter these two lines:

```
30 IF A>100 THEN PRINT "It's a little late to be
thinking about life insurance, isn't it?":END
40 PRINT "We have just the policy that you need."
```

Now clear the screen and run the program. When it asks for your age, try typing in a number less than 100. Then run the program again and type in a number greater than 100. See the difference? (Note: This program requires Extended BASIC on the TI-99/4A.)

Here's how it works. Line 30 compares the value stored in the variable A with 100. If A is greater than 100 (> is the *greater-than* sign), then the condition is met and the computer performs the instruction which immediately follows. *The second half of an IF-THEN statement is executed only when the condition in the first half is true.* (Incidentally, line 30 is also a *multistatement line;* a colon separates the second statement, END, from the IF-THEN statement. This command ends the program after the remark is printed.)

If the user claims to be less than 100 years old, the condition in line 30 is not met. Therefore, the computer ignores the rest of line 30 and continues (or "falls through") to the next line. The computer prints a different message on the screen and, presumably, would go on to the remainder of the program.

Simulated Intelligence

Although this simple four-line program contains only one conditional statement, it illustrates how computers can appear intelligent. The computer not only makes different responses depending on the user's input, it also seems to know that centenarians are unlikely candidates for life insurance. It even reveals a snappy sense of humor. Of course, it's really the programmer talking, not the computer. (Remember this the next time your computer makes a rude remark.)

Simple conditional statements like the one above are the basis for nearly all of what passes as artificial intelligence today. All programs work on the same principle, from the sophisticated modeling software that forecasts the nation's economy to the chess program which threatens the supremacy of the world's top champions. In fact, the chess program published in the December 1984 issue of COMPUTE!, when playing on level 5, uses a similar technique to evaluate up to 50 million possible moves during each turn. Naturally, a program that complicated must be written in machine language, not BASIC, or you'd be waiting months for the computer's response.

The IF-THEN statement isn't the only way to simulate intelligence in BASIC. Most BASICs have at least two other statements which accomplish more or less the same thing. This indicates how important conditional logic really is in programming. (Linguists say you can determine a language's most often used concepts by counting the synonyms—interestingly, somebody once figured out that English has more terms for being inebriated than almost any other concept.)

In BASIC, as we mentioned, IF-THEN is by far the most common conditional statement. Some of the more powerful BASICs—such as IBM BASIC—augment the IF-THEN statement with an ELSE condition. This lets you combine two lines into one. For instance, lines 30 and 40 above could be rewritten like this in IBM BASIC:

```
30 IF A<100 THEN PRINT "We have just the policy
for your needs." ELSE PRINT "It's a little late to be
thinking about life insurance, isn't it?":END
```

ELSE doesn't let you do anything you couldn't do otherwise; it just makes programming more convenient.

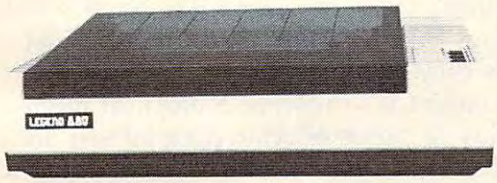
Other examples of conditional statements in BASIC are ON-GOTO and ON-GOSUB. In effect, these let you combine a whole series of IF-THENs into one compact line. They are also called *conditional branching* statements because they branch to other parts of the program. An IF-THEN statement can be used for conditional branching, too. We'll cover both conditional and unconditional branching in next month's column, and also show a couple of ways to avoid long, cumbersome lists of IF-THENs in your programs.

Questions Beginners Ask

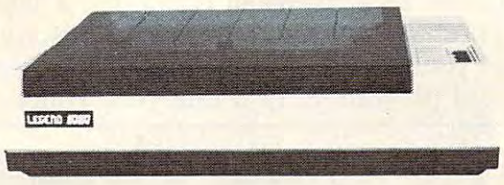
Q What exactly is a *crash* or a *freeze* and how can I avoid it? All I know so far is

"...Darn near letter quality!"

The high quality, square dot technology used by Legend produces a character so clear, so crisp our users tell us it's "darn near letter quality!" We invite comparisons. In fact, we're so confident about our quality we'll be happy to send you an actual sample of legendary output, just for the asking. Legends are perfect for those important reports and proposals as well as regular office correspondence. The graphs and charts you create with Legend are stunningly good! All you have to choose is how fast you want to go. We use a top-quality carbon ribbon common to the world's most popular typewriter that makes each and every character clean and sharp. And we're so sure about the reliability of our Legends we guarantee our print head... for life!



The LEGEND 880 provides over forty fonts, all software-selectable and is rated at 80 cps but purrs along at a comfortable RTS of 104 characters per second. It's designed to work with all popular computers including IBM, most of the IBM-compatibles as well as Apple, TI and Commodore. And all this can be yours today at a really affordable price!



The LEGEND 1080, rated at 100 cps gives you the quality of the 880 at a faster RTS of 140 characters per second. And simple, easy-to-use switch settings bring forty fonts to your fingertips! True Epson compatibility means you can run all the popular software packages including Lotus 1-2-3, Symphony, Framework, Wordstar and more!

The LEGEND 1380 is perfect for high speed, high performance applications. Rated at 130 cps, it produces legendary print quality at an incredible RTS of 163 characters per second. Full IBM graphics compatibility along with downloadable character sets allows you to design your very own fonts and run all of the new IBM graphics software.

Upgrade your printer buffer for only $1.00. For a limited time only you can upgrade the buffer in either your Legend 1080 or Legend 1380. See your dealer for all the details.

For more information about these and the full line of Legendary printers contact Legend Peripheral Products, 6041 Variel Avenue, Woodland Hills, Ca 91367. Telephone (818) 704-9100. Outside CA call toll-free 1-800-321-4484. Telex 662436.

LEGEND PERIPHERAL PRODUCTS

that it can happen when one mistypes numbers in DATA statements, but not why, nor what else can cause it, so I can't yet reason backward from crash to cause. I've been trying to translate a budget and cash-flow analysis program written for the IBM PC so it will run on a Timex Sinclair TS-1000 with a 16K RAM pack, which in theory should take a program up to 900 lines long. But I can't even begin to type it in. The first 30 lines contain a DIM block of about ten lines and other statements. Before I can finish, I get a crash: blank screen, no cursor, keyboard dead, BREAK and STOP keys disabled. Nothing to do but unplug, replug, and start from scratch. Nothing here seems similar to mistyping figures in DATA statements, and I don't see how I could be running out of memory so soon (the manual says you will eventually—not bang all of a sudden—reach a blank screen when you've jammed the poor beast beyond capacity).

Norman Hartweg

A You've described the symptoms of a *system crash* or *freeze* perfectly—the screen often goes haywire, the computer won't respond to commands typed on the keyboard, and your only alternative is to power down and wipe out whatever you were working on.

Disregarding rare hardware failures, system crashes generally happen when the computer gets stuck in what's called an *infinite loop* at the machine level. This means the computer tries to execute a series of instructions which loop back on themselves or cancel each other out. The computer might look paralyzed, but it's really very busy trying to accomplish the impossible. So busy, in fact, that it ignores everything else, including your demands for attention on the keyboard.

As an example, suppose you told somebody to resolve these two statements: "Assume everything I say is always the truth. Now, I'm telling a lie." If everything you say is the truth, then you're telling the truth when you say you're lying. But if you're really lying, then everything you say isn't always the truth. But it's a given that you never tell a lie . . . and so on. You can go back and forth like this forever—except you're smart enough not to try, and a computer isn't. (*Star Trek* fans may recall how Captain Kirk used this classic paradox to provoke an android into a system crash and free his crew.)

System crashes can happen when DATA statements are mistyped, because DATA numbers often contain machine language subprograms. A mistyped number can create a wrong instruction, which in turn can trap the computer in an endless loop. Typing the wrong number in a POKE statement can do the same thing.

However, these crashes happen only when the program runs, not when it's being typed. As you surmised, your symptoms indicate a memory problem. Check these possibilities:

1. The 16K RAM pack on a TS-1000 fits rather loosely. If it's not plugged in all the way, the computer may not be recognizing the extra memory, leaving you with only the 2K internal RAM, not the 16K you think you have. Also, if the pack wiggles when you type on the keyboard, the faulty connection can crash the computer.

2. The DIM statements you mentioned at the beginning of the program may be eating up all your RAM. DIM statements aren't like other statements; they take up more memory than just the characters they're composed of. The purpose of a DIM statement is to reserve a block of memory for an array (a series of related variables). In Sinclair BASIC, each element of a numeric array consumes five bytes each. The statement **DIM X(375)** might look pretty short and harmless, but it actually consumes so much memory that it causes an error when typed into a TS-1000 with 2K of RAM. If the DIM statements are the source of your trouble, you'll have to scale down the IBM program to fit it into your Timex. ©

Adding Sound Effects To Atari

Matt Giwer

Do you want more realistic sounds than beeps? This article gives you five short programs that let your Atari produce some interesting, subtle sounds.

There are many packages available for making the Atari a music box. All provide some degree of musical verisimilitude; that is, they sound good. However, these packages don't really help those who are tired of hearing the same old beeps in their programs.

Let's consider how tones are generated in the real world and try to duplicate them on the computer. And take heart: It will all be done in BASIC. You won't even find a page 6 subroutine. You can improve on what I have done once you understand what is going on.

Frequencies And Amplitude

Sound is generated by vibrating objects. Each object sounds different because it has a different set of resonant frequencies. Also, each object has a different set of loss parameters that determine how quickly the amplitude of the sound will rise and fall. Other factors such as the noise content add to the character of each individual sound.

The resonant frequency of an object is the dominant characteristic, so let's take it first. Strike a bell or play a note on a piano and you don't hear just one frequency, you hear a wide range of frequencies. The lowest frequency is called the fundamental frequency. Most objects also support harmonics of this frequency. The even harmonics are two, four, six, eight, and so forth times as high as the fundamental. The odd harmonics are three, five, seven, nine, and so forth times as high.

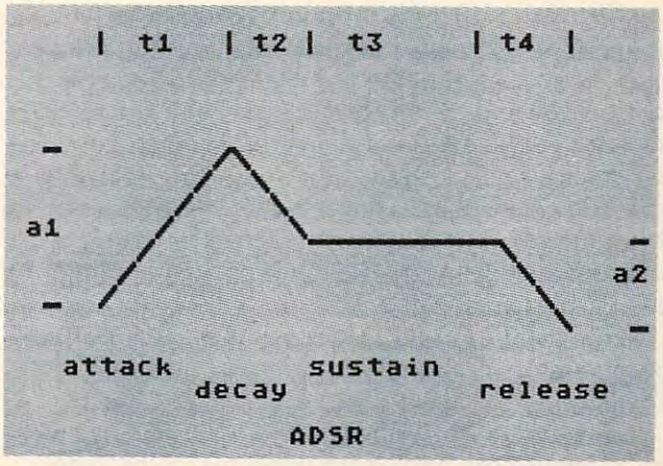
The mix of these frequencies and whether they are odd or even are components of the character of the sound. A piano is rich in even harmonics; a woodwind is rich in odd harmonics. There is one further complication: Most objects have more than one fundamental frequency, and each fundamental frequency may have its own set of harmonics. It is thus obvious why simple beeps lack the complexity of real sounds.

The next factor is how fast the amplitude of these frequencies rises and falls. When a drum is struck, it gives a short, loud sound, so its amplitude rises and falls quickly. A piano note is loud, but is sustained for some time. A woodwind rises slowly in amplitude and falls slowly.

The simplification used for electronic simulation of musical instruments is called the ADSR envelope (shown in the figure). These letters stand for Attack, Decay, Sustain, and Release. Across the top are times 1 through 4 and amplitudes 1 and 2. A struck instrument has a very short t1 and a somewhat longer t2. For a piano, t3 and t4 are relatively long and for a drum, t3 and t4 are short. Also for struck instruments the amplitude a1 will be at least twice as great as a2, but in a woodwind they may be the same.

Noise And Tone

The third factor is the amount of noise. This is easily heard in a snare drum, which contains added wires to make noise. It is less obvious in woodwinds, where the very act of blowing creates some noise. Blowing is similar to whispering, where there is little tonal content and the lips and tongue are modulating noise. A



"breathy" speech pattern is comparable to a horn or woodwind. Thus a low amplitude component of noise can add to the realism.

With these three factors in mind, let's examine a woodwind in detail. The artist is aware that the instrument has a slow attack time, so he will blow harder at the beginning of a note to give it a definite start. In doing so he will increase the noise level. The initial breath will start the instrument vibrating, and the various fundamentals and harmonics will be heard. If the instrument is being played loudly, some of the harder-to-excite frequencies will be heard too.

Once the instrument has established its note, the artist will blow more softly, so the tone content will be up, but the noise component will be down. Finally the note will end, and the artist will stop blowing. The noise component will stop immediately, but the tonal component will continue for a short time. If all this doesn't sound complex enough, imagine what goes on in a slide trombone.

Real sounds are complex, but computer-generated tones are not. So we must make them complex to make them realistic. For a perfect simulation of a piano or clarinet, a Moog synthesizer is definitely preferred. To make do with the Atari sound chip, machine language might be preferable. But if you draw a line with one end being the beeps and the other end the Moog, we'll see that BASIC *can* get us more than halfway toward the Moog.

Experimenting With Sound Qualities

The following programs are examples of some complex sonic effects. These programs directly POKE the audio control registers with amplitude information. The amplitude varies from 0 (off) to 15 (loudest). These amplitude levels are stored in S0$ in pairs of numbers. These are converted to numbers in line 21 and POKEd in line 22. The cycle through the loop provides automatic timing for note duration. Q=1 ^ 1 is used for timing pure tones.

Program 1 is a direct comparison between notes with and without an ADSR amplitude envelope. The S0$ envelope here is that of a struck instrument. Note that the first value in S0$ is 15 for shortest Attack, then a 12 to provide a slower decay, three 08's for sustain, and then 04 and 00 for a slower release. You will certainly notice the difference in tones when you run the program; in fact, the switch from envelope to no envelope is itself a pleasing effect.

Program 2 offers a somewhat different effect by playing two chords with and without the envelope.

Program 3 introduces the effect of harmonics on a single note and again gives a side-by-side comparison. These are the even harmonics found in pianos. Note several points here. In most instruments the fundamental frequency is the loudest, then the harmonics. And the fourth harmonic may be louder than the second. In real instruments there are both odd and even harmonics. A string plucked in the center produces a different sound than a string plucked near the end. I have ignored both of these points in this program.

Program 4 is similar to Program 3 except that it provides odd harmonics with and without the ADSR envelope. Program 5 compares even and odd harmonics with the envelope. In both programs I have ignored the complications—or let's say that I have created a musical sound that cannot be duplicated by a musical instrument.

Remember, the objective of this article was to show ways to add color to your sounds, not to synthesize the Boston Symphony Orchestra. So let me give you some further suggestions. In any of these programs, change line 20 to FOR I=8 TO 1 STEP − 1 and you will have a sound that can be duplicated only by playing a tape recorder backwards. How about adding odd and even harmonics of the same fundamental frequency? Perhaps instead of having the Release part of the envelope go to 00, it would go back up to 15. How about using one of the sound channels to add the slight bit of noise that goes along with some instruments?

Please refer to "COMPUTE!'s Guide To Typing In Programs" before entering these listings.

Program 1: Notes With And Without ADSR Envelope

```
PK Ø GOTO 1ØØ
KM 1Ø REM Sound envelope
AK 2Ø FOR I=1 TO 25
CN 21 TRAP 23:R=VAL(S$(2*I-1,2*I)):T
   RAP 4ØØØØ
LE 22 POKE 53761,16Ø+R:POKE 53763,16Ø
   +R:POKE 53765,16Ø+R:POKE 53767,
   16Ø+R:NEXT I
EF 23 RETURN
MI 1ØØ DIM S$(5Ø)
OB 11Ø S$="151208Ø8Ø8Ø4Ø2ØØ"
JD 19Ø POKE 53768,243
HE 2ØØ GOSUB 2Ø
JE 29Ø POKE 53768,162
HF 3ØØ GOSUB 2Ø
AE 3Ø5 Q=1^1^1^1
IN 31Ø POKE 53768,243
OJ 312 POKE 53761,168:Q=1^1:POKE 5376
   1,Ø
IP 33Ø POKE 53768,162
OL 332 POKE 53761,168:Q=1^1:POKE 5376
   1,Ø
AD 34Ø Q=1^1^1^1
GL 35Ø GOTO 19Ø
LF 1ØØØ REM Same tones with and witho
      ut envelope
```

Program 2: Chords With And Without ADSR Envelope

```
PK Ø GOTO 1ØØ
KM 1Ø REM Sound envelope
NL 2Ø FOR I=1 TO 8
CN 21 TRAP 23:R=VAL(SØ$(2*I-1,2*I)):T
      RAP 4ØØØØ
LE 22 POKE 53761,16Ø+R:POKE 53763,16Ø
      +R:POKE 53765,16Ø+R:POKE 53767,
      16Ø+R:NEXT I
EF 23 RETURN
MI 1ØØ DIM SØ$(5Ø)
OB 11Ø SØ$="151208Ø8Ø8Ø4Ø2ØØ"
DH 19Ø POKE 53760,243:POKE 53762,193:
      POKE 53764,144:POKE 53766,121
HE 2ØØ GOSUB 2Ø
DG 29Ø POKE 53760,243:POKE 53762,182:
      POKE 53764,144:POKE 53766,121
HF 3ØØ GOSUB 2Ø
AE 3Ø5 Q=1^1^1^1
CM 31Ø POKE 53260,243:POKE 53762,193:
      POKE 53764,144:POKE 53766,121
FD 312 POKE 53761,168:POKE 53763,168:
      POKE 53765,168:POKE 56767,168
OD 32Ø Q=1^1
DE 322 POKE 53761,16Ø:POKE 53763,16Ø:
      POKE 53765,16Ø:POKE 56767,16Ø
CM 33Ø POKE 53260,243:POKE 53762,182:
      POKE 53764,144:POKE 53766,121
FF 332 POKE 53761,168:POKE 53763,168:
      POKE 53765,168:POKE 56767,168
AD 34Ø Q=1^1^1^1
GL 35Ø GOTO 19Ø
MD 1ØØØ REM Two chords with and witho
       ut envelope
```

Program 3: Even Harmonics With And Without ADSR Envelope

```
PK Ø GOTO 1ØØ
KM 1Ø REM Sound envelope
NL 2Ø FOR I=1 TO 8
CN 21 TRAP 23:R=VAL(SØ$(2*I-1,2*I)):T
      RAP 4ØØØØ
LE 22 POKE 53761,16Ø+R:POKE 53763,16Ø
      +R:POKE 53765,16Ø+R:POKE 53767,
      16Ø+R:NEXT I
EF 23 RETURN
MI 1ØØ DIM SØ$(5Ø)
OB 11Ø SØ$="151208Ø8Ø8Ø4Ø2ØØ"
ML 19Ø POKE 53760,243:POKE 53762,121:
      POKE 53764,6Ø:POKE 53766,4Ø
HE 2ØØ GOSUB 2Ø
KE 29Ø POKE 53760,162:POKE 53762,81:P
      OKE 53764,4Ø:POKE 53766,27
HF 3ØØ GOSUB 2Ø
AE 3Ø5 Q=1^1^1^1
MF 31Ø POKE 53760,243:POKE 53762,121:
      POKE 53764,6Ø:POKE 53766,4Ø
FD 312 POKE 53761,168:POKE 53763,168:
      POKE 53765,168:POKE 56767,168
OD 32Ø Q=1^1
DE 322 POKE 53761,16Ø:POKE 53763,16Ø:
      POKE 53765,16Ø:POKE 56767,16Ø
JP 33Ø POKE 53760,162:POKE 53762,81:P
      OKE 53764,4Ø:POKE 53766,27
FF 332 POKE 53761,168:POKE 53763,168:
      POKE 53765,168:POKE 56767,168
AD 34Ø Q=1^1^1^1
GL 35Ø GOTO 19Ø
MD 1ØØØ REM Even harmonics envelope a
       nd no envelope
```

Program 4: Odd Harmonics With And Without ADSR Envelope

```
PK Ø GOTO 1ØØ
KM 1Ø REM Sound envelope
NL 2Ø FOR I=1 TO 8
CN 21 TRAP 23:R=VAL(SØ$(2*I-1,2*I)):T
      RAP 4ØØØØ
LE 22 POKE 53761,16Ø+R:POKE 53763,16Ø
      +R:POKE 53765,16Ø+R:POKE 53767,
      16Ø+R:NEXT I
EF 23 RETURN
MI 1ØØ DIM SØ$(5Ø)
OB 11Ø SØ$="151208Ø8Ø8Ø4Ø2ØØ"
KL 19Ø POKE 53760,243:POKE 53762,81:P
      OKE 53764,49:POKE 53766,35
HE 2ØØ GOSUB 2Ø
KB 29Ø POKE 53760,162:POKE 53762,54:P
      OKE 53764,32:POKE 53766,23
HF 3ØØ GOSUB 2Ø
AE 3Ø5 Q=1^1^1^1
KF 31Ø POKE 53760,243:POKE 53762,81:P
      OKE 53764,49:POKE 53766,35
FD 312 POKE 53761,168:POKE 53763,168:
      POKE 53765,168:POKE 56767,168
OD 32Ø Q=1^1
DE 322 POKE 53761,16Ø:POKE 53763,16Ø:
      POKE 53765,16Ø:POKE 56767,16Ø
JM 33Ø POKE 53760,162:POKE 53762,54:P
      OKE 53764,32:POKE 53766,23
FF 332 POKE 53761,168:POKE 53763,168:
      POKE 53765,168:POKE 56767,168
AD 34Ø Q=1^1^1^1
GL 35Ø GOTO 19Ø
EM 1ØØØ REM Odd harmonics envelope an
       d no envelope
```

Program 5: Even And Odd Harmonics With ADSR Envelope

```
PK Ø GOTO 1ØØ
KM 1Ø REM Sound envelope
NL 2Ø FOR I=1 TO 8
CN 21 TRAP 23:R=VAL(SØ$(2*I-1,2*I)):T
      RAP 4ØØØØ
LE 22 POKE 53761,16Ø+R:POKE 53763,16Ø
      +R:POKE 53765,16Ø+R:POKE 53767,
      16Ø+R:NEXT I
EF 23 RETURN
MI 1ØØ DIM SØ$(5Ø)
OB 11Ø SØ$="151208Ø8Ø8Ø4Ø2ØØ"
KL 19Ø POKE 53760,243:POKE 53762,81:P
      OKE 53764,49:POKE 53766,35
HE 2ØØ GOSUB 2Ø
KB 29Ø POKE 53760,162:POKE 53762,54:P
      OKE 53764,32:POKE 53766,23
HF 3ØØ GOSUB 2Ø
AE 3Ø5 Q=1^1^1^1
MF 31Ø POKE 53760,243:POKE 53762,121:
      POKE 53764,6Ø:POKE 53766,4Ø
HI 312 GOSUB 2Ø
JP 33Ø POKE 53760,162:POKE 53762,81:P
      OKE 53764,4Ø:POKE 53766,27
HK 332 GOSUB 2Ø
AD 34Ø Q=1^1^1^1
GL 35Ø GOTO 19Ø
HL 1ØØØ REM Odd and even harmonics wi
       th envelope
```

How TurboTape Works

Harrie De Ceukelaire
With Ottis Cowper, Technical Editor, And Charles Brannon, Program Editor

Last month COMPUTE! unveiled "TurboTape," a breakthrough program that makes Commodore 64 and VIC-20 tapes save and load as fast as disks. Although it's not necessary to know how TurboTape works in order to use it, this month's article explains the inner workings of the technique for programmers and technicians.

How can an ordinary cassette drive transfer data as fast as a 1541 disk drive? A few months ago, the answer would have been that it can't. But that was before "TurboTape." If you tried the TurboTape program published in last month's COMPUTE!, you know that something unusual is going on. VIC and 64 tapes really do load as fast as 1541 disks—sometimes even faster.

But *how?* TurboTape seems to violate a long-standing rule in personal computing. Tapes are always slower than disks, right?

To understand how TurboTape works, it helps to first understand how normal tape SAVEs and LOADs operate. Commodore's scheme for storing data on tape is quite complex—probably the most sophisticated used by any microcomputer manufacturer. The benefit of this complexity is that the system is extremely reliable. While users of other computers are frequently frustrated by programs that won't load properly from tape, many Commodore tape users never see a ?LOAD ERROR message. The disadvantage is that the complex system leads to long waits for programs to load.

Most microcomputers use an analog tape format. Each byte of the file to be stored on tape is broken down into bits, which in turn are converted to short bursts of audio tones. Two distinct tones symbolize the two states of a bit, either a zero or a one. If you've read much about telecommunications, you'll realize this is the same trick used by modems to transfer data over phone lines.

Digital Squares

Commodore, on the other hand, uses a digital tape format. Rather than recording a particular frequency on the tape, a Commodore computer writes a pattern of square waves (called *dipoles* in Commodore's technical literature) on the tape. The two *poles* are created by alternately recording either a strong signal or an equal period of no signal at all. The Commodore system uses square wave patterns of three different periods (lengths): short, medium, and long. When reading the bits back in, the computer monitors the period of each of the waves, and can—within limits—correct for differences in the length of the dipoles caused by one tape drive running slightly faster or slower than another.

Each byte of data is preceded by a marker consisting of a long square wave followed by a medium one. A 0 bit is represented by a short wave followed by a medium wave, while a 1 bit is the opposite—a medium wave followed by a short one. Each byte on tape ends with a parity bit, which is either 0 or 1 as required to make the total number of 1 bits in the byte odd. The first few bits of a byte on tape might be represented graphically as shown in Figure 1.

Using the parity bit, each byte can be checked as it is retrieved from tape. If there is not an odd number of 1 bits in the byte plus its parity bit, an error results.

Figure 1: Standard Tape Dipole Patterns

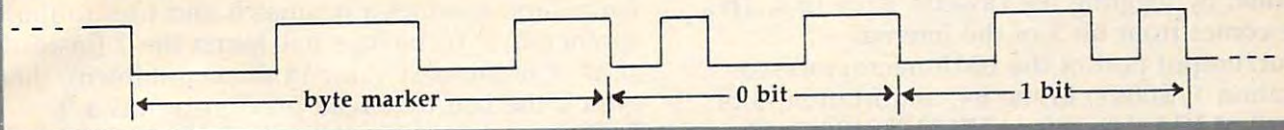


Figure 2: Commodore Tape Storage Layout

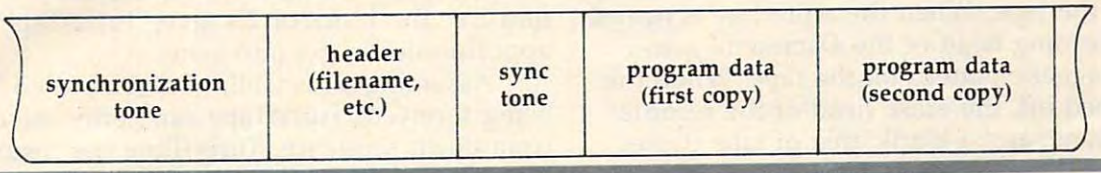


Figure 3: TurboTape Dipole Patterns

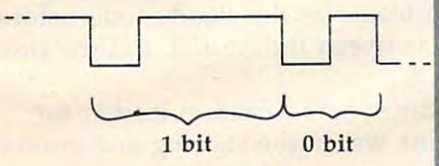


Figure 5: TurboTape Bit-Reading Technique

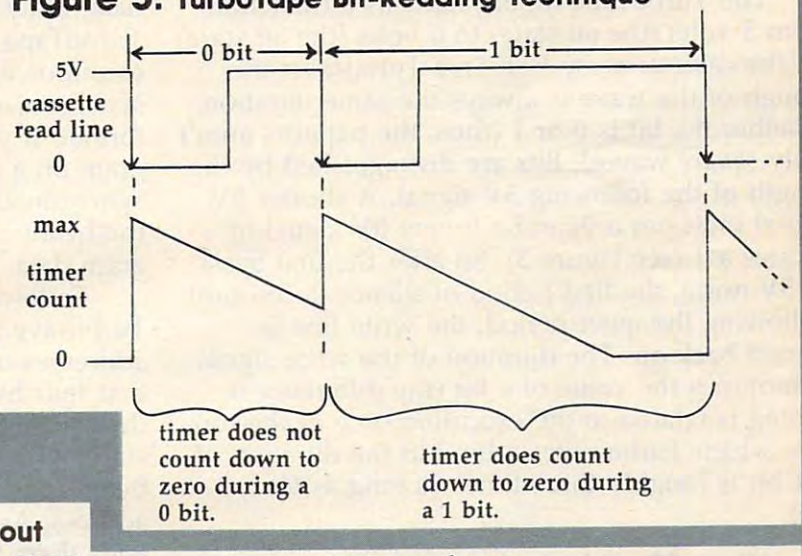
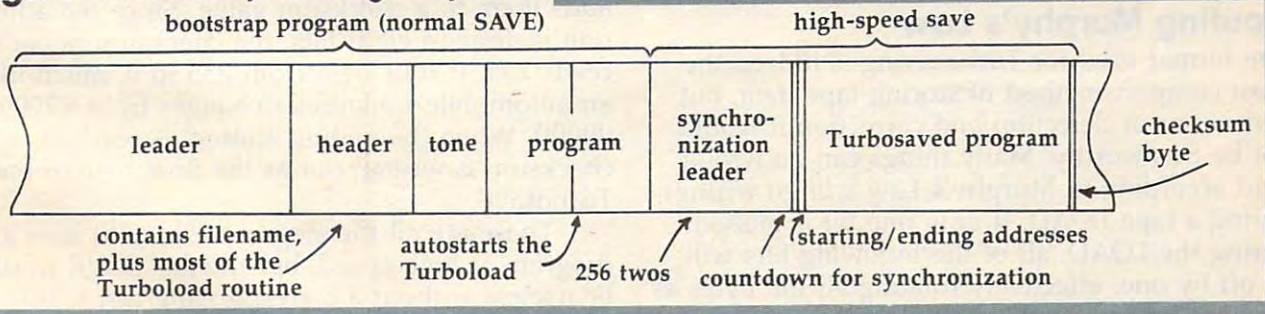


Figure 4: TurboTape Storage Layout



In addition, when you save a program on tape, the computer automatically records it *twice*, end to end. Graphically, a program stored on tape would have the layout shown in Figure 2. If an error is detected in the first recording, the computer remembers where the error occurred and corrects it with data from the second recording. You get the ?LOAD ERROR message only if more than 30 errors are detected on the first pass, or if there are errors in the first pass that can't be corrected in the second.

As you can see, the Commodore tape format is reliable because of its built-in error detection and correction. This, in turn, is the key to speeding up SAVEs and LOADs. Since you can't make the tape run faster, the only alternative is to change the recording format—cut back on Commodore's fail-safe mechanisms. TurboTape uses the bare minimum requirements to store data on tape. It's a method which is much like, yet much simpler than, Commodore's.

Turbowaves

TurboTape also creates a pattern of square waves on the tape, but instead of using a series of square waves to represent 0's and 1's, TurboTape uses a single square wave for each. The duration of the two square waves differs just enough to

permit the loading routine to distinguish between them. TurboTape records the square waves on tape in the same manner as the normal SAVE routine, by toggling the cassette write line. This line comes from bit 3 of the internal input/output port of the 6510 microprocessor (location 1/$0001) in the 64, and from bit 3 of port B of VIA 2 (location 37152/$9120) in the VIC. As long as RECORD and PLAY are pressed on the Datassette, this line controls the signal written to the tape. When the write line is turned on, the recording head of the Datassette generates a magnetic pattern on the tape. When the line is turned off, the erase head of the recorder operates alone, and a blank area of tape passes through.

The TurboTape dipole starts as a transition from 5 volts (the *on* state) to 0 volts (the *off* state) on the cassette write line. In a Turbosave, the trough of the wave is always the same duration, whether the bit is 0 or 1 (thus, the patterns aren't truly *square* waves). Bits are distinguished by the length of the following 5V signal. A shorter 5V signal indicates a 0, and a longer 5V signal indicates a 1 (see Figure 3). So after the first burst of 5V noise, the first period of silence is constant. Following the quiet period, the write line is turned back on. The duration of the write signal determines the value of a bit (the difference in timing is related to the execution time of the routine which Turbowrites a bit, but the duration of a 1 bit is roughly three times as long as for a 0 bit).

Flouting Murphy's Law

The format used for Turbosaving is indeed the most compact method of storing tape data, but without error detection and correction it would not be trustworthy. Many things can go wrong (and according to Murphy's Law *will* go wrong) during a tape LOAD. If only one bit is missed during the LOAD, all of the following bits will be off by one, effectively rotating all the bytes as they are loaded—not a pretty sight.

To help prevent this unbalance, TurboTape precedes the Turbosaved data with a series of synchronization bits. The synchronization leader consists of the byte value of 2 repeated 256 times, followed by a countdown of 9, 8, 7, 6, 5, 4, 3, 2, 1. During a LOAD, TurboTape looks for these bytes. It reads eight bits, then checks to see if the eight bits represent a value of 2. If a 2 is found, TurboTape checks for another 2. Sooner or later, TurboTape runs out of 2's and finds the 9 of the countdown sequence. TurboTape then continues, looking for the rest of the sequence.

Suppose that TurboTape missed one of the bits during synchronization. It would be left with a byte not representing a 2, even if a 2 had been written on tape. At this point, the byte had better be a 9, the start of the countdown, or TurboTape assumes an error. If an error is detected this way, TurboTape assumes a mismatch and tries to find another 2. If TurboTape has found the 2 (instead of an 8 as the next value in the countdown), then even if the bad byte read previously was a 9, TurboTape knows that it was a false 9, not the start of the countdown. As long as the countdown sequence fails, Turbotape keeps trying to find 2's. The block of 2's gives TurboTape 256 opportunities to get into sync.

Assuming all is well, once 2's are no longer being received, TurboTape can verify the correct countdown sequence. TurboTape has insured that it is synchronized with the first bit of actual data. Only if the countdown is mangled will TurboTape fail to synchronize. This leader and countdown system is similar to the one used to synchronize tape reading in the regular SAVE format. If you've ever listened to a stored program on a regular recorder, you've heard the synchronization leader as the steady tone before the header and between the header and the program data.

Following the synchronization leader, the Turbosave routine writes the starting and ending addresses of the program. These are stored as the first four bytes of Turbosaved data. After writing the starting and ending addresses, TurboTape starts writing out bytes from memory, taking the bytes apart bit by bit, beginning at the starting address. As these bytes are written, TurboTape adds them to a checksum value. Since the addition is done in eight bits, the checksum never exceeds 255. It rolls over from 255 to 0, much like an automobile's odometer changes from 99999 to 00000. When the ending address is reached, a checksum is written out as the final byte of the Turbosave.

These are all the steps necessary to save a program at high speed, but the fast SAVE would be useless without a corresponding fast LOAD routine to retrieve the data. And you would lose all the timesaving advantage of the fast SAVE if the fast LOAD routine had to be loaded into memory separately each time you needed to bring a program in from tape. Fortunately, TurboTape provides a loading routine that is transparent to the user.

By Its Own Bootstraps

Each Turbosaved program is preceded on tape by a bootstrap program stored using the normal SAVE format. The bootstrap program contains the entire high-speed loader, so the TurboTape software is not needed to load a Turbosaved program. But how does a normal LOAD become a Turboload?

The portion of the bootstrap program actually saved as a program is quite short: 10 bytes in the 64 version and 14 bytes in the VIC version. The data is saved in nonrelocatable format, so it always loads beginning at location 812 ($032C). It may not be obvious, but this provides a simple but sophisticated way to make the regular LOAD automatically start the Turboload.

One of the last steps the computer takes when completing a standard LOAD is to call the CLALL (CLose ALL files) subroutine in the operating system ROM. CLALL passes through an indirect vector at addresses 812–813 ($32C–32D), but those addresses have been changed by the data from the bootstrap program, so that execution is passed to the start of the Turboload routine at 814 ($32E). However, the few bytes starting from location 814 obviously aren't enough to decipher the data Turbosaved on tape. The major portion of the Turboload machine language routine is in the cassette buffer.

How it gets there is another interesting story. You may not be aware of it, but every program stored on tape has a filename 187 characters long. Each program written to tape by the normal SAVE routine is preceded by a 192-byte header (see Figure 2). The length corresponds to the 192 bytes of the cassette buffer (locations 828–1019). The first five bytes of every tape header are used for a one-byte identifier, a two-byte starting address for the saved program, and a two-byte ending address. The remaining 187 bytes are available for the filename, although only the first 16 are commonly used.

The Turbosave routine makes use of this by filling all the locations after the sixteenth byte of the filename (starting at location 849) with the remainder of the Turboload machine language, where it is written out as part of the filename when the bootstrap program is saved. When the filename is found during the LOAD process, all the data in the program header is loaded into the cassette buffer. Thus, the few bytes of regularly saved data need do little more than transfer control to the remainder of the routine in the buffer. The complete layout of a Turbosaved program would be as shown in Figure 4.

Time Out For Reading

To read a bit, TurboTape makes use of several features of the peripheral interface chips—the CIA (Complex Interface Adapter) on the 64, or the VIA (Versatile Interface Adapter) on the VIC. Each of these chips has a line (FLAG on the CIA and CA1 on the VIA) that can detect a high-to-low signal transition, the beginning of a dipole. These are used as the cassette read lines to the Datassette. To detect the start of a dipole, the Turboload routine monitors bit 4 of location 56333 ($DC0D) on the 64, or bit 1 of location 37165 ($912D) on the VIC. This bit will be set to 1 when the signal being read from tape changes from 5 volts to 0 volts, called the *falling edge* of the dipole (see Figure 5).

To determine whether the bit being read is a 0 or a 1, the Turboload routine starts a timer when the start of the dipole is detected. Each interface adapter chip has two 16-bit timer clocks. On the 64, Timer 2 of CIA #2 is used; the VIC version uses Timer 1 of VIA #1. The timers are like the familiar kitchen timers—they are set for the desired time and allowed to run until the time expires (until they count down to 0). The scheme is to set the timers for a period that is longer than the span of a 0 bit dipole, but shorter than the span of the dipole for a 1 bit. Then, when the next falling edge is detected, the status of the timer is checked. If the timer counted down to 0 before the start of the next dipole, then the time for the bit read was longer than the timer count and thus it was a 1 bit. If the timer is still counting when the next dipole starts, then the length of the dipole being read was shorter than the specified timer count, and thus it was a 0 bit.

The status of the timer can be determined by checking bit 1 of location 56589 ($DD0D) on the 64, or bit 6 of location 37149 ($911D) on the VIC. These will be 0 if the timers are still counting, or 1 if the timers have counted down to 0, which corresponds to the value being read from tape. By collecting these into groups of eight, the bytes of the program can be reassembled. The process is illustrated in Figure 5.

Turboverify operates by reading from tape the bootstrap program for the Turbosaved program to be verified, then modifying some of the Turboload code. It overwrites a store instruction with a compare and branch instruction. Thus, when the Turboload routine takes over, data read from the tape is only compared to the data already in memory, instead of being loaded over the existing data.

The Price Of Speed

After all the program data bytes have been read, one final value is retrieved from the tape. This byte is the checksum previously calculated during the Turbosave. This is the only error detection performed after header synchronization. If the checksum calculated during the Turboload does not match the one read from the tape, the LOAD must have failed.

However, even a correct checksum does not validate a LOAD, because there's more than one way to arrive at a certain sum. Since 2 + 4 + 6 = 1 + 4 + 7, addition is not a fail-safe checksum method. So you must realize that this

speed enhancement does not come without a price. Nevertheless, we've found that the Commodore Datassette is still forgiving enough to make TurboTape reliable.

Unfortunately, the tape reading routines in the bootstrap program are specific to the CIA on the 64 and the VIA on the VIC, since the different chips must be accessed through different memory locations. Also, Turboload makes use of a number of ROM routines that are at different locations in the VIC and 64. So even though the high-speed portion of a Turbosaved program could be read by either machine, the Turboload routine is machine-specific. Since the VIC and 64 Turboload routines are entered automatically, neither routine will work on the wrong machine. There's just not enough room in the cassette buffer for a universal TurboTape LOAD routine that would work on both computers. This means that programs Turbosaved on a 64 can't be loaded into a VIC, and vice versa.

Bypassing Errors

TurboTape works fine in principle, but without a good link with the operating system, it would be cumbersome. For ease of use, TurboTape adds two commands to BASIC: TURBOSAVE (or TSAVE) and TURBOVERIFY (TVERIFY). The TurboTape program as published last month includes a built-in memory mover and relocator.

When you initialize TurboTape, it copies itself to the top of memory (or optionally beginning at location 52606 on the 64), then corrects all the absolute machine language references such as JMPs, JSRs, and address tables. This relocator actually accounts for 170 of the 812 bytes of machine language in TurboTape.

When you type in the command TURBOSAVE, why don't you get a syntax error? It's certainly not a BASIC command. The answer is that when BASIC sees TURBOSAVE, it knows that TURBO is not a BASIC statement, so it assumes that it is a variable. BASIC then looks for the end of the variable, ready to assign it a value. Suddenly, it finds the command SAVE embedded within TURBOSAVE. A command like SAVE is not allowed as part of a variable name, so BASIC prepares to report a syntax error by jumping with the error code through the indirect error vector, contained in locations 768–769 ($300–$301).

This vector normally points to the BASIC ROM error-handling routines, but this is where TurboTape steps in. When first run, TurboTape changes the error vector to point to the relocated TurboTape machine language. From then on, whenever an error happens, TurboTape gains control. If the error is not a syntax error, TurboTape passes it along to the ROM error routine as usual. (It stores the original contents of 768–769 in 678–679, and uses those locations as its own indirect error vector.) For a syntax error, TurboTape checks for either the SAVE or VERIFY token. Since BASIC has rejected TURBO as a variable, the CHRGET routine is left pointing to the token after TURBO. (CHRGET is used by BASIC to scan for characters in a command or program line. Each call returns a new character and sets up CHRGET to point to the next character.) That's how TurboTape detects the SAVE command.

In fact, almost anything can precede the SAVE (such as SPEEDSAVE or even PIZZASAVE), as long as it's seen as a variable. The token which BASIC points to after the variable must be either 148 (SAVE) or 149 (VERIFY); otherwise, TurboTape jumps back to the normal ROM routine and a ?SYNTAX ERROR is properly reported.

Normal SAVEs do not go to TurboTape, since they do not pass through the error routine. Even if a SAVE ends in an error, CHRGET would no longer be pointing to the token for SAVE. This is an extremely elegant way of adding commands to BASIC, and it wedges into BASIC without interfering with BASIC extensions that use CHRGET (such as the DOS wedge) or other system vectors. ©

C Regena

Programming Without A Math Background

"Computer literacy," a required class in many high schools and colleges, is often little more than a class in elementary programming. Programming, however, is really only a small part of computing. Equally odd is the fact that many of these computer literacy classes require courses in algebra, calculus, or some other form of advanced mathematics as a prerequisite. In what way would knowing the calculus help someone learn BASIC?

Why are so many young people (often younger than 15) good programmers even if they've never taken algebra? Clearly, advanced mathematics has little to do with programming.

Of course, you do need to know a little about numbers. You need to know how to count. In a BASIC program the lines are numbered, so you must know the order in which the lines will be executed. Nevertheless, if you think logically, you can even use NUM to automatically number your lines as you are typing and you won't even have to worry about the line numbers.

If you like to program graphics, you should also learn something about basic coordinate geometry. That's just a mathematical term for using a grid. There are 24 rows and 32 columns on a TI-99/4A screen. If you want to place a character in a certain position, you have to tell the computer which row and column.

You'll also encounter numbers in the form of codes. For example, each color on a TI is given a number from 1 to 16. In any CALL SCREEN or CALL COLOR statement where you need a color number, you can look on the color chart to see which number represents which color. There are also codes for color sets, sounds, and characters. But beyond that, the most basic knowledge of addition, subtraction, multiplication, and division will be all you'll need in most cases.

Using Numbers Efficiently

Some skill at recognizing number patterns will help make your programs more efficient. Remember, however, that as long as your program works, it is "correct." There are many ways to accomplish the same task.

For instance, if you can recognize a pattern in your programming statements or among the numbers, quite often you can reduce the number of statements required. Suppose you want to draw seven horizontal lines across the screen. The lines are to be in rows 4, 7, 10, 13, 16, 19, and 22. You could use seven CALL HCHAR statements. Notice, though, that the numbers are each separated by 3. If you start with row 4 and add 3 each time until you get to 22, you'll have the lines you want. A FOR-NEXT loop could draw these same lines in only three statements:

```
200 FOR ROW=4 TO 22 STEP 3
210 CALL HCHAR(ROW,1,95,32)
220 NEXT ROW
```

Here's another problem. Suppose you want to draw a flower in several places on the screen, and each flower takes five characters, two on top of three others. The flowers are scattered randomly, so there's no pattern to their placement. In this case, a subroutine to draw the flower would be appropriate. Before you enter the subroutine, you could specify the row and column positions in the variables R and C. In the subroutine, the CALL HCHAR statements (or CALL VCHAR) need to be expressed in terms of R and C. If the upper-left corner of the flower is in position R,C then the next square would be R,C+1. Below R,C is R+1,C and next to it would be R+1,C+1 then R+1,C+2. The subroutine would look like this:

```
500 CALL HCHAR(R,C,112)
510 CALL HCHAR(R,C+1,113)
520 CALL HCHAR(R+1,C,114)
530 CALL HCHAR(R+1,C+1,115)
540 CALL HCHAR(R+1,C+2,116)
550 RETURN
```

And each time you need a flower, you would call the subroutine like this:

```
700 R=3
710 C=8
720 GOSUB 500
```

Streamlining Your Code

Now let's say you're drawing snakes instead of flowers. The snake still takes five characters, but all in a horizontal line. The subroutine might look like this:

```
500 CALL HCHAR(R,C,112)
510 CALL HCHAR(R,C+1,113)
520 CALL HCHAR(R,C+2,114)
530 CALL HCHAR(R,C+3,115)
540 CALL HCHAR(R,C+4,116)
550 RETURN
```

Notice that there is a pattern among the numbers. In each statement the column number increases by 1 and so does the character number. The five CALL HCHAR statements can be changed to:

```
500 FOR A=0 TO 4
510 CALL HCHAR(R,C+A,112+A)
520 NEXT A
```

A young friend came to me with a program in which he was randomly choosing five words, then printing them on rows 5, 7, 9, 11, and 13. He had to keep track of the words, their placement (order), and the answers. One solution was to DIMension arrays of W$ and ANS$ where the element specified was also the row number—so he had W$(R) and ANS$(R), where R was 5, 7, 9, 11, and 13. This method is easy to understand and worked well, but we were running into memory problems. Those arrays were taking up space because we weren't really using all the elements.

Notice that there is a pattern to the numbers:

Word 1—Row 5
Word 2—Row 7
Word 3—Row 9
Word 4—Row 11
Word 5—Row 13

The row numbers increase by 2. If you multiply each word number by 2, they become 2, 4, 6, 8, 10. Now compare these numbers with 5, 7, 9, 11, 13. Each of the word numbers (multiplied by 2) is 3 less than the row numbers. Therefore, if we have a word number N, the row number would be 2*N+3.

Later in the program, if we know the row number R and want to find the word number, we need to relate 5, 7, 9, 11, 13 to 1, 2, 3, 4, 5. First subtract 3 from the row number, then notice that the result is 2 times the word number. Given the row number R, the word number is (R−3)/2.

Quite often, if you line up a group of numbers you can see a relationship or a pattern. You can usually use standard arithmetic operations to get from one column of numbers to the next. If there is a progression of numbers, you can use a FOR-NEXT loop with a certain STEP size to get the right series of numbers.

Programming A Reflection

In this month's example program, we'll see how numbers can be manipulated to simulate reflections in graphics. The program takes a design you draw in the upper-left quadrant of the screen and creates reflections in the other three quadrants. We don't want the pattern simply repeated (as in the "Quilt Squares" program), rather, we want to actually reverse the image.

First, you draw a design in an area defined by rows 2 through 12 and columns 6 through 16. For example, the drawing starts in row 12 and column 16. This particular square reflects onto the other quadrants in squares (12,17), (13,16), and (13,17). The top-left square of the drawing quadrant is (2,6), or row 2 and column 6. The corresponding squares in the other quadrants are (2,27), (23,6), and (23,27).

In general, for a certain row R and column C, the corresponding square in the upper-right quadrant would be on the same row R and the column number would be 17 (the quadrant starts in the seventeenth column) plus (16−C). The first quadrant ends in column 16, and you subtract the first quadrant's column number to get its distance from the center. The result is 33−C. Another way to look at it is that the column number will be the same distance from the last column as the original square is from the first column—thus 32−C+1 or 33−C.

The corresponding square in the lower-left quadrant will have the same column number C as the original square, but the row will be 12+13−R or 24+R−1, which is 25−R. The lower-right quadrant has the same row as the lower-left quadrant and the same column as the upper-right quadrant. Thus the three corresponding squares are (R,33−C) and (25−R,C) and (25−R,33−C). Lines 620–640 and 1000–1020 use these relationships.

Electronic Snowflake

When I was a child I liked to fold paper, cut a design, then unfold the paper to see what it looked like. Sometimes we would fold the paper to get a six-sided snowflake. Other times we

would fanfold the paper. We also used different variations of simply folding the paper into rectangles.

This "Snowflake" program is the computerized version of cutting paper snowflakes (with no scraps of paper to clean up). Suppose you have a square piece of paper. Fold it in half to make a rectangle, then fold the rectangle in half to make a square. Now cut a design in that square. Unfold the paper and you have a four-sided snowflake.

When you run this program, you will see a large square outlined. You can draw in the upper-left square only. Use the arrow keys to move the cursor, press F to fill the cursor position with color, and press the space bar to preserve the background color (or to erase a previously filled position). When your design is complete, press ENTER. The computer starts at the center and moves outward to reflect your pattern on the other quadrants of the larger square.

When the design is complete, you can press M to modify, S to start a new pattern, P to print the pattern if you have a printer, and ENTER to end the program. If you press M to modify, the cursor starts blinking again and you can resume drawing. But this time your changes appear immediately in the rest of the design. When you're finished, you can press ENTER again. If you press S to start a new pattern, the screen clears and you can start over.

To use the printer option, the printer must be attached and switched on (don't forget the RS-232 interface). Line 800 contains the printer configurations; modify it if necessary. The hard copy printout is elongated but shows the pattern you drew. Filled squares are represented by asterisks and the blanks by dots. If you want, you could even use this pattern for counted cross-stitching or needlepoint.

Program Explanation

Lines 110–200 clear the screen and print the title and instructions. Lines 210–270 define characters used as graphics. Characters 96–99 are used to outline the large square and the drawing quadrant. Character 104 is the filled square, and character 105 is the cursor used in drawing. Character 112 is the yellow dot used to indicate the ENTER key after the snowflake is complete. Lines 280–290 define the colors for the snowflake and the ENTER key symbol. If you wish to use different colors for the snowflake, change the color number 5 in line 280 and the screen color in line 430.

Lines 300–390 wait for you to press ENTER, then continue the instructions. Lines 400–410 wait for you to press any key to start. Lines 420–490 clear the screen, change the screen color to cyan (light blue), then outline the large square and the upper-right quadrant.

Lines 500–510 define the starting row X and column Y for the drawing cursor. Lines 520–540 call the subroutine that is the procedure for moving and filling in squares until the ENTER key is pressed.

When you press ENTER, lines 550–570 make a beeping sound, then erase the lines for the quadrant. Lines 580–660 look at each square in the upper-right quadrant. If they find a filled square, they draw a square in the other quadrants in the corresponding position. This happens in loops, starting with the center square and moving outward (by columns C) and upward (by rows R). When the process is complete, line 670 sounds another beep.

Lines 680–730 print the options to press M for modify, S to start over, P to print, or ENTER to end. Lines 740–780 detect the key pressed and branch accordingly.

Lines 790–920 contain the printing option. You must have a printer connected, and your printer configuration must be specified in line 800. The computer looks at each row from 2 to 23 and each column from 6 to 27 using CALL GCHAR, and then prints a period for a space and an asterisk for a filled square. After the printing is complete, the program branches back to the options of M, S, P, and ENTER.

Lines 930–1030 contain the modify option. First the options at the right of the square are cleared. Then the drawing cursor reappears. Design changes instantly appear in the other three quadrants. When you press ENTER, the program branches back to the options of M, S, P, and ENTER.

Lines 1040–1320 contain the subroutine for the drawing procedure. CALL GCHAR checks to see what character is in position X,Y and calls that character number (G). Lines 1060–1080 blink the cursor while waiting for a keypress. Lines 1090–1300 are the branching statements executed when certain keys are pressed. Line 1310 draws the new character if it is a space or a filled square.

Lines 1330–1340 clear the screen, then end the program.

If you wish to save typing, you can receive a copy of this program by sending a blank cassette or disk, a stamped, self-addressed mailer, and $3 to:

C. Regena
P.O. Box 1502
Cedar City, UT 84720

Please be sure to specify that you need the TI version of Snowflake.

Snowflake

Refer to "COMPUTE!'s Guide To Typing In Programs" before entering this listing.

```
100 REM  SNOWFLAKE
110 CALL CLEAR
120 PRINT TAB(9);"SNOWFLAKE":::
130 PRINT "USE THE ARROW KEYS TO DR
    AW"
140 PRINT :"IN THE UPPER LEFT QUADR
    ANT."
150 PRINT :"PRESS 'F' TO FILL A SQU
    ARE."
160 PRINT :"PRESS SPACE BAR TO ERAS
    E."
170 PRINT :"PRESS <ENTER> WHEN YOU"
180 PRINT :"ARE FINISHED DRAWING."
190 PRINT ::"THE COMPUTER WILL COMP
    LETE"
200 PRINT :"THE SNOWFLAKE."
210 CALL CHAR(96,"0000000000000000FF")
220 CALL CHAR(97,"8080808080808080")
230 CALL CHAR(98,"FF")
240 CALL CHAR(99,"0101010101010101")
250 CALL CHAR(104,"FFFFFFFFFFFFFFFF
    ")
260 CALL CHAR(105,"FF818181818181FF
    ")
270 CALL CHAR(112,"3C7EFFFFFFFF7E3C
    ")
280 CALL COLOR(10,5,1)
290 CALL COLOR(11,12,1)
300 PRINT ::"PRESS <ENTER>."
310 CALL KEY(0,K,S)
320 IF K<>13 THEN 310
330 CALL CLEAR
340 PRINT "AFTER SNOWFLAKE IS COMPL
    ETE,"
350 PRINT :"PRESS <M> TO MODIFY PAT
    TERN"
360 PRINT :"PRESS <S> TO START OVER"
370 PRINT :"PRESS <P> TO PRINT COPY"
380 PRINT :"PRESS <ENTER> TO END."
390 PRINT :::::"PRESS ANY KEY NOW T
    O START."
400 CALL KEY(0,K,S)
410 IF S<1 THEN 400
420 CALL CLEAR
430 CALL SCREEN(8)
440 CALL HCHAR(1,6,96,22)
450 CALL VCHAR(2,28,97,22)
460 CALL HCHAR(24,6,98,22)
470 CALL VCHAR(2,5,99,22)
480 CALL VCHAR(2,17,97,11)
490 CALL HCHAR(13,6,98,11)
500 X=12
510 Y=16
520 CALL SOUND(150,1397,2)
530 GOSUB 1050
540 IF K<>13 THEN 530
550 CALL SOUND(100,1497,2)
560 CALL HCHAR(13,6,32,11)
570 CALL VCHAR(2,17,32,11)
580 FOR R=12 TO 2 STEP -1
590 FOR C=16 TO 6 STEP -1
600 CALL GCHAR(R,C,H)
610 IF H=32 THEN 650
620 CALL HCHAR(R,33-C,H)
630 CALL HCHAR(25-R,C,H)
640 CALL HCHAR(25-R,33-C,H)
650 NEXT C
660 NEXT R
670 CALL SOUND(100,440,2)
680 CALL VCHAR(8,29,60,4)
690 CALL VCHAR(8,31,62,4)
700 CALL HCHAR(8,30,77)
710 CALL HCHAR(9,30,83)
720 CALL HCHAR(10,30,80)
730 CALL HCHAR(11,30,112)
740 CALL KEY(0,K,S)
750 IF S<1 THEN 740
760 IF K=83 THEN 420
770 IF K=13 THEN 1330
780 IF K<>80 THEN 930
790 REM  PRINTER CONFIGURATION
800 OPEN #1:"RS232.BA=600"
810 FOR R=2 TO 23
820 FOR C=6 TO 27
830 CALL GCHAR(R,C,H)
840 IF H<>32 THEN 870
850 PRINT #1:".";
860 GOTO 880
870 PRINT #1:"*";
880 NEXT C
890 PRINT #1:CHR$(13)
900 NEXT R
910 CLOSE #1
920 GOTO 670
930 IF K<>77 THEN 740
940 CALL VCHAR(8,29,32,4)
950 CALL VCHAR(8,30,32,4)
960 CALL VCHAR(8,31,32,4)
970 CALL SOUND(150,1397,2)
980 GOSUB 1050
990 IF K=13 THEN 670
1000 CALL HCHAR(X,33-Y,G1)
1010 CALL HCHAR(25-X,Y,G1)
1020 CALL HCHAR(25-X,33-Y,G1)
1030 GOTO 970
1040 REM  SUB TO DRAW
1050 CALL GCHAR(X,Y,G)
1060 CALL KEY(0,K,S)
1070 CALL HCHAR(X,Y,105)
1080 CALL HCHAR(X,Y,G)
1090 IF K=13 THEN 1320
1100 IF K=70 THEN 1300
1110 IF K=32 THEN 1280
1120 IF K<>88 THEN 1160
1130 IF X=12 THEN 1060
1140 X=X+1
1150 GOTO 1050
1160 IF K<>83 THEN 1200
1170 IF Y=6 THEN 1060
1180 Y=Y-1
1190 GOTO 1050
1200 IF K<>68 THEN 1240
1210 IF Y=16 THEN 1060
1220 Y=Y+1
1230 GOTO 1050
1240 IF K<>69 THEN 1060
1250 IF X=2 THEN 1060
1260 X=X-1
1270 GOTO 1050
1280 G1=32
1290 GOTO 1310
1300 G1=104
1310 CALL HCHAR(X,Y,G1)
1320 RETURN
1330 CALL CLEAR
1340 END
```

MACHINE LANGUAGE

Jim Butterfield, Associate Editor

Multiplication Part 2

In Part 1, we discussed a multiplication such as:

```
(x)             1 1 0 1 0
(y)                 1 0 1
-------------------------
                1 1 0 1 0
              0 0 0 0 0
            1 1 0 1 0
-------------------------
(z)     1 0 0 0 0 0 1 0
```

We indicated that the logic might most usefully work this way:

1. Set the product area (z) to zero.
2. Examine the highest bit of the multiplier (y).
3. If the bit is 1, add the multiplicand (x) into the product (z).
4. If the multiplier (y) has no more bits, quit.
5. Shift the product (z) left one bit.
6. Examine the next highest bit of the multiplier, and go to step 3.

Thus, we start with 11010, shift left to get 110100, add nothing, shift left to get 1101000, add 11010 to give 10000010, then quit. Answer: 10000010, or hex 82, or decimal 130.

Working Another Shift

That's not hard to do, but we have one more trick in our bag. Notice that the product is shifted left. We could test the bits of the multiplier (y) if we shifted it left, too. The highest bits would pop into the carry flag as we shifted, and we could test each bit with a BCC or BCS as it goes by.

Now—and this is the neat part—if we need to shift both the product and the multiplier left, maybe we could put them together and shift them as one large collection of bits. We can see this best graphically:

```
00000101 00000000
Multiplier  Product
```

We'll shift these two as if they were one value. Whenever a bit hits the carry flag, we'll add 11010 (our multiplicand) into the product area. Nothing much will happen at first, since as we shift the two-byte group left, zeros will move into the carry and we won't add a thing. After five shifts, we have:

```
10100000 00000000
```

We still have nothing in our carry flag. But one more long shift, and the high bit will move into the carry:

```
C 01000000 00000000
```

Good! Add the multiplicand into the product area (using a full two-byte addition), and we'll get:

```
01000000 00011010
```

The next two left-shifts yield the following values:

```
          10000000 00110100
and   C   00000000 01101000
```

Aha! The carry bit has been hit again, so we add 11010 into the product area to get:

```
00000000 10000010
```

That's our answer! Correct in both bytes! We know to stop at this point because if we count the shifts we find that we've done eight—exactly the number of bits in the multiplier.

Taking A Bigger Byte

The elegant thing about this kind of multiplication is that the answer is correct over several bytes. For example, if you multiply a one-byte number by another one-byte number, the product may be up to two bytes in length. Our previous example was a simple one: 5 times 26 gives 130, which still fits into one byte. But if we try, say, 48 times 40, we'll need a two-byte area for the answer. Without special comment, let's do it using the same method:

```
    00101000  00000000
    01010000  00000000
    10100000  00000000
C   01000000  00110000
    10000000  01100000
C   00000000  11110000
    00000001  11100000
    00000011  11000000
    00000111  10000000
```

Answer: hex 780, or decimal 1920. Correct in both bytes.

Let's write the code to multiply a number in the A register with one in the X register and place the result in address $0380 (low) and $0381 (high). We'll use $0382 as storage for the multiplicand.

```
        STX   $0382   ;multiplicand
        STA   $0381   ;multiplier
        LDA   #$00
        STA   $0380   ;zero to product
        LDX   #$08    ;number of bits
NXBIT   ASL   $0380
        ROL   $0381
        BCC   NOADD
        CLC
        LDA   $0381
        ADC   $0382
        STA   $0381
        LDA   $0380
        ADC   #$00
        STA   $0380
NOADD   DEX
        BNE   NXBIT
```

It's elegant, it's efficient, and it easily extends to a greater number of bytes for the multiplier and multiplicand. ©

IBM Rebound

All Machine Language Game For PC & PCjr

Chris Metcalf and Marc Sugiyama

Here's a fast, smooth, all machine language adaptation of a classic arcade game. With the modifications included below, it runs on any IBM PC (color/graphics or monochrome adapter) with at least 64K RAM and a disk drive, and any Enhanced Model PCjr.

"Rebound" takes advantage of machine language to streamline the action in this arcade-style game patterned after the popular *Breakout*. By controlling a paddle at the bottom of the screen, your goal is to knock out all the bricks at the top of the screen with a bouncing ball.

Unlike most action games, Rebound works on all three popular types of IBM Personal Computers. Program 1 is for an IBM PC with the color/graphics adapter. Program 2 consists of modifications to make Program 1 work on an IBM PC with the monochrome adapter. And although Program 1 works as is on an IBM PCjr, the modifications contained in Program 3 accelerate the game to compensate for Junior's slower execution speed.

Typing Instructions

If you have a color/graphics PC, type in Program 1. If you have a monochrome PC, type in Program 1 and substitute the lines in Program 2. If you have a PCjr, type in Program 1 and substitute the lines in Program 3. *To be safe, save the program on disk before running it for the first time.*

Next, insert a disk in drive A and type RUN. The BASIC program will create a machine language file on disk with the filename REBOUND.EXE (the drive may whir on and off a few times as the file is created).

When the Ok prompt reappears, exit BASIC to DOS by typing SYSTEM. Make sure the disk with the REBOUND.EXE file is in drive A. To run Rebound, type REBOUND.EXE at the DOS prompt. Almost instantly, the game screen will appear.

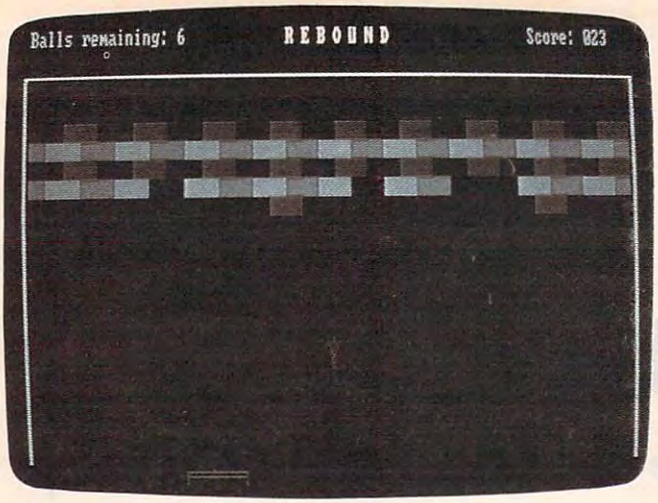
Eight Chances For Glory

To start playing Rebound, press the Enter key. The first ball starts moving downward from the middle of the screen. Your job is to keep it from falling off the bottom of the screen by bouncing it upward toward the rows of bricks.

To bounce the ball, move the paddle back and forth with the left and right Shift keys. You'll have to anticipate where the ball will bounce next, because the paddle can't always move across the screen as fast as the ball can (otherwise the game would be too easy). If you miss a ball, another one starts falling. You get a total of eight balls before the game ends in defeat. If you succeed in knocking out all the bricks, the program resets for another game.

There are five rows of bricks. Bricks on the lowest row are worth one point each, bricks on the second-lowest row are worth two points, etc.

You can freeze Rebound at any time by pressing the space bar. Press any other key to resume play.



"Rebound" is a fast arcade-style game that works on nearly all IBM Personal Computers.

The Esc key restarts a game in progress by replacing all missing bricks and lost balls. It also resets the score to zero.

To stop the game entirely and exit to DOS, press Ctrl-Break on the PC or Function-Break (Fn-B) on the PCjr.

Refer to "COMPUTE!'s Guide To Typing In Programs" article before typing these programs in.

Program 1: Rebound For IBM PC (Color/Graphics)

```
EL 10 ON ERROR GOTO 70
FB 20 OPEN "o",1,"rebound.exe"
CH 30 READ A : IF A<0 THEN 50
NE 40 PRINT #1,CHR$(A); : GOTO 30
EP 50 FOR I=1 TO -1*A : PRINT #1,CHR$(
      0); : NEXT
HJ 60 GOTO 30
QK 70 IF ERR <> 4 THEN ON ERROR GOTO 0
CB 80 CLOSE 1 : END
BF 100 DATA 77,90,32,1,4,0,1,0
KH 110 DATA 32,-3,255,255,74,0,128,0
ID 120 DATA 208,144,-4,32,-3,205,11,-2
PK 130 DATA 6,-479,30,184,-2,80,184,59
OK 140 DATA 0,142,216,142,192,176,3,18
      0
DN 150 DATA 0,205,16,185,-2,182,24,178
OC 160 DATA 79,176,0,183,7,180,6,205
FP 170 DATA 16,181,32,180,1,205,16,182
FD 180 DATA 3,178,3,183,0,232,107,3
DB 190 DATA 176,218,179,15,232,107,3,1
      78
BC 200 DATA 4,232,95,3,176,196,185,72
ME 210 DATA 0,232,97,3,178,76,232,82
LH 220 DATA 3,176,191,232,84,3,182,4
NA 230 DATA 178,3,232,70,3,176,179,232
FI 240 DATA 72,3,178,76,232,60,3,176
NN 250 DATA 179,232,62,3,254,198,128,2
      54
CG 260 DATA 24,114,229,178,4,182,1,190
DM 270 DATA 199,0,232,223,2,198,6,45
OD 280 DATA 0,56,191,3,0,198,133,94
QL 290 DATA 0,48,79,117,248,190,26,0
BP 300 DATA 232,179,2,198,6,10,0,92
EH 310 DATA 198,6,6,0,128,198,6,8
NH 320 DATA -2,191,5,0,182,5,178,4
JF 330 DATA 183,0,190,182,0,247,199,1
AF 340 DATA 0,117,3,190,189,0,185,9
OM 350 DATA 0,232,159,2,226,251,232,11
      3
OJ 360 DATA 2,79,117,224,128,62,11,-2
BL 370 DATA 117,18,198,6,11,0,1,232
NN 380 DATA 166,2,128,62,12,0,1,117
OB 390 DATA 3,233,106,1,254,14,45,0
BI 400 DATA 128,62,45,0,48,115,3,233
FJ 410 DATA 145,1,190,26,0,232,84,2
QP 420 DATA 182,24,178,0,190,225,0,232
HB 430 DATA 96,2,198,6,9,0,35,180
OC 440 DATA 0,205,26,139,194,37,255,15
GB 450 DATA 178,20,246,242,128,196,4,2
      46
QO 460 DATA 196,1,116,3,128,196,52,138
IL 470 DATA 212,182,11,137,22,-2,198,6
DB 480 DATA 4,0,2,128,252,39,119,5
BN 490 DATA 198,6,4,0,3,180,2,205
DH 500 DATA 22,36,3,116,25,60,3,116
IF 510 DATA 21,60,1,116,2,176,255,2
MH 520 DATA 6,9,0,60,2,124,7,60
BG 530 DATA 68,127,3,162,9,0,232,46
IC 540 DATA 1,160,6,0,44,127,177,10
JL 550 DATA 246,225,247,216,5,160,15,1
      39
KC 560 DATA 200,226,254,160,5,0,2,6
JP 570 DATA 6,0,162,5,0,114,3,233
QM 580 DATA 173,0,139,22,-2,160,4,0
FI 590 DATA 36,1,208,224,44,1,2,208
QK 600 DATA 160,4,0,36,2,44,1,128
CB 610 DATA 54,7,0,1,117,2,176,0
PC 620 DATA 2,240,128,254,24,119,39,18
      3
CB 630 DATA 0,232,3,2,180,8,205,16
FB 640 DATA 60,32,116,88,191,13,0,185
EK 650 DATA 4,0,252,242,174,117,41,232
AH 660 DATA 247,0,128,54,4,0,2,198
CP 670 DATA 6,7,-2,235,179,139,22,2
PH 680 DATA 0,232,213,0,139,22,-2,232
KL 690 DATA 206,0,232,175,0,180,14,176
EK 700 DATA 7,205,16,233,2,255,191,17
LH 710 DATA 0,185,3,0,242,174,117,5
BC 720 DATA 128,54,4,0,1,191,20,0
KA 730 DATA 185,6,0,242,174,116,3,233
JF 740 DATA 125,255,128,54,4,0,2,233
BA 750 DATA 117,255,128,54,8,0,1,117
JN 760 DATA 19,82,139,22,2,0,232,143
GK 770 DATA 0,90,232,124,0,137,22,-2
LH 780 DATA 235,9,144,135,22,-2,137,22
HB 790 DATA 2,0,180,1,205,22,116,39
MC 800 DATA 180,0,205,22,128,252,1,117
AC 810 DATA 8,198,6,11,-2,233,219,253
DK 820 DATA 60,32,117,7,180,0,205,22
KG 830 DATA 235,12,144,10,196,117,7,17
      6
GG 840 DATA 3,180,0,205,16,203,128,62
CI 850 DATA 10,-2,116,3,233,215,254,19
      0
PC 860 DATA 99,0,232,231,0,139,22,2
OD 870 DATA 0,232,56,0,139,22,-2,232
MF 880 DATA 49,0,232,18,0,232,7,1
NP 890 DATA 128,62,12,0,1,116,206,233
```

```
KG 900 DATA 151,253,190,128,0,235,218,
       182
GW 910 DATA 24,138,22,9,0,190,209,0
GF 920 DATA 232,206,0,195,82,183,0,232
FM 930 DATA 14,1,176,9,179,3,232,14
HN 940 DATA 1,90,195,82,183,0,232,255
JE 950 DATA 0,176,32,179,7,232,255,0
DD 960 DATA 90,195,254,14,10,0,83,81
LE 970 DATA 82,86,138,254,128,238,6,12
       8
QM 980 DATA 230,1,208,230,128,234,4,42
ME 990 DATA 214,128,226,252,2,214,185,
       4
OP 1000 DATA 0,121,5,178,0,185,2,0
PE 1010 DATA 128,194,4,138,247,128,250
        ,74
ED 1020 DATA 114,3,185,2,0,183,0,232
BE 1030 DATA 190,0,176,32,179,7,232,19
       3
AF 1040 DATA 0,183,11,42,254,2,62,97
GO 1050 DATA 0,136,62,97,0,183,58,190
FK 1060 DATA 2,0,58,188,95,0,119,18
DF 1070 DATA 138,156,95,0,128,235,10,1
       36
DM 1080 DATA 156,95,0,254,132,94,0,78
GM 1090 DATA 117,232,190,26,0,232,43,0
DH 1100 DATA 128,238,6,177,5,210,230,2
       46
CP 1110 DATA 214,58,54,6,0,114,4,136
HA 1120 DATA 54,6,0,94,90,89,91,195
AC 1130 DATA 82,176,1,181,5,177,4,182
PL 1140 DATA 23,178,75,183,7,180,7,205
JH 1150 DATA 16,90,195,83,173,139,208,
       183
GL 1160 DATA 0,232,84,0,172,60,0,116
HJ 1170 DATA 6,180,14,205,16,235,245,9
       1
CM 1180 DATA 195,81,86,183,0,232,64,0
DM 1190 DATA 172,60,0,116,15,138,200,1
       81
MD 1200 DATA 0,172,138,216,172,232,58,
       0
DB 1210 DATA 2,209,235,233,94,89,195,2
       32
DA 1220 DATA 19,0,190,161,0,232,195,25
       5
IB 1230 DATA 180,0,205,22,10,196,116,1
       8
FI 1240 DATA 128,252,28,117,243,139,22
        ,161
CA 1250 DATA 0,178,4,190,229,0,232,192
LJ 1260 DATA 255,195,198,6,12,0,1,195
PD 1270 DATA 80,180,2,205,16,88,195,18
       5
GO 1280 DATA 1,0,180,9,205,16,195,-19
EH 1290 DATA 219,219,219,219,179,218,1
       91,196
BP 1300 DATA 205,218,191,213,184,4,1,6
       6
PB 1310 DATA 97,108,108,115,32,114,101
        ,109
PF 1320 DATA 97,105,110,105,110,103,58
        ,32
JM 1330 DATA 0,32,32,32,32,32,32,32
MI 1340 DATA 32,32,32,32,32,82,32,69
GE 1350 DATA 32,66,32,79,32,85,32,78
JJ 1360 DATA 32,68,32,32,32,32,32,32
BI 1370 DATA 32,32,32,32,32,32,32,32
PF 1380 DATA 32,32,32,83,99,111,114,10
       1
ML 1390 DATA 58,32,-4,27,2,67,111,110
EN 1400 DATA 103,114,97,116,117,108,97
        ,116
HE 1410 DATA 105,111,110,115,33,32,32,
       89
LJ 1420 DATA 111,117,32,119,105,110,33
        ,0
EO 1430 DATA 25,2,84,114,105,117,109,1
       12
PA 1440 DATA 104,33,32,32,78,111,116,1
       04
NJ 1450 DATA 105,110,103,32,99,97,110,
       32
LF 1460 DATA 115,116,111,112,32,109,10
       1,33
GJ 1470 DATA 0,31,16,72,105,116,32,69
FD 1480 DATA 110,116,101,114,32,116,11
       1,32
CD 1490 DATA 66,101,103,105,110,0,4,1
HM 1500 DATA 219,4,2,219,0,2,4,219
OC 1510 DATA 4,6,219,2,4,219,0,20
DJ 1520 DATA 12,32,24,15,32,26,12,32
OA 1530 DATA 0,1,9,32,1,9,213,6
AD 1540 DATA 9,205,1,9,184,1,9,32
BE 1550 DATA 0,79,9,32,0,72,7,32
NO 1560 DATA -8,83,84,65,67,75,32,32
ML 1570 DATA 32,83,84,65,67,75,32,32
NO 1580 DATA 32,83,84,65,67,75,32,32
NB 1590 DATA 32,83,84,65,67,75,32,32
MI 1600 DATA 32,83,84,65,67,75,32,32
ML 1610 DATA 32,83,84,65,67,75,32,32
NO 1620 DATA 32,83,84,65,67,75,32,32
NB 1630 DATA 32,83,84,65,67,75,32,32
ME 1640 DATA 32,83,84,65,67,75,32,32
MH 1650 DATA 32,83,84,65,67,75,32,32
NK 1660 DATA 32,83,84,65,67,75,32,32
NN 1670 DATA 32,83,84,65,67,75,32,32
NA 1680 DATA 32,83,84,65,67,75,32,32
ND 1690 DATA 32,83,84,65,67,75,32,32
NK 1700 DATA 32,83,84,65,67,75,32,32
NN 1710 DATA 32,83,84,65,67,75,32,32
KG 1720 DATA 32
```

Program 2: Modifications For Monochrome PC

```
OM 1290 DATA 176,177,178,219,179,218,1
        91,196
GC 1490 DATA 66,101,103,105,110,0,4,15
NB 1500 DATA 176,4,7,177,0,2,15,178
FE 1510 DATA 4,7,219,2,15,178,0,20
IG 1520 DATA 7,32,24,15,32,26,7,32
JG 1530 DATA 0,1,7,32,1,7,213,6
JB 1540 DATA 7,205,1,7,184,1,7,32
PE 1550 DATA 0,79,7,32,0,72,7,32
```

Program 3: Modifications For PCjr

```
CN 540 DATA 1,160,6,0,44,127,177,6
BP 550 DATA 246,225,247,216,5,206,9,13
       9
```

Apple Bowling Champ

Original Program By Joseph Ganci
Apple Adaptation By Patrick Parrish, Programming Supervisor

Now you can go bowling without the expense of renting special shoes or suffering the embarrassment of rolling a gutter ball in front of dozens of people. "Bowling Champ" is a game for one to four players which runs on any Apple II-series computer.

Some computer games, such as *Pac-Man* or *Adventure*, create their own unique fantasy worlds, while others are simulations of reality. "Apple Bowling Champ" is an example of the latter.

It's not easy to take a game with countless physical variables such as bowling and reduce it to numbers so it can be re-created by a computer—especially a microcomputer. Compromises must be made. Usually the game must be modified in major ways to make it possible to program. The result is a hybrid game, an approximation of reality, that resembles the original but has new aspects of its own.

Apple Bowling Champ is a reasonable simulation of a game of tenpins, given the limitations imposed by a BASIC program which must remain short enough to publish in a magazine. The elements of skill and luck have been preserved, and the scoring is authentic.

Up To Four Players

When you run Bowling Champ, the program asks for the number of players. Up to four people can play. Next, enter the players' names. To fit the names on the 40-column screen, the program truncates entries to eight characters.

Now you're ready to bowl the first frame. The bowling ball moves rapidly up and down across the alley until you press the space bar. This rolls the ball down the alley and knocks over the pins—unless you've thrown a gutter ball. The trick is to time your release so the ball rolls down the center of the alley to score a strike.

In case you're unfamiliar with how a game of tenpins is scored, here's a brief summary.

A game consists of ten frames or turns. Each player gets one or two balls per frame. If you roll a strike—knocking down all tenpins with your first ball—you don't get a second ball, but the current ball's score is ten plus the total of your next two throws.

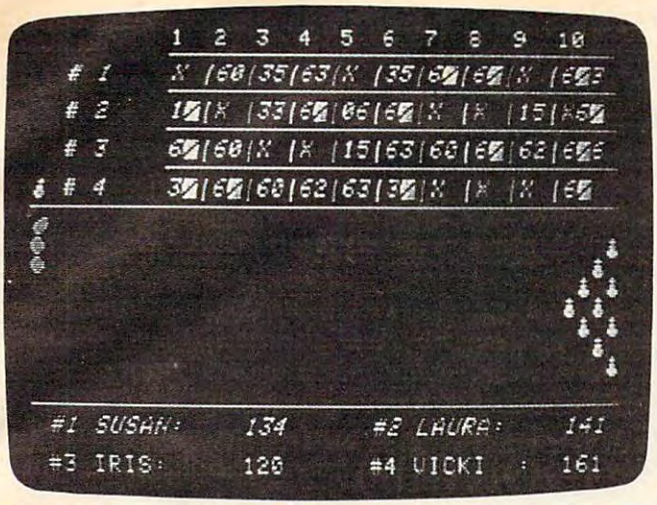
If some pins are left standing after your first ball, you get a second ball. If you knock down all the remaining pins, it counts as a spare, and the current ball's score is ten plus your next throw.

If any pins remain after your second ball (no strike or spare), the number of pins knocked down in that frame is added to your previous score.

Rolling a spare in the tenth (last) frame gains you one extra ball; rolling a strike in the tenth frame gains two extra balls.

Therefore, a perfect game—ten strikes during regular play plus two strikes with the extra balls—scores 300 points. Needless to say, this doesn't happen very often, either in real bowling or in Bowling Champ.

Since Bowling Champ follows every rule of scoring for regular bowling, you can learn how to score by carefully observing the game. The only



Four players compete in "Apple Bowling Champ": A blinking bowling pin next to a player's scorecard shows whose turn is next.

difference is that the computer does not wait until the end of a frame to update the score; it updates it after every ball.

Adjusting The Difficulty

Novice bowlers may find that the ball moves too fast for them to aim. On the other hand, more experienced players may want to speed up the ball to make the game harder. You can easily make either modification by changing the delay loop in line 480. The statement in 480 reads:

480 FOR R=1 TO 10:NEXT

Replacing the 10 with a larger number slows down the ball; a smaller number speeds up the ball. You might try a value between 20 and 50 for youngsters. For expert players, remove line 480 altogether.

Apple Bowling Champ

```
100  HIMEM: 36096: GOSUB 770
110  GOSUB 830: GOSUB 970
120  GOSUB 1160
130  HOME : POKE 230,32: CALL 62450: HGR
     : POKE 6,0: POKE 7,141: POKE 54,0:
      POKE 55,3: CALL 1002
140  GOSUB 1230
150  REM  -MAIN LOOP-
160  FOR Q = 1 TO 10: FOR Z9 = 0 TO A -
     1
170  FOR R = 1 TO 10: VTAB 2 * (Z9 + 1)
      + 1: HTAB 1: PRINT " ";: FOR F =
     1 TO 30: NEXT F: HTAB 1: PRINT "$"
     : FOR F = 1 TO 30: NEXT F: NEXT R
180 B1 = 0: GOSUB 360
190  IF J1 <  > 10 THEN B1 = 1: GOSUB 3
     90
200  IF Q = 10 THEN  ON S GOTO 210,270,
     270,210,310
210  VTAB (Z9 + 1) * 2 + 1: HTAB 1: PRINT
     " ": NEXT : NEXT : VTAB 24: HTAB 1
     0: POKE  - 16368,0: PRINT "PLAY AG
     AIN (Y/N)?";
220  IF  PEEK ( - 16384) < 128 THEN 220
230 K =  PEEK ( - 16384) - 128: IF K <
      > 78 AND K <  > 89 THEN  POKE  -
     16368,0: GOTO 220
240  IF K = 89 THEN 120
250  POKE  - 16368,0: HOME : TEXT : END
260  REM  -10TH FRAME : EXTRA BALLS-
270  VTAB 24: HTAB 5: PRINT "TAKE TWO M
     ORE BALLS, "NA$(Z9 + 1);".";
280  FOR I = 1 TO 2000: NEXT : VTAB 24:
      HTAB 5: PRINT  SPC( 30);
290 S(Z9) = S - 1:B1 = 1: GOSUB 360: IF
     J <  > 10 THEN 340
300  GOTO 330
310  VTAB 24: HTAB 5: PRINT "TAKE ONE M
     ORE BALL, "NA$(Z9 + 1);".";
320  FOR I = 1 TO 2000: NEXT : VTAB 24:
      HTAB 5: PRINT  SPC( 29);
330 S(Z9) = 1:B1 = 2: GOSUB 360: GOTO 2
     10
340 S(Z9) = 1:B1 = 2: GOSUB 390: GOTO 2
     10
350  REM  -FIRST BALL-
360  FOR I = 1 TO 10: VTAB A(I): HTAB B
     (I): PRINT "$": NEXT
370 PS = 1:J1 = 0: GOTO 400
380  REM  -SECOND BALL-
390 PS = 0
400  GOSUB 450:T = T(Z9):S = S(Z9):T =
     T + J
410  ON S(Z9) GOSUB 660,690,710,730,750
420 T(Z9) = T:S(Z9) = S
430  VTAB 21 + (A < 3) + 2 * (Z9 > 1) *
     (A > 2): HTAB 37 - (Z9 / 2 =  INT
     (Z9 / 2)) * 22: PRINT T(Z9): RETURN

440  REM  -ROLL BALL-
450 H = 1:C = 19:E = 11:D =  - 1: POKE
      - 16368,0
460  FOR V = C TO E STEP D: HTAB H: VTAB
     V: PRINT "*";
470  IF  PEEK ( - 16384) > 127 THEN T5 =
     V:V = E: NEXT : GOTO 510
480  FOR R = 1 TO 10: NEXT
490  HTAB H: PRINT " ";
500  NEXT V:D =  - D:T5 = C:C = E:E = T
     5: GOTO 460
510 V = T5: FOR H = 1 TO 35: HTAB H: VTAB
     V: PRINT " *";: FOR R = 1 TO 10: NEXT
     : NEXT
520 J = 0
530  IF ( SCRN( H,2 * (V - 1)) + 16 *  SCRN(
     H,2 * (V - 1) + 1) - 128) <  > 36 THEN
     570
540  POKE  - 16336,0:J = J + 1: FOR D =
      - 1 TO 1 STEP 2:X1 = V:X2 = H
550 X1 = X1 + D:X2 = X2 + 1: IF ( SCRN(
     X2,(X1 - 1) * 2) + 16 *  SCRN( X2,
     (X1 - 1) * 2 + 1) - 128) = 36 THEN
      HTAB X2 + 1: VTAB X1: PRINT " ";:
     J = J + 1: POKE  - 16336,0: GOTO 5
     50
560  NEXT
570  HTAB H: VTAB V: PRINT " *";:H = H +
     1: IF H < 40 THEN 530
580 J1 = J1 + J
590  VTAB 2 * Z9 + 3: HTAB 7 + 3 * Q +
     B1:G = J + 48
600  IF J1 <  > 10 THEN 630
610  IF PS THEN G = 88: GOTO 630
620 G = 47
```

```
630  PRINT  CHR$ (G)
640  HTAB H: VTAB V: PRINT " ";: RETURN
650  REM  -SCORING ROUTINES-
660  IF J1 <  > 10 THEN  RETURN
670  IF PS THEN S = 2: RETURN
680 S = 5: RETURN
690 T = T + J: IF J = 10 THEN S = 3: RETURN
700 S = 4: RETURN
710 T = T + J * 2: IF J <  > 10 THEN S =
    4
720  RETURN
730 T = T + J: IF J1 = 10 THEN S = 5: RETURN
740 S = 1: RETURN
750 T = T + J: IF J = 10 THEN S = 2: RETURN
760 S = 1: RETURN
770  DIM A(10),B(10): FOR I = 1 TO 10: READ
     A(I),B(I):X = X + A(I) + B(I): NEXT
     : IF X <  > 540 THEN  PRINT "ERROR
      IN DATA STATEMENTS FOR PIN POSITI
     ONS.": STOP
780  RETURN
790  REM  -PIN DATA-
800  DATA  12,40,13,39,14,38,14,40
810  DATA  15,37,15,39,16,38,16,40
820  DATA  17,39,18,40
830 X = 0: FOR I = 768 TO 852: READ A:X
     = X + A: POKE I,A: NEXT : IF X <
     > 7734 THEN  PRINT "ERROR IN DATA
     STATEMENTS FOR ML AT 768.": STOP
840  DATA  133,69,134,70,132,71,166,7
850  DATA  10,10,176,4,16,62,48,4
860  DATA  16,1,232,232,10,134,27,24
870  DATA  101,6,133,26,144,2,230,27
880  DATA  165,40,133,8,165,41,41,3
890  DATA  5,230,133,9,162,8,160,0
900  DATA  177,26,36,50,48,2,73,127
910  DATA  164,36,145,8,230,26,208,2
920  DATA  230,27,165,9,24,105,4,133
930  DATA  9,202,208,226,165,69,166,70
940  DATA  164,71,76,240,253
950  RETURN
960  REM  LOAD REDEFINED CHARACTERS
970 X = 0:AD = 36096: FOR L = 1 TO 16: READ
     B: FOR I = AD + B TO AD + B + 7: READ
     A:X = X + A: POKE I,A: NEXT :X = X
      + B: NEXT : IF X <  > 6223 THEN  PRINT
     "ERROR IN CHARACTER DATA STATEMENT
     S.": STOP
980  RETURN
990  DATA  0,0,0,0,0,0,0,0
1000  DATA  24,20,20,62,20,62,20,20,0
1010  DATA  32,8,28,8,28,28,62,62,28
1020  DATA  80,28,62,127,127,127,62,28,
     0
1030  DATA  120,63,31,79,103,115,121,12
     4,126
1040  DATA  128,28,34,50,42,38,34,28,0
1050  DATA  136,8,12,8,8,8,8,28,0
1060  DATA  144,28,34,32,24,4,2,62,0
1070  DATA  152,62,32,16,24,32,34,28,0
1080  DATA  160,16,24,20,18,62,16,16,0
1090  DATA  168,62,2,30,32,32,34,28,0
1100  DATA  176,56,4,2,30,34,34,28,0
1110  DATA  184,62,32,16,8,4,4,4,0
1120  DATA  192,28,34,34,28,34,34,28,0
1130  DATA  200,28,34,34,60,32,16,14,0
1140  DATA  448,34,34,20,8,20,34,34,0
1150  REM  TITLE SCREEN
1160  TEXT : HOME : VTAB 7: HTAB 12: PRINT
     "BOWLING CHAMP!": FOR I = 1 TO 4:N
     A$(I) = "": NEXT
1170  POKE  - 16368,0: VTAB 10: HTAB 7:
     PRINT "HOW MANY BOWLERS (1-4): ";
1180  IF  PEEK ( - 16384) < 128 THEN 11
     80
1190 A =  PEEK ( - 16384) - 128: IF A <
     49 OR A > 52 THEN 1160
1200  PRINT  CHR$ (A):A = A - 48: POKE
      - 16368,0: FOR I = 1 TO A: VTAB 1
     4 + I: HTAB 6: PRINT "BOWLER #"I"'
     S NAME: ";
1210  INPUT A$:NA$(I) =  LEFT$ (A$,8): NEXT
     : FOR I = 0 TO A - 1:T(I) = 0: NEXT
     : RETURN
1220  REM  DRAW GAME SCREEN
1230  VTAB 1: HTAB 10: PRINT "1  2  3
     4  5  6  7  8  9  10"
1240  HCOLOR= 3: HPLOT 63,11 TO 279,11
1250  FOR I = 1 TO A: VTAB 2 * I + 1: HTAB
     3: PRINT "# "I
1260  FOR J = 12 TO 36 STEP 3: HPLOT 7 *
     (J - 1) + 3,I * 2 * 8 TO 7 * (J -
     1) + 3,I * 2 * 8 + 8: NEXT
1270  HPLOT 63,2 * I * 8 + 11 TO 279,2 *
     I * 8 + 11: NEXT
1280  FOR I = 1 TO A STEP 2: VTAB 20 +
     (A < 3) + I: HTAB 1: PRINT " #"I"
     "NA$(I)":";: IF NA$(I + 1) <  > ""
      THEN  HTAB 23: PRINT " #"I + 1" "
     NA$(I + 1)":"
1290  NEXT I
1300  HPLOT 0,75 TO 279,75: HPLOT 0,155
      TO 279,155
1310  FOR I = 0 TO A - 1:S(I) = 1: NEXT
     : RETURN : REM  INITIALIZE SCORE S
     TATE
```

©

Advanced Sound Effects On The 64

Philip I. Nelson, Assistant Technical Editor

Here are some secrets to creating unusual sound effects with the Commodore 64's built-in synthesizer chip. Using the accompanying program, you can experiment with different sounds without programming.

The Commodore 64's SID (Sound Interface Device) chip is capable of creating rich, extraordinarily complex sounds—but its power doesn't come without a price. There aren't any sound commands in Commodore BASIC, so everything must be done with POKEs. It's tedious to look up all those POKE values and easy to get sidetracked, since you must define several *parameters* (controlling values) to make even a simple sound. Many programmers, including professionals, grow frustrated and settle for crude beeps and whooping noises, wasting the 64's classiest sound features.

The program following this article is designed to help beginners learn about two of the 64's advanced sound effects: ring modulation and synchronization. It lets you produce a tone with two sound channels, and also switch either effect on and off just by pressing one of the 64's special function keys. Don't worry if the following explanations seem confusing at first; they'll make more sense after you've tried the program.

Independent Voices

Any sound can be visualized as a *waveform*, like the cross section of a ripple on a pond. The Commodore 64 is capable of reproducing four different waveforms. Three of them (the triangle, sawtooth, and pulse waves) produce clear tones, and the fourth (the noise wave) makes a rushing or hissing sound. Figure 1 represents each of these waveforms. You can assign any one of the four waveforms to any of the 64's three sound channels, or *voices*.

Each of the computer's three voices normally plays independently. That is, each voice sounds the same, no matter what the other two are doing. If you make voice 1 beep and voice 2 growl, voice 1 always makes the same beep even if you change voice 2's growl to a screech. For a simple analogy, picture each voice as playing through a separate channel, like the two channels on a home stereo system.

Ring modulation and synchronization go beyond this to create *interactive* effects, in which a parameter controlling one voice also affects the sound produced by a second voice. In both cases, the special effect is created by a difference in the frequencies (pitches) of the two voices.

Synchronization

Synchronization is the simpler of the two effects. You may picture it as mixing two voices in one channel so that their waveforms intermingle. The result is often a rhythmic or beating effect, produced as the peaks and valleys of the two waves move in and out of step with each other.

When the two waves are more nearly in step, their combined sound is more pronounced. When their peaks and valleys are more nearly opposed, they tend to cancel each other out, and the combined sound is quieter. Figure 2 shows a simplified diagram of both extremes. If you program both voices so their frequencies are always identical, synchronization produces no audible effect.

In addition to the original tones each waveform produces by itself, synchronization adds nonharmonic *overtones* (also called *sidebands*). The overtones are entirely new waveforms which would not exist without synchronization. For instance, imagine someone pounding a huge gong. Gong sounds are full of nonharmonic overtones, which are created as different areas of the big, flexible metal plate vibrate in and out of phase.

In simplest terms, synchronizing two voices gives you both original tones plus new overtones. However, the original tones predominate.

Ring Modulation

Ring modulation is a special type of synchronization in which overtones almost completely suppress the original tones. What you're left with is a sound composed chiefly of nonharmonic overtones. The results are often surprising and bear little if any resemblance to so-called natural sounds.

Used with care, ring modulation can produce haunting, beautiful effects. However, it works through a complex interaction of two waveforms, largely suppressing what you'd hear without the feature. So it can be difficult to handle if you don't know how it works in the first place.

Experimenting With Effects

Let's hear how these effects sound. Type in the program, save it, and type RUN. The program is set up with several default parameters, so to hear a quick example, just press RETURN at every prompt. The default parameters will be displayed in each case.

You should hear a flutey tone sweeping up the scale, over and over. To pause the tone during its upward sweep, press the CTRL key. (Don't worry about accidentally hitting the RUN/STOP key; it's been disabled.)

To switch on synchronization, press the f7 special function key. The f5 key switches on ring modulation, and the f3 key activates both effects at once.

When synchronization is selected, you'll hear the beating effect as the tone ascends in pitch and the two voices move in and out of phase with each other. Ring modulation creates a rich, spacey sound. Note that you can pause the tone with CTRL while pressing a function key. As you'll hear, the sounds are far less exciting when both frequencies remain fixed. The most interesting effects are made by changing parameters in realtime.

In these two-voice effects, one of the voices is called the *carrier*, and the other the *program* voice. These terms are derived from electronics, meaning that the first voice *carries* the signal (produces the basic sound), and the second voice *programs* (modulates) it. In this example program, voice 1 produces the carrier tone, and voice 3 programs voice 1.

In both synchronization and ring modulation, it is the *frequency* of the program voice which affects the carrier voice. The other program voice parameters have no effect on the carrier (of course, they will affect the program voice if it is turned on).

Shifting Frequencies

Now that you've heard these special effects with the program voice set for a fixed frequency, let's try changing the frequency while the tone is being produced. To raise the frequency of the program voice, press either SHIFT key. To lower it, press the Commodore logo key (next to the left SHIFT). The most pronounced effects are produced by decreasing the program frequency during a rising tone, and vice versa.

Now let's hear a descending tone. Press the f1 key to stop the sound, and enter the following values when prompted:

Rising/falling?	**F**
Carrier waveform	**T**
Program waveform	**(any waveform works)**
Hear program voice?	**N**
Program frequency	**9**
Starting frequency	**200**
Ending frequency	**5**
Loop rate	**6**

Experiment with the program for a while, trying out different parameters. For example, try producing the same sound with a smaller loop rate. Press f1 to enter edit mode, then press RETURN after the first seven prompts. Now enter .75 for the loop rate. Pressing RETURN at a prompt preserves the old value, so you need to type in only the parameters you want to change (however, you must always enter the loop rate for a falling tone).

When picking the waveforms, press T for a triangle wave, P for the pulse waveform, and so on. When you select a rising tone, the starting frequency must be smaller than the ending frequency. To create a falling tone, the first value must be larger than the second. If you make a mistake, use the DELete key to back up. The program signals an error if you enter illegal values. If you accidentally type in a letter when a number is required, the computer prints ?REDO FROM START. No harm is done; just enter the number you want.

The loop rate controls how fast the carrier frequency is changed as the tone moves up or down the scale. It corresponds to the STEP value in the FOR-NEXT loop that creates the tone (see lines 13–17). The smaller the loop rate (fractions are allowed), the slower the frequency will change, and vice versa. When the starting and ending frequencies are far apart, you can specify a large value for the loop rate; however, if you specify a starting frequency that is close to the ending frequency, you must keep the loop rate small to avoid causing an error in the program.

Programming Your Own Sounds

You can use this program to start building a library of sound effects. Just play around until you

find a sound you like, copy down the values from the screen, and plug them into your own program.

As you'll discover by experimenting, these special effects work well with certain combinations, and poorly (or not at all) with others. Ring modulation works only when you set the carrier voice to the triangle waveform. Synchronization works with any waveform, but synchronizing any frequency with the noise waveform (a nearly random combination of many frequencies) doesn't accomplish much. The sawtooth and pulse waves often sound similar.

Most of the time, you'll want to keep the program voice silent, using only its frequency to control the carrier (in which case its other parameters are irrelevant). However, you can press Y when prompted to hear the program voice. If you have trouble understanding how an effect works, try listening to the program voice for a while.

Ring modulation and synchronization are most pronounced when the program frequency is considerably lower than the carrier frequency and remains fixed, as in the above examples. Changing the program frequency to a higher fixed value makes the two voices move in and out of phase more rapidly. Run the last example, and change the program frequency from 9 to 22. Now select synchronization, and you'll hear a sharp, *meow-meow* sound.

Controlling Voices With Voices

You can use ring modulation or synchronization with any of the 64's three voices, but the voice relationships are fixed: voice 1 modulates voice 2, voice 2 modulates voice 3, and voice 3 modulates voice 1.

Thus, if you want to synchronize or ring modulate voice 1, you must use voice 3 as the program voice, and so on. Again, it is the frequency of the program voice which affects the result. This simple tutorial program uses only the high byte frequency register for each voice; of course, you can achieve much finer frequency control by using both the high and low bytes.

To select these special effects in BASIC, simply add 2, 4, or 6 to the normal POKE value for the waveform register of the voice you want to affect. For instance, POKE 54276,17 selects the triangle waveform for voice 1. POKE 54276,19 adds synchronization to the triangle wave (17+2=19). POKE 54276,21 enables a ring-modulated triangle wave; and POKE 54276,23 turns on both effects at once. Use POKE 54276,67 to select synchronization with the pulse waveform, and so forth.

Naturally, you can use these effects with more than one voice at a time. If you select synchronization in voices 1 and 3, then voice 1 will be affected by voice 3's frequency, and voice 3 will be affected by voice 2's frequency. However, because multivoice modulation creates so many overtones, it's easy for things to get out of hand. If you create a three-note musical chord with triangle waves in every voice, and then switch each to ring modulation, the result will be anything but musical.

Play with those frequencies for a while, though, and you'll find you can push the *overtones* into complex chords. Such chords have a ringing, live sound, and contain more than three notes. Interesting effects can also be created by tuning one or more voices slightly off-key.

Hints For Programmers

This program employs a few tricks you might find useful. Many programmers use a long series of individual POKEs to set up the SID chip at the beginning of a program. Line 1020 shows how to do this with a FOR-NEXT loop that READs the values from DATA statements and POKEs them into the SID chip. This makes your program easier for others to read and for you to modify. Note, however, that Commodore recommends POKEing attack/decay registers before waveform registers; the program follows this rule by POKEing the desired waveform values later on, in line 370.

To detect a single keypress, you can PEEK location 197 as we did in lines 14 and 15 (Z=197). Sometimes, however, you want to let the user do two things at once from the keyboard. In this program, for instance, you can select effects with a function key and simultaneously change the program frequency or pause the sound.

By PEEKing location 653, you can tell whether the CTRL, SHIFT, or Commodore logo key is pressed with another key (see line 16; Y=653). Location 653 holds the following values when the indicated key is pressed:

1 = SHIFT
2 = Commodore
4 = CTRL

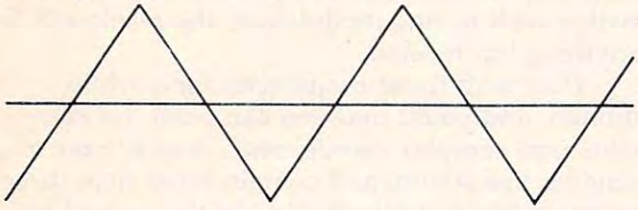
You can also detect combinations of these keys. Location 653 contains a 3 when both SHIFT and the Commodore key are pressed, 5 when SHIFT and CTRL are pressed, and so on. Checking for these keys gives you great flexibility in designing keyboard input. However, it's prudent to disable the RUN/STOP key when using them.

The program disables the RUN/STOP key in line 1010 with POKE 788,52. However, you can still exit the program by hitting RUN/STOP and RESTORE together. In the same line, POKE 657,128 prevents the computer from flipping the

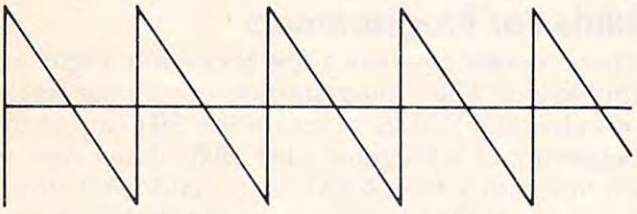
entire screen display from uppercase to lowercase if the SHIFT and Commodore keys are pressed simultaneously.

Figure 1: Commodore 64 Waveforms

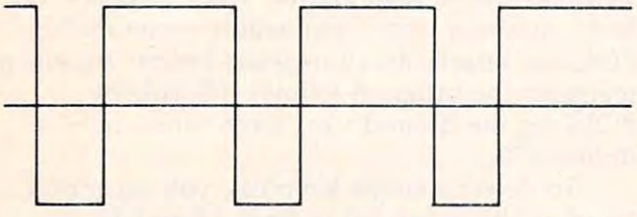
Triangle



Sawtooth



Pulse



Noise

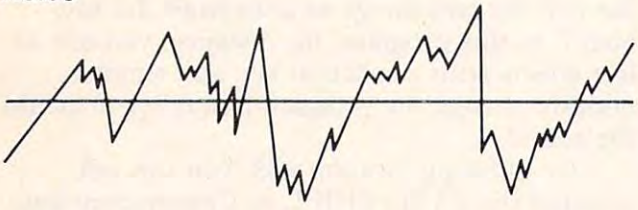
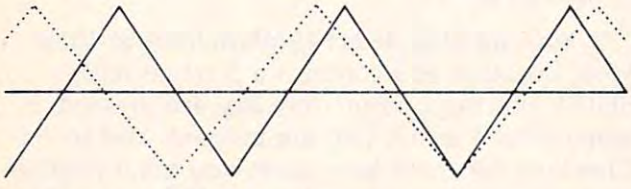
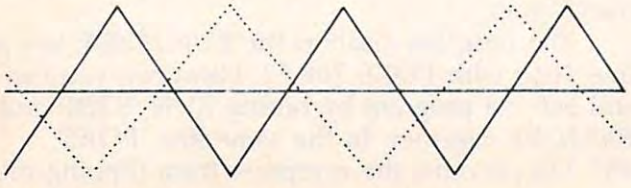


Figure 2: Synchronization

Waves nearly in step



Waves far out of step



Sound Effects Demonstrator

Please refer to "COMPUTE!'s Guide To Typing In Programs" before entering this listing.

```
1 GOSUB 1000:GOTO100                    :rem 118
2 PRINTCHR$(145)C$:FORJ=1TO400:NEXT:PRINT
  ER$CHR$(145):RETURN                    :rem 35
4 Z1=UN:ZZ=ED:RETURN                    :rem 109
5 ZZ=ZZ-LR:RETURN                       :rem 191
6 POKEW1,V1+TU:RETURN                   :rem 142
7 POKEW1,V1+FR:RETURN                   :rem 126
8 POKEW1,V1+SX:RETURN                   :rem 146
9 PF=PF+UN:IFPF>FFTHENPF=FF              :rem 39
10 RETURN                                :rem 65
11 PF=PF-UN:IFPF<UNTHENPF=UN            :rem 126
12 RETURN                                :rem 67
13 Z1=ZR:FORZZ=BGTOEDSTEPLR             :rem 121
14 IFPEEK(Z)=NNTHENPOKEW1,V1:GOTO16
                                         :rem 16
15 ONPEEK(Z)GOSUB10,10,6,4,8,7          :rem 202
16 ONPEEK(Y)GOSUB9,11,10,5               :rem 8
17 POKEH1,ZZ:POKEH3,PF:POKEBF,ZR:NEXT:IFZ
   1=UNTHEN19                           :rem 165
18 GOTO13                                :rem 6
19 POKEH1,ZR:POKEH3,ZR:POKEW1,ZR:POKEW3,Z
   R:POKE198,ZR                          :rem 29
100 PRINTFL$;:INPUTFF$                  :rem 137
110 IFFF$<>"R"ANDFF$<>"F"THENFF$="":GOSUB
    2:GOTO100                           :rem 184
120 PRINTUL$FF$:PRINTCV$;:INPUTVV$
                                        :rem 238
130 IFVV$<>"T"ANDVV$<>"S"ANDVV$<>"P"ANDVV
    $<>"N"THENGOSUB2:GOTO120             :rem 0
140 FORJ=1TO4:IFVV$=VL$(J)THENV1=VC(J)
                                        :rem 105
150 NEXT                                :rem 213
160 PRINTUL$VV$:PRINTPV$;:INPUTVW$:rem 32
170 IFVW$<>"T"ANDVW$<>"S"ANDVW$<>"P"ANDVW
    $<>"N"THENGOSUB2:GOTO160             :rem 12
180 FORJ=1TO4:IFVW$=VL$(J)THENV3=VC(J)
                                        :rem 112
190 NEXT                                :rem 217
200 PRINTUL$VW$:PRINTNF$;:INPUTYS$ :rem 9
210 IFYS$<>"Y"ANDYS$<>"N"THENGOSUB2:GOTO2
    00                                  :rem 158
220 IF YS$="N"THENV3=V3-UN               :rem 16
230 PRINTUL$YS$:PRINTPF$;:INPUTPF:rem 211
240 IFPF<UNORPF>FFTHENGOSUB2:GOTO230
                                        :rem 132
250 PRINTNL$PF:PRINTBG$;:INPUTBG :rem 122
260 IFBG<ZRORBG>FFTHENGOSUB2:GOTO250
                                        :rem 119
270 PRINTNL$BG:PRINTED$;:INPUTED :rem 111
280 IFED<ZRORBG=EDORED>FFTHENGOSUB2:GOTO2
    70                                  :rem 107
290 IFFF$="R"ANDED<BGTHENGOSUB2:GOTO270
                                        :rem 187
300 IFFF$="F"ANDED>BGTHENGOSUB2:GOTO270
                                        :rem 169
310 PRINTNL$ED:PRINTLR$;:INPUTLR :rem 148
320 IFLR<=ZRORLR>FFTHENGOSUB2:GOTO310
                                        :rem 216
330 IFFF$="R"ANDLR>ED-BGTHENGOSUB2:GOTO31
    0                                   :rem 126
340 IFFF$="F"ANDLR>BG-EDTHENGOSUB2:GOTO31
    0                                   :rem 115
350 IFFF$="F"THENLR=-LR                 :rem 115
360 PRINTNL$ABS(LR):PRINTCHR$(158)A$:PRIN
    TB$:PRINTF$:PRINTCHR$(158)A$   :rem 63
```

```
370 POKEH3,PF:POKEW1,V1:POKEW3,V3 :rem 82
380 GOTO13                        :rem 56
999 REM INITIALIZE                :rem 129
1000 PRINTCHR$(147)CHR$(5)CHR$(142):POKE5
     3281,0:POKE53280,0:Z=197:BF=198:Y=65
     3                            :rem 188
1010 POKE657,128:POKE788,52:S=54272:VM=S+
     24:FORJ=STOVM:POKEJ,0:NEXT   :rem 146
1020 FORJ=STOVM:READQ:POKEJ,Q:NEXT:rem 26
1025 FF$="R":BG=5:ED=125:LR=2:VV$="T":VW$
     ="T":PF=11:YS$="N"           :rem 102
1030 ZR=0:UN=1:TU=2:FR=4:SX=6:NN=64:FF=25
     5:H1=S+1:W1=S+4:H3=S+15:W3=S+18
                                  :rem 63
1040 R$=CHR$(18)                  :rem 51
1050 A$=R$+"{37 SPACES}"          :rem 77
1060 PRINTA$                      :rem 185
1070 PRINTR$"{4 SPACES}SOUND MODULATION D
     EMONSTRATOR{4 SPACES}"       :rem 54
1080 PRINTA$                      :rem 187
1090 B$=R$+CHR$(158)+" F7=SYNCH F5=RING F
     3=BOTH F1=RESTART "+CHR$(159)
                                  :rem 108
1095 F$=R$+CHR$(158)+" CTRL=PAUSE COM=FRE
     Q DN SHFT=FREQ UP "+CHR$(159)
                                  :rem 163
1100 C$=CHR$(158)+"{31 SPACES}"+R$+"ERROR
      "+CHR$(159)                 :rem 123
1105 ER$="{8 LEFT}{5 SPACES}"     :rem 235
1110 BL$=R$+CHR$(159)             :rem 68
1115 UL$=CHR$(145):FORJ=1TO31:UL$=UL$+CHR
     $(29):NEXT:UL$=UL$+"{2 SPACES}"
                                  :rem 104
1118 NL$=UL$+CHR$(157)            :rem 165
1120 FL$=BL$+" RISING OR FALLING TONE? (R
     ,F) "+CHR$(146)              :rem 228
1130 BG$=BL$+" STARTING FREQUENCY
     {4 SPACES}(0-255) "+CHR$(146):rem 80
1140 ED$=BL$+" ENDING FREQUENCY{6 SPACES}
     (0-255) "+CHR$(146)          :rem 154
1150 LR$=BL$+" LOOP RATE{13 SPACES}(1-255
     ) "+CHR$(146)                :rem 176
1160 CV$=BL$+" CARRIER WAVEFORM{4 SPACES}
     (T,S,P,N) "+CHR$(146)        :rem 132
1170 PV$=BL$+" PROGRAM WAVEFORM{4 SPACES}
     (T,S,P,N) "+CHR$(146)        :rem 162
1180 PF$=BL$+" PROGRAM FREQUENCY
     {5 SPACES}(1-255) "+CHR$(146):rem 15
1190 NF$=BL$+" HEAR PROGRAM VOICE?
     {5 SPACES}(Y,N) "+CHR$(146)  :rem 10
1200 FORJ=1TO4:READQ:VC(J)=Q:NEXT :rem 85
1210 FORJ=1TO4:READQ$:VL$(J)=Q$:NEXT
                                  :rem 203
1300 RETURN                       :rem 164
2000 DATA 5,0,128,7,0,15,240:REMVOICE1
                                  :rem 12
2010 DATA 0,0,0,0,0,0,0,0:REMVOICE2:rem 251
2020 DATA 5,0,128,7,0,15,240:REMVOICE3
                                  :rem 16
2030 DATA 0,0,0,15:REMFILTERS,VOLUME
                                  :rem 148
2040 DATA 17,33,65,129:REMWAVEFORMS
                                  :rem 17
2050 DATA T,S,P,N                 :rem 170 ©
```

Computers And Society

David D. Thornburg, Associate Editor

The Processed Word

My craft (such as it is) is writing. I write for a living, and I am able to live from my writing. I used to do other things for income, but when it comes right down to it, I am entranced by the power of words and by the ease with which two simple substances—paper and ink—can combine to form documents that can cause laughter, pain, joy, fear, and even boredom.

Pretty magical, this writing business.

Of course, as a writer, I have assembled a modest collection of writing tools—pads of paper, pens, typewriters, terminals, computers—the usual stuff.

One question writers ask from time to time is how their tools influence (dare I say determine?) what they write. Some critics argue, for example, that no great artistic works are going to be created on a word processor. These critics go on to suggest that the only good writing is done with tools like pencils, or perhaps typewriters.

In The Mind

These critics are confusing the tool with the result. The word processor will create no works of art at all. I use mine six hours a day, and it has yet to create anything of its own in its spare time. But then again, I don't expect my pens to create anything, and I am sure that our ancestors didn't expect wonders from sticks pressed into fresh mud, either.

Why the literary critics have missed the point eludes me. It probably comes from their own lack of exposure to a good word processing system. In fact, good writing takes place in the mind, not in the pen. I do my writing without ever lifting a pen, and then use whatever tool is at hand to transcribe this writing onto paper. Do my writing implements influence what I write? Perhaps they do, but only to a very small extent. (For example, I wouldn't be writing on this topic if I didn't use a word processor.)

As a writer, I have found that there are other factors that are much more influential than my choice of transcription tools. The first of these is exposure to good writing. I don't know any good writer who doesn't spend time reading other authors' books. Wherever I write, I have shelves lined with the works of others. Many of these books are technical and many are not. The shelf containing computer books also contains the works of Shakespeare. Aristotle shares shelf space with my Apple Logo manual. Exposure to good writing can be very important to an author—any author.

The second factor that I find as important as any other is having a good place to do my writing. I can write almost anywhere (on airplanes, for example), but my best writing comes when I am in a special place that is conducive to creative thought.

Each Its Own Charm

I am most fortunate to have three places that are conducive to writing. The first is a condominium high in the hills south of San Francisco. From my living room I can look into a verdant ravine, and trees fill my sight for as far as I can see. Further south, in an unused school in Mountain View, I have a rambling office that looks out on mulberry trees. The pitched roof of this office lends a certain resonance to the room when the winter rains beat against it. Inside, the spacious area has a warmth that encourages thoughts to flow. My third special place is in Monterey, in sight of the ocean. With wooded trails nearby, and miles of beach to explore, I find this area to be very encouraging to the creative process.

Each of these places has its own charm, but they all have one characteristic in common. In each of them I can find the quiet and solitude that seems to be important to me. When I am

---

David Thornburg has used several word processors to write 11 books, including The KoalaPad Book, Computer Art and Animation *(a Logo book available in versions for the TI, Radio Shack, Atari, and Commodore computers), and* Exploring Logo Without a Computer *(published by Addison-Wesley). His whimsical look at computing (*101 Ways to Use a Macintosh*) has been published by Random House. Later this year, his first book on artificial intelligence applications in Logo (*Beyond Turtle Graphics*) will be published by Addison-Wesley. Thornburg welcomes letters from readers, but regrets that he is not able to answer all his mail. Correspondence should be sent to him in care of* COMPUTE!.

writing, I can tolerate no distractions—no background music, no telephones, no conversation—nothing, save for the sounds of the birds, or seals, or the rustle of the leaves, or the patter of the rain.

So if you ask me to identify my most important tools as a writer, they would be access to the writing of others, and a quiet place in which to write.

So what about the technology of transcription? Doesn't it have a role?

Of course it does—but its role is largely neutral or negative. What I mean is that I am rarely liberated by my writing instruments; I am often impeded by them. As a tinkerer I am forever buying new pens, papers, word processors, and other tools, looking for ways to facilitate transcription. This is a highly personal quest. Some writers can dictate their manuscripts into a tape recorder. I cannot. Some writers prefer to use only No. 2 pencils and yellow legal pads. I do not.

The point is that there is no one best transcription tool, just as there is no one best writing style, or one best author.

While my use of tools changes from time to time, my process for transcribing a manuscript is roughly the following:

My first draft is often (but not always) written in longhand with a (gasp!) cheap fountain pen in a blank book or legal pad. I persist in using a fountain pen because I like the feel of it gliding along the paper. I don't like changing ink cartridges, but I tolerate this inconvenience.

Flubs, Snags

Why, in this age of computers, do I horse around with buggy whip technology? There are several reasons. First, I need to use a highly portable writing medium since I don't always know where I am going to be when I will want to write something. Second—and more important—the written page is not a page of pure text. It is a graphic document as well. Words can be underlined, crossed out, and added in the margins. Ideas for later parts of the document can be jotted down in the middle of a page and circled to show that they are part of something else.

As I am writing the document, I turn off all conscious judgment. Spelling errors, grammatical flubs, syntactic snags, all these go unnoticed at this stage. All I want to do is transcribe my thoughts. Period.

Once the document is captured on paper, I usually review it once, and make any large changes that come to mind. Next, I transcribe my writing into a computer system, either by entering it into my word processor directly, or by first entering it into a portable computer that accompanies me when I am on the road.

Both my portable (Radio Shack Model 100) and desktop (Apple Macintosh with *MacWrite*) computers have intuitively simple, virtually modeless, word processors. Because of what I do as I transcribe my handwritten text into the computer, it is important that the word processor be intuitively easy to operate.

In fact, I have a cardinal rule regarding word processor selection. I refuse to use a word processor whose manual is larger than the document I want to create. Since I am not in the process of compiling an encyclopedia, I have yet to work with *WordStar*.

The Product, Not The Tool

The task of transcription to the computer is one I undertake myself. I can't delegate it to a secretary because this transcription process is another chance to refine what I have written. By the time my document is in the computer, all that remains is to check the spelling and make grammatical corrections.

From creation to preparation of a final manuscript, I usually use several writing tools. I ask something different from each of them—but I am flexible. I sometimes capture my ideas with a keyboard instead of a pen, and the results are fine.

As with so many other areas of technology, we need to separate the tool from the product. The writing product can be produced in many ways, and it is the function of our tools to make these ways as easy to use as possible.

A Secret Process

In fact, there are many advantages to using a word processor over a typewriter. One of the beauties of using a word processor is that it lets you prepare a letter-perfect document. I find that my writing is better when the document looks nice. In this regard, the word processor helps me to be a better writer.

On the flip side of this argument, writers who use word processors tend not to retain copies of earlier drafts. There may be some scribbled notes and a final manuscript, with no documentary evidence of the process by which one became transformed into the other. Scholars who are interested in exploring the development of a book will have a harder time as today's Hemingways create intermediate drafts that are edited rather than rewritten.

But this shouldn't influence the quality of the author's writing—it only keeps the process a secret.

Maybe this makes writing a bit more magical!

Donald B. Trivette

Inside *King's Quest*

Byron and I were playing *King's Quest*—a new adventure game for the IBM PCjr written by Sierra. "But how does it work?" he wanted to know. "How would you even go about writing a program like that?" As a bright computer science major, Byron can write programs to perform reverse Polish notation, link-lists in Pascal, and all those other exotic things that students learn to do. But he couldn't begin to guess how *King's Quest* was written. Neither could I.

Everyone is familiar with arcade-style games in which you win points either for zapping strange-looking creatures or for not getting zapped yourself. An adventure game is entirely different. Winning an adventure game requires logic and puzzle-solving ability, not eye-hand coordination with a joystick.

Searching For Treasure

In *King's Quest* you play the game as Sir Grahame, a computer-animated knight who roams the kingdom of Daventry looking for three treasures. As Sir Grahame explores the kingdom, he can pick up objects like a dagger, a carrot, and a goat (yes! a goat) that may eventually help him locate the magical treasures. However, finding the objects is no cinch—some are in plain view but easily overlooked, while others are hidden in stumps and at the tops of trees. And of course there are hazards to overcome and puzzles to solve. How can Sir Grahame get across a bridge guarded by a troll? Kill the troll with the dagger? Don't try it! This is a game of strategy, *not* violence. There are several solutions to each puzzle and the more innovative and peaceful Sir Grahame is, the more points he gets.

The mechanics of playing the game are simple. You control Sir Grahame with either a joystick or the cursor-arrow keys. When Sir Grahame moves out of one scene, say to the right, a new scene appears on the screen. The main kingdom of Daventry is six scenes from north to south and eight scenes from east to west. The kingdom wraps around itself so that the 48 scenes are continuous. There are 32 other scenes for the interiors of caves and houses.

On the bottom four lines of the screen, you can type simple verb-noun sentences like *Take a carrot*, *Kill the troll*, or *Look at the river*. This area also displays messages and warnings for Sir Grahame: *The river is swift and deep.* (One quickly learns not to swim in swift, deep rivers.)

The graphics and animation in the PCjr version of *King's Quest* are spectacularly better than in any other adventure game I've seen. The three-dimensional quality makes it seem like Sir Grahame is moving through an animated cartoon. He can bump into and go around objects, climb trees and swim in water, duck behind rocks, and jump into the air. If he walks behind a rock, his legs are invisible; if he walks in front of a rock, part of the rock is invisible. When Sir Grahame moves, his arms and legs move, and the background shows between them. While we take that kind of animation for granted in a movie, it is not easily accomplished in a computer program on a machine without sprites. That's the part that had Byron and me puzzled. *How do they do that?* I called Sierra to find out.

A $700,000 Computer Game

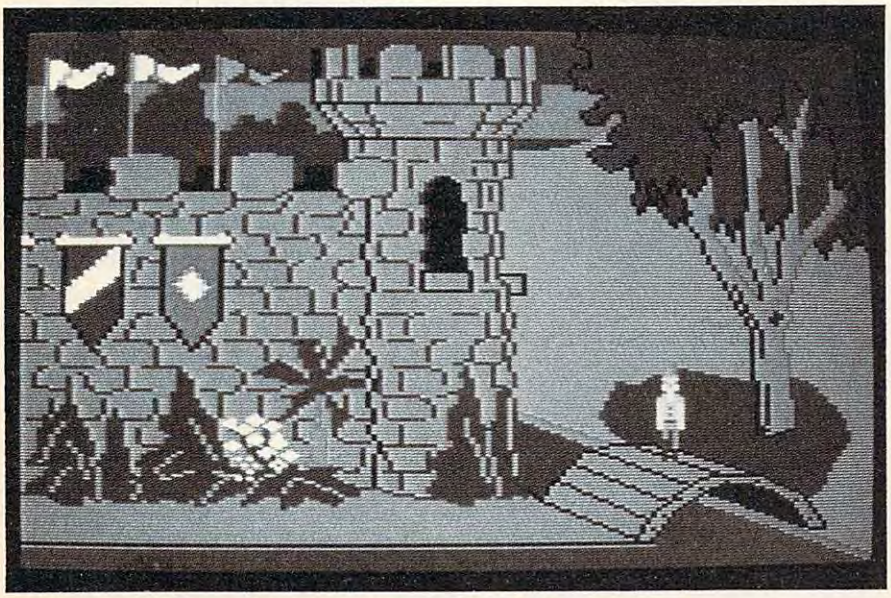
A year before the PCjr was announced—when the "Peanut" was just a rumor to the rest of us—IBM asked Sierra to create a game that would show off the new computer's color graphics capabilities. IBM supplied Sierra with a prototype Junior.

Roberta Williams, who had worked on five other adventure games, was given the task of designing something completely new and different. Eighteen months later, Williams and a team of six programmers and artists had created *King's Quest*—at a cost of over $700,000.

First, Williams wrote the story. She based it not on strange characters with strange names, but rather on familiar characters from literature.

Do you remember "Billy Goats Gruff" and "Hansel and Gretel"? Once the story line was established, the artists prepared detailed color drawings of each of the 80 scenes. In adventure-game jargon, scenes are called *rooms*. Each drawing was then traced on a Calcomp Graphics Tablet. This process automatically generated the instructions which tell the computer how to reproduce each room, saving the programmers months of tedious work.

The drawing instructions for each room are stored as separate files on the disk. (Technically, they are stored at absolute sector addresses, not actually in disk files.) When Sir Grahame moves from one room to another, the computer loads the instructions for displaying the new room, draws the room point by point, and then fills in the color. It takes about four seconds for the PCjr to draw and color a room. A faster approach would have been to store the room *images* themselves on disk, already drawn. But that method would have used considerably more disk space and reduced the number of rooms in the game. (Actually, it's entertaining to watch the computer draw and color each scene.)



Sir Grahame begins his quest at the castle of Daventry.

The Invisible Skeleton

The first scene in *King's Quest* is the castle of Daventry where Sir Grahame's quest begins (see photo). The lions, flags, stone blocks, alligators, and plants make this the most detailed room in the kingdom. It takes 2400 bytes of instructions to tell the PCjr how to draw and color this scene. By contrast, the easiest room to draw requires only 470 bytes of instructions.

The scenes are more than just static backgrounds. For instance, if Sir Grahame staggers a bit and runs into a castle wall, he stops. How does the program know when the character hits something? There's a skeleton, an invisible structure, behind each picture. If you could see this skeleton, you'd notice lots of lines running all over the screen defining where Sir Grahame can and cannot walk. The lines were drawn into each room with the graphics tablet.

In addition to concealing hidden lines, each room also assigns priorities to every object it contains (trees, rocks, flowers, etc.). These priorities—numbers from 1 to 15—give *King's Quest* its three-dimensional quality.

Objects at the top of the screen have low priority; those at the bottom, high. Sir Grahame's priority changes as he moves around the room. For example, a tree in the middle of the screen might have a priority of 9. When Sir Grahame is in front of the tree—closer to the bottom of the screen—his priority might be 11. As he moves up the screen, his priority changes. If he is behind the tree in a scene, his priority is less than that of the tree. By comparing priorities each time Sir Grahame moves, the program makes decisions about how to draw the screen. If Sir Grahame's priority is higher than the object behind him, he is visible and the object (or part of it) is invisible. If his priority is lower, as when he steps *behind* a tree, then he (or part of him) disappears.

Each time Sir Grahame (or any object) moves, the surrounding region on the screen is saved in the computer's memory in one of four *save-areas*. The program checks the new location for skeleton lines and priorities, then adjusts Sir Grahame and the surrounding area for any changes—perhaps part of a rock became visible in front of his legs. Finally, Sir Grahame and the surrounding area are redrawn on the screen. And that's how Sir Grahame walks around the kingdom of Daventry.

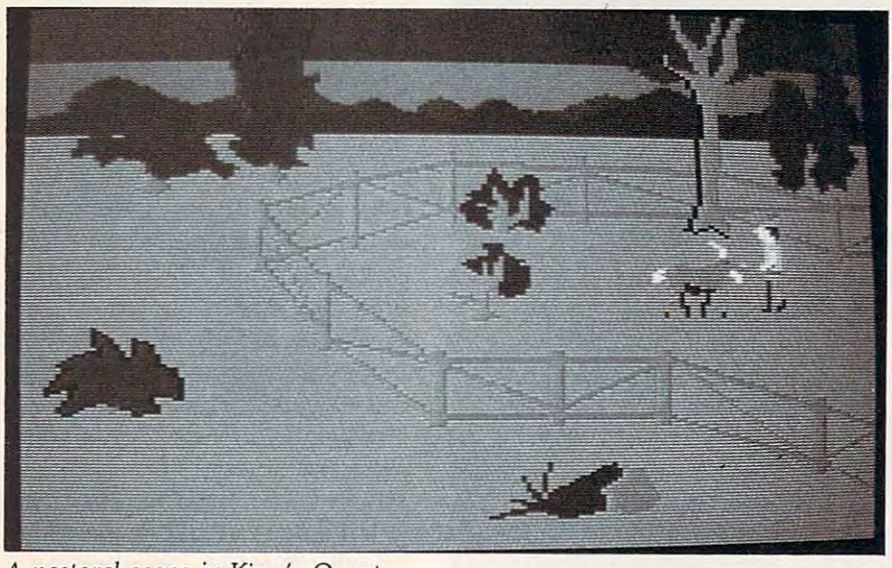
The Game's Own Language

In addition to the graphics for each room, there is a set of logical statements. These are written in a special language devised by Sierra called the Game Adaptation Language. The program constantly loops through these statements looking for something to change. They work sort of like a group of IF-THEN statements in BASIC.

For example, in one room, room 10, a goat randomly wanders around inside a pen (see photo). The pen extends into room number 11 on the right. If the goat happens to wander out of room 10, the program must erase the goat. The program knows the goat by the codename *14* and Sir Grahame by the name *Ego*. So if Ego moves to room 11 in search of the goat, the program must remember to draw 14 in room 11. The statement in Game Adaptation Language looks like this:

IF HAS-GOAT 0 AND OBJHIT-EDGE 14 AND EDGEOBJ-HIT 1 AND GOAT-GONE 0 AND SHOW-CARROT 0 THEN ASSIGN GOAT-ROOM 11, ERASE 14.

If I understood what that meant, I'd be writing adventure games instead of magazine articles, but with a little examination we can pretty much figure out what's going on. In programming logic, the numeral 0 means *false, no,* or *off,* and the numeral 1 means *true, yes,* or *on.* Thus, in English, the statement might read: "If Ego doesn't have the goat and if the goat has hit the edge of the room and if the edge of the room has been hit and if the goat is in the room in the first place and if the goat has not been shown the carrot, then [Whew!] remember to draw the goat in room 11 and erase the goat from room 10."



A pastoral scene in King's Quest.

And remember, that is just one of the logic statements that goes with room 10—there are a total of 180 logic lines for this room alone. The logic statements give the program its personality; they tell the program what to do and when to do it.

Inevitable Bugs

Anyone who has ever written a computer program knows that program logic is a fertile field for errors. If the programmer forgets to tell the computer just one little detail, the results can be amusing—and disastrous. Preliminary versions of *King's Quest* were not without bugs. It took several weeks to find out why, if Sir Grahame jumped in the air as he moved from one room to another, he skidded into the new room totally out of control. That bug has been fixed, but a few others still lurk in the current PCjr version. (But not, says Sierra, in the versions it markets directly.)

During one game, Sir Grahame was standing at the edge of a bridge. The goat was coming back across the bridge, and as it passed, I made Sir Grahame jump. A foolish thing to do, I know, but I was celebrating. Unfortunately, the screen filled with horizontal lines and the game came to a swift end. The programmer who wrote the logic statements for that scene forgot to allow for the possibility that Ego might jump as the 14 passes in front of him. Nevertheless, the errors are very few.

Some things you might consider an error are not errors at all. For example, there can be only four animated objects (including Sir Grahame) in any one room. The program has only four *save-areas* for animated objects. Somewhere along the way (I won't tell you where), Sir Grahame can acquire some magic beans. Getting those beans is the toughest part of *King's Quest.* (It helps to have read *Grimms' Fairy Tales.*) But try to plant the beans in the room where you found them and the screen will say: "You can't do that here." That's because *that* room already has four animated objects. In fact, you can plant the beans in only about half of the rooms in *King's Quest* because of the four-object-maximum rule. That's not a bug or an error, it's just a programming tradeoff to conserve memory.

There *is* one room in *King's Quest* with five objects. I don't want to give anything away, but: Should Sir Grahame (1) give the Woodcutter (2) and his wife (3) the right object (4), then he can take the fiddle (5). This apparent violation of the rule is permitted because the programmers pulled a fast one. Look closely when Sir Grahame puts *the something* on the table and you'll see the Woodcutter disappear for a moment. The programmer briefly swaps objects to keep within the limits. Tricky.

A Sequel On The Way

Roberta Williams is a perfectionist. There just wasn't time or memory to put everything in *King's Quest* that she wanted. She wishes the language interpreter had a larger vocabulary, that Sir Grahame could drop objects he has picked up, and that he could be even more animated. She wishes some of the characters, like the wolf, could roam from room to room. But she says the sequel, due in February or March, will be even better.

In *King's Quest II*, Sir Grahame—who becomes King Grahame when you solve *King's Quest*—goes in search of a wife. Along the way he meets Dracula and King Neptune, and rides a flying carpet. And, somehow, the folks at Sierra found a way to squeeze 94 rooms onto the disk. I can hardly wait.

IBM markets the PCjr version of *King's Quest* and Sierra markets versions for the IBM PC, Apple IIc, Apple IIe, and Tandy 1000 (a new computer scheduled for release in January 1985). All versions cost $49 and require 128K of memory and a disk drive. In addition, the PC version runs on most IBM compatibles. (When the PC version is displayed on an RGB monitor, the graphics are in the standard four-color medium-resolution mode; but connect your PC to a television and you'll get the same spectacular colors as the PCjr version.) ©

INSIGHT: Atari

Bill Wilkinson

I am much gratified by the response to my decision to work harder on answering readers' questions. I have received several *very* interesting letters with both good comments and good questions. Since it is always fun to defend Atari BASIC against the outside world, let me start with a subject near and dear to my heart.

Benchmarks

Several readers have asked me why Atari BASIC compares so unfavorably to other computers on certain benchmarks. The two most commonly mentioned are the *BYTE* magazine benchmarks and the *Creative Computing* benchmark invented by David Ahl. Stan Smith, of Los Angeles, asked some very pointed questions, which I will try to answer here.

The *BYTE* benchmark is reproduced below in Atari BASIC. It is the often-mentioned "Sieve of Erastothenes," a program which produces (and counts) prime numbers. Its primary advantage as a benchmark is that it can be implemented in virtually any language (although only with much difficulty when using Logo and its ilk). It relies only on addition and logical choices, with very little number crunching.

```
10 DIM N$(8192)
20 N$="0":N$(8192)="0":N$(2,819
   2)=N$
30 FOR I=1 TO 8192:IF N$(I,I)="
   1" THEN 60
40 PRIME=I+I+1:CNT=CNT+1:K=I
50 K=K+PRIME:IF K<8193 THEN N$(
   K,K)="1":GOTO 50
60 NEXT I
70 PRINT CNT : REM BETTER PRINT
    1899!!!
```

An aside: If you have seen the *BYTE* original and are puzzled by my changes, be aware of three things: (1) I had to use a string because there is not enough room for an array of 8192 elements. (2) The math was modified very slightly to accommodate the fact that string indices start at one, instead of zero. (3) Multiple statements per line simplify the original somewhat.

Anyway, why is Atari BASIC so slow (317 seconds versus, for example, the IBM PC at 194 seconds)? Primarily for three reasons. First, note

all the numbers in this listing, which must be treated as integers. Line numbers and indices are always kept and calculated as floating-point numbers, but all must be converted to integers before being used. (You simply can't GOTO line 137.38, can you?) And, sigh, the routine in the Atari Operating System ROMs which converts numbers to integers is incredibly slow (in fact, it is the only floating-point routine we modified when we produced BASIC A+ and BASIC XL).

Second, Atari BASIC performs FOR-NEXT loops by remembering the line number of the FOR statement. Then, when NEXT is encountered, BASIC must search for the FOR line, just as if a GOTO had been used. (Other BASICs remember the actual memory address of the FOR statement. Faster, but less flexible. Atari BASIC allows you to STOP in the middle of a loop, change the program, *and continue*, something no other home computer BASIC allows. (This—among many other things—is in direct opposition to *Consumer Reports'* claim that Atari BASIC is hard for beginners.)

Third, if you type in and use this listing as shown, you are paying almost a 50 percent penalty in speed, thanks to Atari's screen DMA and Vertical Blank Interrupts taking up a significant portion of the processing time. The simple addition of the following two lines will improve the time for this little test to 211 seconds:

```
5  POKE 54286,0 : POKE 54272,0
65 POKE 54286,64
```

All of a sudden, Atari BASIC isn't even *near* the bottom of the list. And, yet, there is more we can do to improve the machine's performance. As many have suggested, you can install the Newell Fastchip, a replacement for the floating-point routines built into your computer (available from many dealers, produced by Newell Industries of Plano, Texas).

Or you can change to another BASIC. Obviously, there is Atari's Microsoft BASIC. It produces results very close to those of Applesoft; but it, too, can be improved by turning off screen DMA, etc. And there is OSS's own BASIC XL. Using a combination of clever programming and a Fastchip, the BASIC XL program below will count up all those prime numbers in 58.5 seconds, about three times as fast as Microsoft BASIC on an IBM PC can do it. 'Nuff said. (Except a P.S.: The Set 3 in line 10 requests zero-time FOR loops, something not available in many BASICs, which alone accounts for about 20 seconds worth of improvement.)

```
10 FAST: POKE 54286,0: POKE 542
   72,0: SET 3,1: DIM N$(8192):
    N=ADR(N$)
30 FOR I=0 TO 8191
50 IF NOT PEEK(N+I) THEN PRIME=
   I+I+3: CNT=CNT+1: FOR K=I+PR
   IME TO 8191 STEP PRIME: POKE
    N+K,1: NEXT K
60 NEXT I
70 POKE 54286,64: POKE 559,34:
   PRINT CNT
```

Measures Of Accuracy

The Ahl benchmark is listed below. It purports to measure both accuracy and number-crunching ability. It does neither very well. Still, we have to ask why Atari BASIC is near dead last in its rankings, requiring 6 minutes and 45 seconds to complete the test.

```
10 FOR N=1 TO 100: A=N
20 FOR I=1 TO 10: A=SQR(A): R=R
   +RND(0): NEXT I
30 FOR I=1 TO 10: A=A^2: R=R+RN
   D(0): NEXT I
40 S=S+A: NEXT N
50 PRINT "ACCURACY="; ABS(1010-
   S/5), "RANDOM="; ABS(1000-R)
```

The culprit here (in terms of time-wasting) is line 30, with its A=A^2. Atari BASIC, in common with most small computer BASICs, calculates powers according to a formula:

$$x^y = \exp(y * \log(x))$$

where log() is the natural logarithm function and exp() is the exponent-of-e function.

If you don't understand that, don't worry about it. The point is that the calculation of such a simple thing as a number to the second power involves the calculation of a logarithm and an exponentiation. And why is that so bad? Simply because the floating-point routines in the Atari OS ROMs are too slow. Again, the solution is to install the Newell Fastchip and/or turn off DMA and VBI (as outlined above).

I am indebted to Clyde Spencer, one of the founders of the Bay Area Atari Users Group (one of the oldest), for supplying me with a most surprising figure. Spencer reports that, using the Fastchip and with DMA turned off, he obtained a timing of 1 minute 38 seconds, a very respectable (albeit not record-shattering) performance. I still wouldn't use my Atari for advanced scientific applications, but it is more than adequate for most purposes.

There is a problem with the "accuracy" figures in this test, however. First, because Ahl's accuracy number is the result of 1000 simple sums, it is clearly possible that a particular machine may exhibit wildly variant results for various numbers and still show a good figure in his test. (To illustrate, assume that the SQR() function randomly tosses in an error of plus or minus

one. If it tossed in an equal number of errors, they would balance to zero. Yet choosing to make the loop just one unit shorter [FOR N=1 TO 999] might give a completely different result. To be fair, this is a very unlikely result with modern math algorithms; but, still, one never knows.) A minor change to his program would improve the testing qualities considerably:

```
40 S = S + ABS( A-N ) : NEXT N
```

Do you see the difference? This method produces the sum of the errors, and doesn't fall prey to offsetting errors.

The Random Number Trap

There is no hope for the accuracy of this random number tester, though. I will quote Clyde Spencer on this matter: "If the numbers are *truly* random and *not* normally distributed, *any* difference between 0 and 1000 is possible. All you can say is that you would have a high *probability* of . . . being near zero for a perfect random number generator." The benchmark test falls into the infamous BASIC repeating-random-sequence trap.

In most BASICs, when you command a program to run, the pseudorandom generator is *always* reseeded with the *same* number. So each and every time you will get the same results, with Ahl's test. And, depending on what seed is chosen, you may get truly phenomenal results (because you happened to hit a hot spot in the generator's sequence). Now, though, try starting the generator off with a different (and randomly chosen) seed each time. What happens? The test's randomness figure wanders all over the place.

Once again, to quote Spencer, " . . . in eight tests I obtained numbers ranging from 1.6 to 24.2, with the mean being 7.02 . . . . "

Finally, I would like to point out that Ahl's test penalizes small machine BASIC interpreters in yet another way: When you have 32K bytes to spend on a BASIC, one thing you do is insure that numbers to a power are performed by successive multiplications, if possible. Thus Cromemco 32K Structured BASIC (for example) performs A^2 with just one multiply. In other words, it converts A^2 to A*A. If you manually substitute that same form in Ahl's program, the times for almost all of the smaller and less expensive machines will improve dramatically. (Surprisingly, though, the accuracy figures may not change. After all, the original version may have had offsetting errors.) Of course, if you need to use noninteger powers in your programs, this comment doesn't apply, and the benchmark's results are a bit more meaningful for you.

Well, what does all this long-winded discussion boil down to? Two simple points: (1) Always presume that a benchmark program is worth slightly less than the paper it is printed on. (2) If you want to do number crunching on your Atari computer (against my best advice), go out and buy the Newell Fastchip. (And it won't hurt to try some other languages.)

HELP? HELP!

Besides noting that GRAPHICS 15 on the XL machines is easily accessible (it's equivalent to mode 7½ on older machines), Mark Butler, of Antioch, California, asked for some information about the HELP key.

Simply put, pushing the HELP key on an XL machine causes an interrupt (I'm not sure which one) that, in turn, causes the Operating System to set a HELP flag. The magic location is $2DC, 732 decimal. Pushing HELP, either alone or in combination with CONTROL or SHIFT, forces the OS to put a value here, as shown below:

Key(s) Pressed	Value in $2DC (732)
HELP alone	$11 (17 decimal)
CONTROL+HELP	$91 (145)
SHIFT+HELP	$41 (65)

To use $2DC, you must POKE it back to zero after you have decided that someone needs HELP which you are going to act on.

Butler also requested a program which would, for example, print out an error message for the last BASIC error number when the HELP key is pressed. While not a *really* difficult project, it is a bit too heavy for this column. On the other hand, it would be trivial to add a HELP capability to many BASIC programs. Why not try it?

As long as we are on this subject, I would like to also note the effects of the 1200XL's function keys on another memory location, $2F2 (754 decimal). The various possible values are listed below. Note that CONTROL used with a function key is not generally accessible after keyboard input, since these combinations have special meanings to the OS and the editor handler. We will thus ignore them here.

Key(s) Pressed	Value in $2F2 (754)
F1 alone	$03 (3 decimal)
SHIFT+F1	$43 (67)
F2 alone	$04 (4)
SHIFT+F2	$44 (68)
F3 alone	$13 (19)
SHIFT+F3	$53 (83)
F4 alone	$14 (20)
SHIFT+F4	$54 (84)

Too bad all machines don't have function keys, isn't it?

Cassettes And The XL Machines

Guy Servais, of Norfolk, Virginia, was one of several who I inadvertently ignored when I discussed holding down the OPTION key while

booting an 800XL computer. My apologies for slighting you cassette owners.

Still, my general comments apply: If you purchase a cassette program which includes instructions telling you to remove your BASIC cartridge, you *must* hold down the OPTION key while booting that cassette. The kicker here, though, is that you must *also* hold down the START button to force the boot in the first place. Under the conditions mentioned, I recommend holding down *both* buttons until you actually hear the tone on the tape being accepted by the computer.

Servais also asked me if you can "disable the built-in BASIC . . . and can type in programs written in machine language." I can only presume that he has either seen or used other brands of computers which have some sort of minimonitor which allows you to access the bits and bytes of memory. (For example, Apple II computers have a small monitor which you can get to.)

Sorry, Guy, but there ain't no such thing on an Atari computer. You have three choices:

1. Use BASIC. This isn't quite as bad as it sounds. Look at the MLX machine language loader which COMPUTE! uses. It is a good tool for entering machine language written by others.

2. Buy a cartridge-based assembler. The old Atari *Assembler Editor* cartridge is often available at a substantial discount. It's not great, but it's much better than the simple monitors on other machines.

3. Buy a disk drive. This will open up a whole new vista in machine language. There are several appropriate assemblers for disk users.

Even though I have said this before, it bears repeating: The *first* peripheral you should buy is a disk drive. Only use cassette if you are desperate, and never waste your money on a printer until you have a disk.

Can You Help?

Servais mentioned one more thing in his letter which disturbed me. He is experiencing the infamous Atari BASIC editing lockup in his 800XL with the built-in BASIC. I had believed that the 800XL's BASIC had cured that problem (though it left a few other bugs lying around). Now, truthfully, I haven't used much besides BASIC XL in the last year, so I have not been aware of this problem at all.

Has anyone documented the circumstances under which lockup occurs? Please write and tell us. Once again, since BASIC is in ROM, I doubt there is a fix for the problem. But if we are aware of why and how it occurs, we may be able to warn others away from those conditions. ©

TELECOMPUTING TODAY

Arlan R. Levitan

I work full-time in the IBM support district of a large telecommunications utility. About two weeks ago I was informed that my request to attend a one-week conference sponsored by IBM in San Francisco had been approved. Now, I don't particularly care for flying, especially a five-hour flight to the West Coast. Besides, the corporate travel bureau we use always has had a habit of routing me to my destination with three-hour stopovers in backwater airports and plane changes that would give Colonel Chuck Yeager fits.

I decided to try a feature offered by most of the commercial information services, an electronic edition of the Official Airline Guide (OAG). A few words of explanation are in order for those of you unfamiliar with the OAG. The regular paper edition of the guide is printed once every two weeks and contains fare and schedule listings for all of the commercial airlines in the United States. It is theoretically available to regular travelers at little or no charge. Actually, getting a recent copy from an airline or travel agent usually requires giving up a complete set of fingerprints and your first-born child.

The electronic edition of OAG, while not free, does have some advantages. Its on-line information is always current and, most important, it offers extensive search capabilities. Using the electronic OAG, it's a snap to find the lowest possible fares available, even ones that many travel agents can miss. Sure, I could have called around to every airline in town and got the same information for free. But "free" in this case means spending about an hour making ten or so phone calls, plus being subjected to canned Muzak while waiting for a reservations operator.

Looking up the flights via the electronic OAG took around 90 seconds and cost about $1.50. Besides, it was a lot of fun to call the airline I had settled on and have all of the flight information before the fact. Electronic OAG will really come into its own when the information services also offer on-line ticket bookings for all airlines. Don't be surprised if such services are commonly available by the end of the year.

Telecomputing On The Run

Two days before I was to leave, I got a call from my editor at COMPUTE!.

"Arlan, this is Tom. Where's the February column?"

"Are you kidding? I just sent January two weeks ago, and I'll never live down the fact that it was actually on time. Besides, I'm going to be out of town for a week. There's no way I can get it to you by next week." I smiled, thinking of leisurely strolling along Fisherman's Wharf in the cool of a San Francisco evening.

"Hey, didn't you just buy a lap computer with a built-in modem? You can take it with you, write the column on it and then transmit it directly into our computer via phone, right?"

I silently cursed myself for ever mentioning my new acquisition in passing conversation.

Actually, taking a modem on the run turned out to be a pretty good idea. The Sunday I arrived, it was raining heavily, and I was too jet-lagged to want to go anywhere. Having a portable computer with a modem saved me from having to endure *Knight Rider*. From the comfort of my hotel room, I logged onto my favorite commercial information services, chatted with some of my electronic compadres, and perused items of interest on the various forums I participate in.

Monday night, I left a message on a local computerized bulletin board system (BBS) asking for restaurant recommendations from the locals, rather than trust the "Dine At Our Advertisers" booklets that litter hotels.

Wednesday, I received a call from my place of work in Michigan. There was a minor problem with one of the computer subsystems I was responsible for. I could have spent over an hour on the phone describing how to deal with the problem in detail. Instead, I dialed into the system with my lap computer, and analyzed and fixed the problem in about ten minutes. Also on Wednesday, I dialed back into the local BBS and read the response to my restaurant inquiry. One

of the recommendations looked particularly enticing, and that evening my friends and I had a great Szechwan dinner at a place that wasn't listed in any of the where-to-eat booklets.

And on Thursday night, after an exhausting day of meetings, I wrote this column in my hotel room when I could have been wasting my time touring the city in a cable car.

But seriously, do I regret telecomputing on the run? Absolutely not. I doubt if I'll travel on business without telecomputing power again.

Watch for more under-$500 consumer-oriented lap portables with built-in modems in 1985. It's been found that integrated telecommunications is crucial to the success of lap computers. Consider two nearly identical lap machines, the Tandy TRS-80 Model 100 and the NEC 8201A. Both are manufactured by the Japanese firm Kyocera. The NEC has a clearly superior keyboard and more memory capacity than the Model 100. So why does the Model 100 outsell it by more than 20 to 1? The Tandy has a built-in modem, and as the sales figures show, that makes all the difference.

Even Commodore is said to be considering adding a modem to its portable SX-64 in an effort to spur sales. Indeed, it may soon be difficult to buy a microcomputer without a modem. The current availability of a $10, 300 bps modem-on-a-chip will have a profound effect on the telecomputing user base in the next few years. Because built-in modems add lots of functionality for little additional cost, you can expect the majority of new machines introduced in 1986 and beyond to sport integrated telecommunications.

Info-Wars

The commercial information service wars are heating up again. Recently, CompuServe has been gaining ground in the corporate computing community with its business-tailored Executive Information Service. This has the No. 1 business info provider—Dow Jones News/Retrieval—scrambling in response. DJNR has cut its rates by 25 percent across the board for general services and regular stock quotes, and has announced a major new service—stock quotes based upon "last trade."

What are last trade quotes? Until the announcement of this service, all of the stock quotes offered by the major consumer information utilities have been delayed 15 to 20 minutes. Anyone familiar with the stock market knows that 15 minutes can be a lifetime in the price of an individual issue. Last trade quotes report the most recent price paid for a stock, based on the last transaction logged by the exchange on which the stock is listed.

Transactions are reported much more quickly under the new system. The time varies from exchange to exchange, but 20 seconds or less is not uncommon. Moderately serious investors who don't have a broker willing to spend an hour at a time on the phone with them will find the new service a real blessing.

The last trade quotes will cost DJNR users a $12 monthly service charge in addition to the normal connect-time charges for quotes. The monthly surcharge covers the fees paid by Dow Jones to the exchanges for the more timely quote information.

DJNR didn't have much time to gloat before The Source—a rival information service owned by Reader's Digest—struck back in spades, announcing its own last trade quotes. Although The Source's $20 monthly surcharge is a bit higher than DJNR's, The Source introduced a powerful adjunct service. By special arrangement with Spears Incorporated, a discount brokerage house, stock watchers on The Source can set up an account with Spears and issue trading orders *on-line*. According to Spears, most orders will be executed within two minutes of issuance. An extensive portfolio tracking system is also available on-line and may be used to value actual holdings or to track "paper portfolios" (for those who wish to dabble for fun rather than real money).

New On-Line Services

The information service war is spreading beyond the business sector as well. Two new on-line services—People/Link and Play/Net—are starting up with introductory connect-time charges significantly below those charged by the established leaders.

People/Link will offer services similar to CompuServe's popular Nationwide CB and Special Interest Forums as well as electronic mail at only $2.95 an hour. Play/Net sounds even more aggressive, offering on-line games with medium-resolution color graphics to users who purchase its proprietary terminal program. Although the Commodore 64 is the only machine Play/Net supports at this time, the service claims it will support IBM and Apple computers by this spring. Play/Net's introductory connect time charges? An unbelievable $2.00 an hour! Both Play/Net's and People/Link's hourly charges will certainly rise after the two firms have lured enough users to make continued operation of the services viable, but such predatory pricing should help keep the rates charged by the big boys down to earth.

Arlan R. Levitan
Source: TCT987

COMPUTE!'s Guide To Typing In Programs

Before typing in any program, you should familiarize yourself with your computer. Learn how to use the keyboard to type in and correct BASIC programs. Read your manuals to understand how to save and load BASIC programs to and from your disk drive or cassette unit. Computers are precise—take special care to type the program *exactly* as listed, including any necessary punctuation and symbols. To help you with this task, we have implemented a special listing convention as well as a program to help check your typing—the "Automatic Proofreader." Please read the following notes before typing in any programs from COMPUTE!. They can save you a lot of time and trouble.

Since programs can contain some hard-to-read (and hard-to-type) special characters, we have developed a listing system that spells out in abbreviated form the function of these control characters. You will find these special characters within curly braces. For example, {CLEAR} or {CLR} instructs you to insert the symbol which clears the screen on the Atari or Commodore machines. A symbol by itself within curly braces is usually a control key or graphics key. If you see {A}, hold down the CONTROL key and press A. Commodore machines have a special control key labeled with the Commodore logo. Graphics characters entered with the Commodore logo key are enclosed in a new kind of special bracket. A graphics character can be listed as [<A>]. In this case, hold down the Commodore logo key as you type A. Our Commodore listings are in uppercase, so shifted symbols are underlined. A graphics heart symbol (SHIFT-S) would be listed as S. One exception is {SHIFT-SPACE}. Hold down SHIFT and press the space bar.

If a number precedes a symbol, such as {5 RIGHT}, {6 S}, or [<8 Q>], you would enter five cursor rights, six shifted S's, or eight Commodore-Q's. On the Atari, inverse characters (printed in white on black) should be entered with the Atari logo key. Since spacing is sometimes important, any more than two spaces will be listed, for example, as: {6 SPACES}. A space is never left at the end of a line, but will be moved to the next printed line as {SPACE}. There are no special control characters found in our IBM PC/PCjr, TI-99/4A, and Apple program listings. For your convenience, we have prepared this quick-reference key for the Commodore and Atari special characters:

Atari 400/800/XL

When you see	Type	See	
{CLEAR}	ESC SHIFT <		Clear Screen
{UP}	ESC CTRL -	↑	Cursor Up
{DOWN}	ESC CTRL =	↓	Cursor Down
{LEFT}	ESC CTRL +	←	Cursor Left
{RIGHT}	ESC CTRL *	→	Cursor Right
{BACK S}	ESC DELETE		Backspace
{DELETE}	ESC CTRL DELETE		Delete character
{INSERT}	ESC CTRL INSERT		Insert character
{DEL LINE}	ESC SHIFT DELETE		Delete line
{INS LINE}	ESC SHIFT INSERT		Insert line
{TAB}	ESC TAB		TAB key
{CLR TAB}	ESC CTRL TAB		Clear tab
{SET TAB}	ESC SHIFT TAB		Set tab stop
{BELL}	ESC CTRL 2		Ring buzzer
{ESC}	ESC ESC		ESCape key

Commodore PET/CBM/VIC/64

When You Read:	Press:	See:	When You Read:	Press:	See:
{CLR}	SHIFT CLR/HOME		{GRN}	CTRL 6	
{HOME}	CLR/HOME		{BLU}	CTRL 7	
{UP}	SHIFT CRSR ↑		{YEL}	CTRL 8	
{DOWN}	CRSR ↓		{F1}	f1	
{LEFT}	SHIFT CRSR →		{F2}	f2	
{RIGHT}	CRSR →		{F3}	f3	
{RVS}	CTRL 9		{F4}	f4	
{OFF}	CTRL 0		{F5}	f5	
{BLK}	CTRL 1		{F6}	f6	
{WHT}	CTRL 2		{F7}	f7	
{RED}	CTRL 3		{F8}	f8	
{CYN}	CTRL 4		←	←	
{PUR}	CTRL 5		↑	SHIFT ↑	

The Automatic Proofreader

Also, we have developed a simple, yet effective program that can help check your typing. Type in the appropriate Proofreader program for your machine, then save it for future use. On the VIC, 64, or Atari, run the Proofreader to activate it, then enter NEW to erase the BASIC loader (the Proofreader will still be active, hidden in memory, as a machine language program). Pressing RUN/STOP–RESTORE or SYSTEM RESET deactivates the Proofreader. You can use SYS 886 to reactivate the VIC/64 Proofreader, or PRINT USR(1536) to reenable the Atari Proofreader. The IBM Proofreader is a BASIC program that lets you enter, edit, list, save, and load programs that you type. It simulates the IBM's BASIC line editor.

Using The Automatic Proofreader

Once the Proofreader is active, try typing in a line. As soon as you press RETURN, either a number (on the Commodore) or a pair of letters

(Atari or IBM) appears. The number or pair of letters is called a *checksum*. Try making a change in the line, and notice how the checksum changes.

All you need to do is compare the value provided by the Proofreader with the checksum printed in the program listing in the magazine. In Commodore listings, the checksum is a number from 0 to 255. It is set off from the rest of the line with *rem*. This prevents a syntax error if the checksum is typed in, but the REM statements and checksums need *not* be typed in. It is just there for your information.

In Atari and IBM listings, the checksum is given to the left of each line number. Just type in the program, a line at a time (without the printed checksum) and compare the checksum generated by the Proofreader to the checksum in the listing. If they match, go on to the next line. If not, check your typing: You've made a mistake. On the Commodore and Atari Proofreader, spaces are not counted as part of the checksum, and no check is made to see that you've typed in the characters in the right order. If characters are transposed, the checksum will still match the listing. Because of the checksum method used, do not use abbreviations, such as ? for PRINT. However, the Proofreader does catch the majority of typing errors most people make. The IBM Proofreader is even pickier; it *will* detect errors in spacing and transposition. Also, be sure you leave Caps Lock on, except when you need to enter lowercase characters.

Special Proofreader Notes For Commodore Cassette Users

The Proofreader resides in the cassette buffer, which is used during tape LOADs and SAVEs. Be sure to press RUN/STOP–RESTORE before you save or load a program, to get the Proofreader out of the way. If you want to use the Proofreader with tape, run the Proofreader, then enter these two lines *exactly* as shown, pressing RETURN after each one:

```
A$="PROOFREADER.T":B$="{10 SPACES}"
    :FORX=1TO4:A$=A$+B$:NEXT
FORX=886TO1018:A$=A$+CHR$(PEEK(X))
    :NEXT:OPEN 1,1,1,A$:CLOSE1
```

Then press RECORD and PLAY on a blank tape, and a special version of the Proofreader will be saved to tape. Anytime you need to reload the Proofreader after it has been erased, just rewind the tape, type OPEN1:CLOSE1, then press PLAY. When READY comes back, enter SYS 886.

IBM Proofreader Commands

Since the IBM Proofreader replaces the computer's normal BASIC line editor, it has to include many of the direct-mode IBM BASIC commands. The syntax is identical to IBM BASIC. Commands simulated are LIST, LLIST, NEW, FILES, SAVE, and LOAD. When listing your program, press any key (except Ctrl-Break) to stop the listing. If you enter NEW, the Proofreader will prompt you to press Y to be especially sure you mean yes.

Two new commands are BASIC and CHECK. BASIC exits the Proofreader back to IBM BASIC, leaving the Proofreader in memory. CHECK works just like LIST, but shows the checksums along with the listing. After you have typed in a program, save it to disk. Then exit the Proofreader with the BASIC command, and load the program into the normal BASIC environment (this will replace the Proofreader in memory). You can now run the program, but you may want to resave it to disk. This will shorten it on disk and make it load faster, but it can no longer be edited with the Proofreader. If you want to convert a program to Proofreader format, save it to disk with SAVE "filename",A.

VIC/64 Proofreader

```
100 PRINT"{CLR}PLEASE WAIT...":FORI=886TO10
    18:READA:CK=CK+A:POKEI,A:NEXT
110 IF CK<>17539 THEN PRINT"{DOWN}YOU MADE
    {SPACE}AN ERROR":PRINT"IN DATA STATEMEN
    TS.":END
120 SYS886:PRINT"{CLR}{2 DOWN}PROOFREADER A
    CTIVATED.":NEW
886 DATA 173,036,003,201,150,208
892 DATA 001,096,141,151,003,173
898 DATA 037,003,141,152,003,169
904 DATA 150,141,036,003,169,003
910 DATA 141,037,003,169,000,133
916 DATA 254,096,032,087,241,133
922 DATA 251,134,252,132,253,008
928 DATA 201,013,240,017,201,032
934 DATA 240,005,024,101,254,133
940 DATA 254,165,251,166,252,164
946 DATA 253,040,096,169,013,032
952 DATA 210,255,165,214,141,251
958 DATA 003,206,251,003,169,000
964 DATA 133,216,169,019,032,210
970 DATA 255,169,018,032,210,255
976 DATA 169,058,032,210,255,166
982 DATA 254,169,000,133,254,172
988 DATA 151,003,192,087,208,006
994 DATA 032,205,189,076,235,003
1000 DATA 032,205,221,169,032,032
1006 DATA 210,255,032,210,255,173
1012 DATA 251,003,133,214,076,173
1018 DATA 003
```

Atari Proofreader

```
100 GRAPHICS 0
110 FOR I=1536 TO 1700:READ A:POKE I
    ,A:CK=CK+A:NEXT I
120 IF CK<>19072 THEN ? "Error in DA
    TA Statements.  Check Typing.":E
    ND
130 A=USR(1536)
140 ? :? "Automatic Proofreader Now
    Activated."
```

```
150 END
1536 DATA 104,160,0,185,26,3
1542 DATA 201,69,240,7,200,200
1548 DATA 192,34,208,243,96,200
1554 DATA 169,74,153,26,3,200
1560 DATA 169,6,153,26,3,162
1566 DATA 0,189,0,228,157,74
1572 DATA 6,232,224,16,208,245
1578 DATA 169,93,141,78,6,169
1584 DATA 6,141,79,6,24,173
1590 DATA 4,228,105,1,141,95
1596 DATA 6,173,5,228,105,0
1602 DATA 141,96,6,169,0,133
1608 DATA 203,96,247,238,125,241
1614 DATA 93,6,244,241,115,241
1620 DATA 124,241,76,205,238,0
1626 DATA 0,0,0,0,32,62
1632 DATA 246,8,201,155,240,13
1638 DATA 201,32,240,7,72,24
1644 DATA 101,203,133,203,104,40
1650 DATA 96,72,152,72,138,72
1656 DATA 160,0,169,128,145,88
1662 DATA 200,192,40,208,249,165
1668 DATA 203,74,74,74,74,24
1674 DATA 105,161,160,3,145,88
1680 DATA 165,203,41,15,24,105
1686 DATA 161,200,145,88,169,0
1692 DATA 133,203,104,170,104,168
1698 DATA 104,40,96
```

IBM Proofreader

```
10 'Automatic Proofreader Version 2.00 (L
   ines 270,510,515,517,620,630 changed f
   rom V1.0)
100 DIM L$(500),LNUM(500):COLOR 0,7,7:KEY
    OFF:CLS:MAX=0:LNUM(0)=65536!
110 ON ERROR GOTO 120:KEY 15,CHR$(4)+CHR$
   (7):ON KEY(15) GOSUB 640:KEY (15) ON
   :GOTO 130
120 RESUME 130
130 DEF SEG=&H40:W=PEEK(&H4A)
140 ON ERROR GOTO 650:PRINT:PRINT"Proofre
   ader Ready."
150 LINE INPUT L$:Y=CSRLIN-INT(LEN(L$)/W)
   -1:LOCATE Y,1
160 DEF SEG=0:POKE 1050,30:POKE 1052,34:P
   OKE 1054,0:POKE 1055,79:POKE 1056,13:
   POKE 1057,28:LINE INPUT L$:DEF SEG:IF
    L$="" THEN 150
170 IF LEFT$(L$,1)=" " THEN L$=MID$(L$,2)
   :GOTO 170
180 IF VAL(LEFT$(L$,2))=0 AND MID$(L$,3,1
   )=" " THEN L$=MID$(L$,4)
190 LNUM=VAL(L$):TEXT$=MID$(L$,LEN(STR$(L
   NUM))+1)
200 IF ASC(L$)>57 THEN 260 'no line numbe
   r, therefore command
210 IF TEXT$="" THEN GOSUB 540:IF LNUM=LN
   UM(P) THEN GOSUB 560:GOTO 150 ELSE 15
   0
220 CKSUM=0:FOR I=1 TO LEN(L$):CKSUM=(CKS
   UM+ASC(MID$(L$,I))*I) AND 255:NEXT:LO
   CATE Y,1:PRINT CHR$(65+CKSUM/16)+CHR$
   (65+(CKSUM AND 15))+" "+L$
230 GOSUB 540:IF LNUM(P)=LNUM THEN L$(P)=
   TEXT$:GOTO 150 'replace line
240 GOSUB 580:GOTO 150 'insert the line
260 TEXT$="":FOR I=1 TO LEN(L$):A=ASC(MID
   $(L$,I)):TEXT$=TEXT$+CHR$(A+32*(A>96
   AND A<123)):NEXT
270 DELIMITER=INSTR(TEXT$," "):COMMAND$=T
   EXT$:ARG$="":IF DELIMITER THEN COMMAN
   D$=LEFT$(TEXT$,DELIMITER-1):ARG$=MID$
   (TEXT$,DELIMITER+1) ELSE DELIMITER=IN
   STR(TEXT$,CHR$(34)):IF DELIMITER THEN
    COMMAND$=LEFT$(TEXT$,DELIMITER-1):AR
   G$=MID$(TEXT$,DELIMITER)
280 IF COMMAND$<>"LIST" THEN 410
290 OPEN "scrn:" FOR OUTPUT AS #1
300 IF ARG$="" THEN FIRST=0:P=MAX-1:GOTO
   340
310 DELIMITER=INSTR(ARG$,"-"):IF DELIMITE
   R=0 THEN LNUM=VAL(ARG$):GOSUB 540:FIR
   ST=P:GOTO 340
320 FIRST=VAL(LEFT$(ARG$,DELIMITER)):LAST
   =VAL(MID$(ARG$,DELIMITER+1))
330 LNUM=FIRST:GOSUB 540:FIRST=P:LNUM=LAS
   T:GOSUB 540:IF P=0 THEN P=MAX-1
340 FOR X=FIRST TO P:N$=MID$(STR$(LNUM(X)
   ),2)+" "
350 IF CKFLAG=0 THEN A$="":GOTO 370
360 CKSUM=0:A$=N$+L$(X):FOR I=1 TO LEN(A$
   ):CKSUM=(CKSUM+ASC(MID$(A$,I))*I) AND
    255:NEXT:A$=CHR$(65+CKSUM/16)+CHR$(6
   5+(CKSUM AND 15))+" "
370 PRINT #1,A$+N$+L$(X)
380 IF INKEY$<>"" THEN X=P
390 NEXT :CLOSE #1:CKFLAG=0
400 GOTO 130
410 IF COMMAND$="LLIST" THEN OPEN "lpt1:"
    FOR OUTPUT AS #1:GOTO 300
420 IF COMMAND$="CHECK" THEN CKFLAG=1:GOT
   O 290
430 IF COMMAND$<>"SAVE" THEN 450
440 GOSUB 600:OPEN ARG$ FOR OUTPUT AS #1:
   ARG$="":GOTO 300
450 IF COMMAND$<>"LOAD" THEN 490
460 GOSUB 600:OPEN ARG$ FOR INPUT AS #1:M
   AX=0:P=0
470 WHILE NOT EOF(1):LINE INPUT #1,L$:LNU
   M(P)=VAL(L$):L$(P)=MID$(L$,LEN(STR$(V
   AL(L$)))+1):P=P+1:WEND
480 MAX=P:CLOSE #1:GOTO 130
490 IF COMMAND$="NEW" THEN INPUT "Erase p
   rogram - Are you sure";L$:IF LEFT$(L$
   ,1)="y" OR LEFT$(L$,1)="Y" THEN MAX=0
   :GOTO 130:ELSE 130
500 IF COMMAND$="BASIC" THEN COLOR 7,0,0:
   ON ERROR GOTO 0:CLS:END
510 IF COMMAND$<>"FILES" THEN 520
515 IF ARG$="" THEN ARG$="A:" ELSE SEL=1:
   GOSUB 600
517 FILES ARG$:GOTO 130
520 PRINT"Syntax error":GOTO 130
540 P=0:WHILE LNUM>LNUM(P) AND P<MAX:P=P+
   1:WEND:RETURN
560 MAX=MAX-1:FOR X=P TO MAX:LNUM(X)=LNUM
   (X+1):L$(X)=L$(X+1):NEXT:RETURN
580 MAX=MAX+1:FOR X=MAX TO P+1 STEP -1:LN
   UM(X)=LNUM(X-1):L$(X)=L$(X-1):NEXT:L$
   (P)=TEXT$:LNUM(P)=LNUM:RETURN
600 IF LEFT$(ARG$,1)<>CHR$(34) THEN 520 E
   LSE ARG$=MID$(ARG$,2)
610 IF RIGHT$(ARG$,1)=CHR$(34) THEN ARG$=
   LEFT$(ARG$,LEN(ARG$)-1)
620 IF SEL=0 AND INSTR(ARG$,".")=0 THEN A
   RG$=ARG$+".BAS"
630 SEL=0:RETURN
640 CLOSE #1:CKFLAG=0:PRINT"Stopped.":RET
   URN 150
650 PRINT "Error #";ERR:RESUME 150
```

MLX Machine Language Entry Program For Commodore 64 And VIC-20

Charles Brannon, Program Editor

MLX is a labor-saving utility that allows almost fail-safe entry of machine language programs published in COMPUTE!. You need to know nothing about machine language to use MLX—it was designed for everyone. There are separate versions for the Commodore 64 and expanded VIC-20 (at least 8K).

MLX is a new way to enter long machine language (ML) programs with a minimum of fuss. MLX lets you enter the numbers from a special list that looks similar to BASIC DATA statements. It checks your typing on a line-by-line basis. It won't let you enter illegal characters when you should be typing numbers. It won't let you enter numbers greater than 255 (forbidden in ML). It won't let you enter the wrong numbers on the wrong line. In addition, MLX creates a ready-to-use tape or disk file.

Using MLX

Type in and save the appropriate version of MLX (you'll want to use it in the future). When you're ready to type in an ML program, run MLX. MLX asks you for two numbers: the starting address and the ending address. These numbers are given in the article accompanying the ML program.

When you run MLX, you'll see a prompt corresponding to the starting address. The prompt is the current line you are entering from the listing. It increases by six each time you enter a line. That's because each line has seven numbers—six actual data numbers plus a *checksum number*. The checksum verifies that you typed the previous six numbers correctly. If you enter any of the six numbers wrong, or enter the checksum wrong, the computer rings a buzzer and prompts you to reenter the line. If you enter it correctly, a bell tone sounds and you continue to the next line.

MLX accepts only numbers as input. If you make a typing error, press the INST/DEL key; the entire number is deleted. You can press it as many times as necessary back to the start of the line. If you enter three-digit numbers as listed, the computer automatically prints the comma and goes on to accept the next number. If you enter less than three digits, you can press either the space bar or RETURN key to advance to the next number. The checksum automatically appears in inverse video for emphasis.

To simplify your typing, MLX redefines part of the keyboard as a numeric keypad (lines 581–584):

```
    U  I  O              7  8  9
H   J  K  L  become  0   4  5  6
    M  ,  .              1  2  3
```

MLX Commands

When you finish typing an ML listing (assuming you type it all in one session), you can then save the completed program on tape or disk. Follow the screen instructions. If you get any errors while saving, you probably have a bad disk, or the disk is full, or you've made a typo when entering the MLX program itself.

You don't have to enter the whole ML program in one sitting. MLX lets you enter as much as you want, save it, and then reload the file from tape or disk later. MLX recognizes these commands:

SHIFT-S: Save
SHIFT-L: Load
SHIFT-N: New Address
SHIFT-D: Display

When you enter a command, MLX jumps out of the line you've been typing, so we recommend you do it at a new prompt. Use the Save command to save what you've been working on. It will save on tape or disk, as if you've finished, but the tape or disk won't work, of course, until you finish the typing. Remember what address you stop at. The next time you run MLX, answer all the prompts as you did before, then insert the disk or tape. When you get to the entry prompt, press SHIFT-L to reload the partly completed file into memory. Then use the New Address command to resume typing.

To use the New Address command, press SHIFT-N and enter the address where you previously stopped. The prompt will change, and you can then continue typing. Always enter a New Address that matches up with one of the line numbers in the special listing, or else the checksum won't work. The Display command lets you display a section of your typing. After you press SHIFT-D, enter two addresses within the line number range of the listing. You can abort the listing by pressing any key.

64 MLX: Machine Language Entry

```
10 REM LINES CHANGED FROM MLX VERSION 2.0
   0 ARE 750,765,770 AND 860        :rem 50
20 REM LINE CHANGED FROM MLX VERSION 2.01
    IS 300                          :rem 147
```

```
100 PRINT"{CLR}{6}";CHR$(142);CHR$(8);:PO
    KE53281,1:POKE53280,1          :rem 67
101 POKE 788,52:REM DISABLE RUN/STOP
                                   :rem 119
110 PRINT"{RVS}{39 SPACES}";       :rem 176
120 PRINT"{RVS}{14 SPACES}{RIGHT}{OFF}{*}
    £{RVS}{RIGHT} {RIGHT}{2 SPACES}{*}
    {OFF}{*}£{RVS}£{RVS}{14 SPACES}";
                                   :rem 250
130 PRINT"{RVS}{14 SPACES}{RIGHT} {G}
    {RIGHT} {2 RIGHT} {OFF}£{RVS}£{*}
    {OFF}{*}{RVS}{14 SPACES}";     :rem 35
140 PRINT"{RVS}{41 SPACES}"        :rem 120
200 PRINT"{2 DOWN}{PUR}{BLK} MACHINE LANG
    UAGE EDITOR VERSION 2.02{5 DOWN}"
                                   :rem 238
210 PRINT"{5}{2 UP}STARTING ADDRESS?
    {8 SPACES}{9 LEFT}";           :rem 143
215 INPUTS:F=1-F:C$=CHR$(31+119*F)
                                   :rem 166
220 IFS<256OR(S>40960ANDS<49152)ORS>53247
    THENGOSUB3000:GOTO210          :rem 235
225 PRINT:PRINT:PRINT              :rem 180
230 PRINT"{5}{2 UP}ENDING ADDRESS?
    {8 SPACES}{9 LEFT}";:INPUTE:F=1-F:C$=
    CHR$(31+119*F)                 :rem 20
240 IFE<256OR(E>40960ANDE<49152)ORE>53247
    THENGOSUB3000:GOTO230          :rem 183
250 IFE<STHENPRINTC$;"{RVS}ENDING < START
    {2 SPACES}":GOSUB1000:GOTO 230
                                   :rem 176
260 PRINT:PRINT:PRINT              :rem 179
300 PRINT"{CLR}";CHR$(14):AD=S     :rem 56
310 A=1:PRINTRIGHT$("0000"+MID$(STR$(AD),
    2),5);":";                     :rem 33
315 FORJ=ATO6                      :rem 33
320 GOSUB570:IFN=-1THENJ=J+N:GOTO320
                                   :rem 228
390 IFN=-211THEN 710               :rem 62
400 IFN=-204THEN 790               :rem 64
410 IFN=-206THENPRINT:INPUT"{DOWN}ENTER N
    EW ADDRESS";ZZ                 :rem 44
415 IFN=-206THENIFZZ<SORZZ>ETHENPRINT"
    {RVS}OUT OF RANGE":GOSUB1000:GOTO410
                                   :rem 225
417 IFN=-206THENAD=ZZ:PRINT:GOTO310
                                   :rem 238
420 IF N<>-196 THEN 480            :rem 133
430 PRINT:INPUT"DISPLAY:FROM";F:PRINT,"TO
    ";:INPUTT                      :rem 234
440 IFF<SORF>EORT<SORT>ETHENPRINT"AT LEAS
    T";S;"{LEFT}, NOT MORE THAN";E:GOTO43
    0                              :rem 159
450 FORI=FTOTSTEP6:PRINT:PRINTRIGHT$("000
    0"+MID$(STR$(I),2),5);":";     :rem 30
451 FORK=0TO5:N=PEEK(I+K):PRINTRIGHT$("00
    "+MID$(STR$(N),2),3);",";      :rem 66
460 GETA$:IFA$>""THENPRINT:PRINT:GOTO310
                                   :rem 25
470 NEXTK:PRINTCHR$(20);:NEXTI:PRINT:PRIN
    T:GOTO310                      :rem 50
480 IFN<0 THEN PRINT:GOTO310       :rem 168
490 A(J)=N:NEXTJ                   :rem 199
500 CKSUM=AD-INT(AD/256)*256:FORI=1TO6:CK
    SUM=(CKSUM+A(I))AND255:NEXT    :rem 200
510 PRINTCHR$(18);:GOSUB570:PRINTCHR${146
    );                             :rem 94
511 IFN=-1THENA=6:GOTO315          :rem 254
515 PRINTCHR$(20):IFN=CKSUMTHEN530
                                   :rem 122
520 PRINT:PRINT"LINE ENTERED WRONG : RE-E
    NTER":PRINT:GOSUB1000:GOTO310:rem 176
530 GOSUB2000                      :rem 218
540 FORI=1TO6:POKEAD+I-1,A(I):NEXT:POKE54
    272,0:POKE54273,0              :rem 227
550 AD=AD+6:IF AD<E THEN 310       :rem 212
560 GOTO 710                       :rem 108
570 N=0:Z=0                        :rem 88
580 PRINT"{£}";                    :rem 81
581 GETA$:IFA$=""THEN581           :rem 95
582 AV=-(A$="M")-2*(A$=",")-3*(A$=".")-4*
    (A$="J")-5*(A$="K")-6*(A$="L"):rem 41
583 AV=AV-7*(A$="U")-8*(A$="I")-9*(A$="O"
    ):IFA$="H"THENA$="0"           :rem 134
584 IFAV>0THENA$=CHR$(48+AV)       :rem 134
585 PRINTCHR$(20);:A=ASC(A$):IFA=13ORA=44
    ORA=32THEN670                  :rem 229
590 IFA>128THENN=-A:RETURN         :rem 137
600 IFA<>20 THEN 630               :rem 10
610 GOSUB690:IFI=1ANDT=44THENN=-1:PRINT"
    {OFF}{LEFT} {LEFT}";:GOTO690   :rem 62
620 GOTO570                        :rem 109
630 IFA<48ORA>57THEN580            :rem 105
640 PRINTA$;:N=N*10+A-48           :rem 106
650 IFN>255 THEN A=20:GOSUB1000:GOTO600
                                   :rem 229
660 Z=Z+1:IFZ<3THEN580             :rem 71
670 IFZ=0THENGOSUB1000:GOTO570     :rem 114
680 PRINT",";:RETURN               :rem 240
690 S%=PEEK(209)+256*PEEK(210)+PEEK(211)
                                   :rem 149
691 FORI=1TO3:T=PEEK(S%-I)         :rem 67
695 IFT<>44ANDT<>58THENPOKES%-I,32:NEXT
                                   :rem 205
700 PRINTLEFT$("{3 LEFT}",I-1);:RETURN
                                   :rem 7
710 PRINT"{CLR}{RVS}*** SAVE ***{3 DOWN}"
                                   :rem 236
715 PRINT"{2 DOWN}(PRESS {RVS}RETURN{OFF}
     ALONE TO CANCEL SAVE){DOWN}":rem 106
720 F$="":INPUT"{DOWN} FILENAME";F$:IFF$=
    ""THENPRINT:PRINT:GOTO310      :rem 71
730 PRINT:PRINT"{2 DOWN}{RVS}T{OFF}APE OR
    {RVS}D{OFF}ISK: (T/D)"         :rem 228
740 GETA$:IFA$<>"T"ANDA$<>"D"THEN740
                                   :rem 36
750 DV=1-7*(A$="D"):IFDV=8THENF$="0:"+F$:
    OPEN15,8,15,"S"+F$:CLOSE15     :rem 212
760 T$=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(T$
    ):POKE782,ZK/256               :rem 3
762 POKE781,ZK-PEEK(782)*256:POKE780,LEN(
    T$):SYS65469                   :rem 109
763 POKE780,1:POKE781,DV:POKE782,1:SYS654
    66                             :rem 69
765 K=S:POKE254,K/256:POKE253,K-PEEK(254)
    *256:POKE780,253               :rem 17
766 K=E+1:POKE782,K/256:POKE781,K-PEEK(78
    2)*256:SYS65496                :rem 235
770 IF(PEEK(783)AND1)OR(191ANDST)THEN780
                                   :rem 111
775 PRINT"{DOWN}DONE.{DOWN}":GOTO310
                                   :rem 113
780 PRINT"{DOWN}ERROR ON SAVE.{2 SPACES}T
    RY AGAIN.":IFDV=1THEN720       :rem 171
781 OPEN15,8,15:INPUT#15,E1$,E2$:PRINTE1$
    ;E2$:CLOSE15:GOTO720           :rem 103
790 PRINT"{CLR}{RVS}*** LOAD ***{2 DOWN}"
                                   :rem 212
795 PRINT"{2 DOWN}(PRESS {RVS}RETURN{OFF}
     ALONE TO CANCEL LOAD)"        :rem 82
800 F$="":INPUT"{2 DOWN} FILENAME";F$:IFF
    $=""THENPRINT:GOTO310          :rem 144
810 PRINT:PRINT"{2 DOWN}{RVS}T{OFF}APE OR
    {RVS}D{OFF}ISK: (T/D)"         :rem 227
```

```
820 GETA$:IFA$<>"T"ANDA$<>"D"THEN820
                                    :rem 34
830 DV=1-7*(A$="D"):IFDV=8THENF$="0:"+F$
                                    :rem 157
840 T$=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(T$
    ):POKE782,ZK/256                :rem 2
841 POKE781,ZK-PEEK(782)*256:POKE780,LEN(
    T$):SYS65469                    :rem 107
845 POKE780,1:POKE781,DV:POKE782,1:SYS654
    66                              :rem 70
850 POKE780,0:SYS65493              :rem 11
860 IF(PEEK(783)AND1)OR(191ANDST)THEN870
                                    :rem 111
865 PRINT"{DOWN}DONE.":GOTO310      :rem 96
870 PRINT"{DOWN}ERROR ON LOAD.{2 SPACES}T
    RY AGAIN.{DOWN}":IFDV=1THEN800
                                    :rem 172
880 OPEN15,8,15:INPUT#15,E1$,E2$:PRINTE1$
    ;E2$:CLOSE15:GOTO800            :rem 102
1000 REM BUZZER                     :rem 135
1001 POKE54296,15:POKE54277,45:POKE54278,
     165                            :rem 207
1002 POKE54276,33:POKE 54273,6:POKE54272,
     5                              :rem 42
1003 FORT=1TO200:NEXT:POKE54276,32:POKE54
     273,0:POKE54272,0:RETURN       :rem 202
2000 REM BELL SOUND                 :rem 78
2001 POKE54296,15:POKE54277,0:POKE54278,2
     47                             :rem 152
2002 POKE 54276,17:POKE54273,40:POKE54272
     ,0                             :rem 86
2003 FORT=1TO100:NEXT:POKE54276,16:RETURN
                                    :rem 57
3000 PRINTC$;"{RVS}NOT ZERO PAGE OR ROM":
     GOTO1000                       :rem 89
```

VIC MLX: Machine Language Entry

```
100 PRINT"{CLR}{PUR}";CHR$(142);CHR$(8);
                                    :rem 181
101 POKE 788,194:REM DISABLE RUN/STOP
                                    :rem 174
110 PRINT"{RVS}{14 SPACES}"         :rem 117
120 PRINT"{RVS} {RIGHT}{OFF}£{*}£{RVS}
    {RIGHT} {RIGHT}{2 SPACES}£{*}{OFF}£{*}
    £{RVS}£{RVS} "                  :rem 191
130 PRINT"{RVS} {RIGHT} {G}{RIGHT}
    {2 RIGHT} {OFF}£{RVS}£{*}{OFF}£{*}
    {RVS} "                         :rem 232
140 PRINT"{RVS}{14 SPACES}"         :rem 120
200 PRINT"{2 DOWN}{PUR}{BLK}A FAILSAFE MA
    CHINE":PRINT"LANGUAGE EDITOR{5 DOWN}"
                                    :rem 141
210 PRINT"{BLK}{3 UP}STARTING ADDRESS":IN
    PUTS:F=1-F:C$=CHR$(31+119*F)    :rem 97
220 IFS<256ORS>32767THENGOSUB3000:GOTO210
                                    :rem 2
225 PRINT:PRINT:PRINT:PRINT         :rem 123
230 PRINT"{BLK}{3 UP}ENDING ADDRESS":INPU
    TE:F=1-F:C$=CHR$(31+119*F)      :rem 158
240 IFE<256ORE>32767THENGOSUB3000:GOTO230
                                    :rem 234
250 IFE<STHENPRINTC$;"{RVS}ENDING < START
    {2 SPACES}":GOSUB1000:GOTO 230
                                    :rem 176
260 PRINT:PRINT:PRINT               :rem 179
300 PRINT"{CLR}";CHR$(14):AD=S      :rem 56
310 PRINTRIGHT$("0000"+MID$(STR$(AD),2),5
    );":";:FORJ=1TO6                :rem 234
320 GOSUB570:IFN=-1THENJ=J+N:GOTO320
                                    :rem 228
390 IFN=-211THEN 710                :rem 62
400 IFN=-204THEN 790                :rem 64
410 IFN=-206THENPRINT:INPUT"{DOWN}ENTER N
    EW ADDRESS";ZZ                  :rem 44
415 IFN=-206THENIFZZ<SORZZ>ETHENPRINT"
    {RVS}OUT OF RANGE":GOSUB1000:GOTO410
                                    :rem 225
417 IFN=-206THENAD=ZZ:PRINT:GOTO310
                                    :rem 238
420 IF N<>-196 THEN 480             :rem 133
430 PRINT:INPUT"DISPLAY:FROM";F:PRINT,"TO
    ";:INPUTT                       :rem 234
440 IFF<SORF>EORT<SORT>ETHENPRINT"AT LEAS
    T";S;"{LEFT}, NOT MORE THAN";E:GOTO43
    0                               :rem 159
450 FORI=FTOTSTEP6:PRINT:PRINTRIGHT$("000
    0"+MID$(STR$(I),2),5);":";      :rem 30
455 FORK=0TO5:N=PEEK(I+K):IFK=3THENPRINTS
    PC(10);                         :rem 34
457 PRINTRIGHT$("00"+MID$(STR$(N),2),3);"
    ,";                             :rem 157
460 GETA$:IFA$>""THENPRINT:PRINT:GOTO310
                                    :rem 25
470 NEXTK:PRINTCHR$(20);:NEXTI:PRINT:PRIN
    T:GOTO310                       :rem 50
480 IFN<0 THEN PRINT:GOTO310        :rem 168
490 A(J)=N:NEXTJ                    :rem 199
500 CKSUM=AD-INT(AD/256)*256:FORI=1TO6:CK
    SUM=(CKSUM+A(I))AND255:NEXT     :rem 200
510 PRINTCHR$(18);:GOSUB570:PRINTCHR$(20)
                                    :rem 234
515 IFN=CKSUMTHEN530                :rem 255
520 PRINT:PRINT"LINE ENTERED WRONG":PRINT
    "RE-ENTER":PRINT:GOSUB1000:GOTO310
                                    :rem 129
530 GOSUB2000                       :rem 218
540 FORI=1TO6:POKEAD+I-1,A(I):NEXT:rem 80
550 AD=AD+6:IF AD<E THEN 310        :rem 212
560 GOTO 710                        :rem 108
570 N=0:Z=0                         :rem 88
580 PRINT"{+}";                     :rem 79
581 GETA$:IFA$=""THEN581            :rem 95
585 PRINTCHR$(20);:A=ASC(A$):IFA=13ORA=44
    ORA=32THEN670                   :rem 229
590 IFA>128THENN=-A:RETURN          :rem 137
600 IFA<>20 THEN 630                :rem 10
610 GOSUB690:IFI=1ANDT=44THENN=-1:PRINT"
    {LEFT} {LEFT}";:GOTO690         :rem 172
620 GOTO570                         :rem 109
630 IFA<48ORA>57THEN580             :rem 105
640 PRINTA$;:N=N*10+A-48            :rem 106
650 IFN>255 THEN A=20:GOSUB1000:GOTO600
                                    :rem 229
660 Z=Z+1:IFZ<3THEN580              :rem 71
670 IFZ=0THENGOSUB1000:GOTO570      :rem 114
680 PRINT",";:RETURN                :rem 240
690 S%=PEEK(209)+256*PEEK(210)+PEEK(211)
                                    :rem 149
692 FORI=1TO3:T=PEEK(S%-I)          :rem 68
695 IFT<>44ANDT<>58THENPOKES%-I,32:NEXT
                                    :rem 205
700 PRINTLEFT$("{3 LEFT}",I-1);:RETURN
                                    :rem 7
710 PRINT"{CLR}{RVS}*** SAVE ***{3 DOWN}"
                                    :rem 236
720 INPUT"{DOWN} FILENAME";F$       :rem 228
730 PRINT:PRINT"{2 DOWN}{RVS}T{OFF}APE OR
    {RVS}D{OFF}ISK: (T/D)"          :rem 228
740 GETA$:IFA$<>"T"ANDA$<>"D"THEN740
                                    :rem 36
750 DV=1-7*(A$="D"):IFDV=8THENF$="0:"+F$
                                    :rem 158
760 T$=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(T$
    ):POKE782,ZK/256                :rem 3
```

```
762 POKE781,ZK-PEEK(782)*256:POKE780,LEN(
    T$):SYS65469                    :rem 109
763 POKE780,1:POKE781,DV:POKE782,1:SYS654
    66                              :rem 69
765 POKE254,S/256:POKE253,S-PEEK(254)*256
    :POKE780,253                    :rem 12
766 POKE782,E/256:POKE781,E-PEEK(782)*256
    :SYS65496                       :rem 124
770 IF(PEEK(783)AND1)OR(ST AND191)THEN780
                                    :rem 111
775 PRINT"{DOWN}DONE.":END          :rem 106
780 PRINT"{DOWN}ERROR ON SAVE.{2 SPACES}T
    RY AGAIN.":IFDV=1THEN720        :rem 171
781 OPEN15,8,15:INPUT#15,E1$,E2$:PRINTE1$
    ;E2$:CLOSE15:GOTO720            :rem 103
782 GOTO720                         :rem 115
790 PRINT"{CLR}{RVS}*** LOAD ***{2 DOWN}"
                                    :rem 212
800 INPUT"{2 DOWN} FILENAME";F$     :rem 244
810 PRINT:PRINT"{2 DOWN}{RVS}T{OFF}APE OR
     {RVS}D{OFF}ISK: (T/D)"         :rem 227
820 GETA$:IFA$<>"T"ANDA$<>"D"THEN820
                                    :rem 34
830 DV=1-7*(A$="D"):IFDV=8THENF$="0:"+F$
                                    :rem 157
840 T$=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(T$
    ):POKE782,ZK/256                :rem 2
841 POKE781,ZK-PEEK(782)*256:POKE780,LEN(
    T$):SYS65469                    :rem 107
845 POKE780,1:POKE781,DV:POKE782,1:SYS654
    66                              :rem 70
850 POKE780,0:SYS65493              :rem 11
860 IF(PEEK(783)AND1)OR(ST AND191)THEN870
                                    :rem 111
865 PRINT"{DOWN}DONE.":GOTO310      :rem 96
870 PRINT"{DOWN}ERROR ON LOAD.{2 SPACES}T
    RY AGAIN.{DOWN}":IFDV=1THEN800
                                    :rem 172
880 OPEN15,8,15:INPUT#15,E1$,E2$:PRINTE1$
    ;E2$:CLOSE15:GOTO800            :rem 102
1000 REM BUZZER                     :rem 135
1001 POKE36878,15:POKE36874,190     :rem 206
1002 FORW=1TO300:NEXTW              :rem 117
1003 POKE36878,0:POKE36874,0:RETURN
                                    :rem 74
2000 REM BELL SOUND                 :rem 78
2001 FORW=15TO0STEP-1:POKE36878,W:POKE368
    76,240:NEXTW                    :rem 22
2002 POKE36876,0:RETURN             :rem 119
3000 PRINTC$;"{RVS}NOT ZERO PAGE OR ROM":
    GOTO1000                        :rem 89
```

©

COMPUTE!

Modifications Or Corrections To Previous Articles

Enhanced 64 DOS Support

Author Stephen Melsheimer reports that a number of readers have uncovered a problem with the "LOAD and RUN" (↑) command of this utility (p. 163, November 1984). Occasionally a program—especially one with DATA statements—will fail to load and run properly. To correct this, he suggests that the following two lines be modified as shown:

```
1510 DATA 144, 255, 32, 89, 166, 76
1630 DATA 76, 144, 205, -1, 9616
```

Additionally, the article (p. 166) mentions that the wedge can be activated from within a program with SYS 52222. The correct value is SYS 52224.

64 Paintbox

There are several errors in the article for this graphics utility from the December 1984 issue. It states (p. 118) that you should use the filename 64 PAINTBOX if you want to use the loader program (Program 2). Actually, Program 2 looks for the filename PAINTBOX. You can either rename the saved file to PAINTBOX or change the string in line 230 of Program 2 to 64 PAINTBOX. Also, a line is missing in the first paragraph of the right column of that page. The sentence in parentheses should read: "(If you're using tape, Program 2 should precede 64 Paintbox on the tape, and the 8 in line 230 should be changed to a 1.)"

Apple Chess

The Apple version of "Chess" from the December 1984 issue (p. 102) works fine on the Apple IIc, but the program doesn't respond to keyboard commands when used on the II+ or IIe. To correct this, change the following lines:

```
350 IF PEEK (-16384) < 128 THEN 350
550 I = PEEK (-16384): POKE -16368,0
```

Thanks to Gene and Brian Schmidt for discovering this oversight.

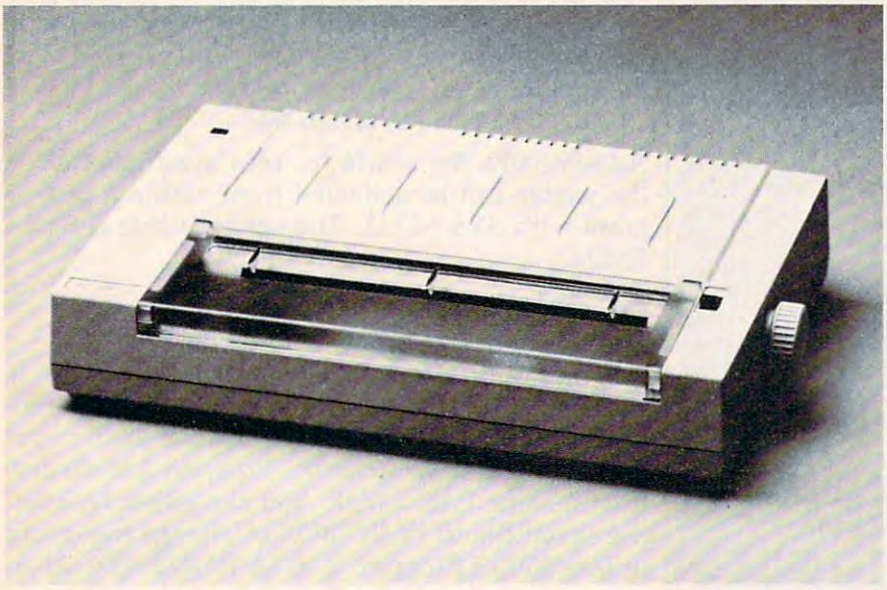
Atari Reflection

The Atari version (Program 1, p. 59) of this game from the November 1984 issue requires the two players to share one joystick. If you have two joysticks, reader William Q. Zapf suggests the following simple modifications to allow the program to read two joysticks:

```
NJ 1240 POKE 77,N0:Q=STICK(TURN-1):IF
        Q=10 OR Q=14 OR Q=N6 THEN IF
        YP>N1 THEN YP=YP-N1
MO 1340 IF STRIG(TURN-1) THEN 1240
```

©

NEWS&PRODUCTS



The Riteman LQ is a letter-quality printer that retails for $299 and prints 20 characters per second.

New Printers

Riteman Computer Printers has introduced a line of printers which is compatible with most home computers. Among the new printers are:

The Riteman LQ ($299), a letter-quality printer which has a print speed of 20 characters per second (cps); the Riteman Plus ($399), a dot-matrix printer which can print at 120 cps; the Riteman Blue Plus ($499), a dot-matrix printer which prints at 140 cps and can print graphics. Included are eight international character sets and 32 block graphic characters. The Riteman II ($549 with 2K RAM and $599 with 8K RAM) is a dot-matrix printer with a print speed of 160 cps.

Riteman Computer Printers
Airport Business Park
431 Oak St.
Inglewood, CA 90302

Apple Home Applications Package

Work Force II, a collection of six programs for home and office, has been introduced by Core Concepts for Apple II and III computers.

Included in the package are a loan analyzer; a checkbook balancer; a calculator; the line writer, a line-at-a-time word processor for such small writing jobs as envelopes; the savings analyzer; and the wages analyzer.

All six programs support printing, so hard copies can be made of data. Suggested retail price is $34.95.

Core Concepts
P.O. Box 24157
Tempe, AZ 85282

New Games For Commodore, Atari

Microcomputer Games, a division of The Avalon Hill Game Company, has released a number of new games for various computers. Among the new titles are the following:

Market Forces, a business game which simulates the buying and selling of commodities ($16 cassette; $21 disk, for

A screen shot from Market Forces, *a new game from Microcomputer Games, Inc., for the Commodore 64 and Atari computers.*

Commodore 64 and Atari computers); *Gulf Strike*, a simulation of land, air, and naval combat in the Middle East ($30 disk, for Atari computers); and *Clear for Action*, a ship-to-ship combat game ($25 cassette, $30 disk, Atari computers).

Microcomputer Games, Inc.
The Avalon Hill Game Co.
4517 Harford Rd.
Baltimore, MD 21214

Apple II Word Processor

The *Milliken Word Processor*, a program designed to teach children the fundamentals of writing on a computer, has been released for Apple II series computers by Milliken Publishing Company.

It teaches how to use the computer for composing, structuring, editing, and filing written material, and has most basic word-processing functions, including graphics to ease understanding of functions.

Designed for children ages seven and older, the word processor retails for $69.95.

Milliken Publishing Co.
1100 Research Blvd.
P.O. Box 21579
St. Louis, MO 63132

New Text Adventures

Infocom has released two new text adventures for most home computers, *Suspect* and *The Hitchhiker's Guide to the Galaxy.*

In *Suspect*, the player takes the role of a newspaper reporter invited to a masquerade ball—who ends up being accused of murder. You must prove your innocence, and also find out who committed the crime and why.

The Hitchhiker's Guide to the Galaxy is an adaptation of the novel of the same name by Douglas Adams. The player is protagonist Arthur Dent, who goes off on a journey through the universe with his friend, Ford Prefect.

Suspect retails for $39.95 for the Atari and Commodore 64 versions, and $44.95 for versions on most other personal computers. *Hitchhiker's Guide to the Galaxy* has a suggested retail price of $34.95 for the Commodore and Atari versions, with other versions retailing for $39.95.

Infocom, Inc.
55 Wheeler St.
Cambridge, MA 02138

Bridge For IBM, Apple, Commodore

A bridge game for one or more players, *BridgePro*, has been released for the IBM PC and PCjr, Apple II series, and Commodore 64 computers by Computer Management Corporation.

BridgePro allows one person to bid with all hands randomly dealt. Other options are two-player versions, a best hand option, replay of hands, and separate instructions for beginning bridge players.

Suggested retail price is $35. *BridgePro* is available on disk.

Computer Management Corporation
2424 Exbourne Ct.
Walnut Creek, CA 94596

Critical Thinking Program

MaxThink, a program for the IBM PC and compatibles which provides commands for high-level thinking processes such as analysis, synthesis, and evaluation, has been introduced by MaxThink, Inc.

The software uses commands for organizing, analyzing, evaluating, planning, and thinking about information.

The suggested retail price is $60. It requires 192K of memory.

MaxThink, Inc.
230 Crocker Ave.
Piedmont, CA 94610

Personal Productivity Software

Arrays, Inc./Continental Software has introduced several personal productivity software packages for Apple and IBM computers, including:

The Home Accountant Expanded, an accounting package for Apple IIc and IIe computers ($74.95 suggested retail price); and a new, compiled version of *The Home Accountant Plus* ($149.95) for the IBM PC.

Also, educational versions of *Ultra-File*, a filing, reporting, and graphics package; *Property Management*, a program to record transaction history for residential and/or commercial income property-related charges;

Lyco Computer Marketing & Consultants

TOLL FREE 800-233-8760

apple

Scarborough

Songwriter	$24.75
Picturewrit	$24.75
Phi Beta F	$32.75
Mastertype	$24.75
Run f Money	$24.75
Net Worth	$52.75

Microprose

Solo Flight	$22.75
NATO	$22.75
Spitfire	$22.95
F-15 Strike	$22.75
Air Rescue	$22.75

SSI

Baseball	$22.75
Questron	$26.75
Germany 1985	$32.75
50 Missions	$21.75

Spinnaker

Alphabet	$18.75
Story Machine	$19.75
Kids on Keys	$18.75
Grandma	$19.75
Snooper Troop	$22.75

Broderbund

Bank St. Writer	$42.75
Bank St. Filer	$42.75
Bank St. Mailer	$42.75
Bank St. Spell	$42.75
Mask of Sun	$24.95
Choplifter	$22.95
Lode Runner	$22.95

Graphics Tablet

Supersketch	$49.95
Kolala	$84.95

ATARI

800XL COMPUTER	CALL
1050 DRIVE	CALL
1010 RECORDER	$55.00
1020 PRINTER	$59.00
1025 PRINTER	$189.00
1027 PRINTER	$249.00
1030 MODEM	$59.00
MONKEYWRENCH II	$52.75
HOME ACCOUNT D	$44.75
TAX ADVANTAGE	$35.75

SUB LOGIC

Flight Simulator II C-64	32.75
Flight Simulator II Atari	32.75
Flight Simulator II Apple	32.75

Trillium

Shadowkeep	$26.75
Fahrenheit 451	$26.75
Amazon	$26.75

Synapse

Synfile	$48.95
Syncalc	$48.95
Syncomm	$29.95
Syntrend	$48.95

Graphics Tablet

Supersketch	$32.95
Kolala	$69.95
THE ILLUSTRATOR	$99.95
SPIDER EATER	$22.50
SPEEICOPTER	$27.75

BUSINESS

VISICALC	$159.75
LETTER PERFECT R	59 00
DATA PERFECT	$89.75
FILE MANAGER	$69.75
HOME FILE MGR	$69.75

DEADLINE	$34.75
ENCHANTER	$34.75
INFIDEL	$34.75
PLANETFALL	$34.75
STAR CROSS	$34.75
SUSPENDED	$34.75
WITNESS	$34.75
ZORK I	$34.75
ZORK II	$34.75

Scarborough

Songwriter	$24.75
Picturewrit	$24.75
Mastertype	$24.75
Run f Money	$24.75

Microprose

Solo Flight	$22.75
NATO	$22.75
Spitfire	$19.95
F-15 Strike	$22.75
Air Rescue	$22.75

SSI

Baseball	$22.75
Questron	$26.75
50 Missions	$21.75

Spinnaker

Alphabet	$18.75
Story Machine	$21.75
Kids on Keys	$18.75
Grandma	$19.75
Snooper Troop	$22.75

Broderbund

Bank St. Writer	$42.75
Spellmaker	$19.95
Mask of Sun	$24.95
Choplifter	$22.95
Lode Runner	$22.95

COMMODORE

COMMODORE

C64 COMPUTER	CALL
SX 64 COMPUTER	CALL
C1541 DISK DRIVE	$239.00
C1526 PRINTER	$269.00
MPS801 PRINTER	$215.00
C1702 MONITOR	$249.00
C64105 LOGO 64	$45.00
C64106 PILOT 64	$35.00
SIMON'S BASIC	$29.00

SSI

Baseball	$22.75
Germany 1984	$32.75
50 Missions	$21.75

PERSMAL PERIPHEIALS

Super Sketch-Atari	32.95
Super Sketch-C-64	37.95
Super Sketch-TI99/4A	37.95
Super Sketch-Apple	52.95
Super Sketch-IBM PC	52.95

Timeworks

Inventory	$32.75
Sales	$32.75
Accts. Rec	$32.75
Accts. Rec	$32.75
G. Ledger	$39.75
Data Mgr	$14.75
Checkbook	$14.75
Star Battle	$14.75
Cave of Word	$18.75

Spinnaker

Alphabet	$18.75
Story Machine	$21.75
Kids on Keys	$18.75
Grandma	$19.75
Kidwriter	$19.75
Snooper Troop	$21.75

CARDCO

LIGHT PEN	$29.75
5 SLOT EXPAN. 64	$54.00
64 WRITE NOW	$39.00
64 MAIL NOW	$29.00
20 WRITE NOW	$29.00
64 KEYPAD	$64.00
UNIV. CASS. INT	$29.75
PRINTER UTILITY	$19.75
6 SLOT EXPAN.	$79.96
3 SLOT EXPAN.	$24.95

Scarborough

Songwriter	$24.75
Picturewrit	$24.75
Phi Beta F	$24.75
Mastertype	$24.75
Run f Money	$24.75
Net Worth	$24.75

Batteries Included

Paper Clip	$59.95
Spell Pak	$34.95
Consultant	$64.95
Paper Clip with Spell Pak	$79.95
Home Pak	$34.95
BUS CARD	$139.95
80 Column Board	$139.95

Microprose

Solo Flight	$22.75
NATO	$22.75
Spitfire	$19.95
F-15 Strike	$22.75
Air Rescue	$22.75

ADVENTURE

Diskey	32.95
Ultra Disassembler	32.95

MONITORS

TAXAN

210 Color RGB	255
100 Green	115
105 Amber	125
400 Color RGB	295
410 Color RGB	349
420 Color IBM	449
121 Green IBM	145
122 Amber IBM	149

ZENITH

ZVM 122A Amber	86
ZVM 1236 Green	82
ZVM 124 Amber - IBM	129
ZVM 131 Color	275
ZVM 133 RGB	389
ZVM 135 Composite	449
ZVM 136 Hi Res Color	589

GORILLA

12" Green	$ 82.00
12" Amber	$ 88.00

AMDEK

300 Green	125
300 Amber	145
310 Amber - IBM	159
Color 300-Audio	265
Color 500-Composite	379
Color 600	545
Color 700	635
Color 710	675

NEC

JB 1260 Green	99.00
JB 1201 Green	135.00
JB 1205 Amber	145.00
JC 1215 Color	255.00
JC 1216 RGB	399.00
JC 460 Color	349.00

SAKATA

SC-100 Color	229
STSI Tinstand	29
SG 1000 Green	99
SA 1000 Amber	109

MODEMS

MICROBITS

MPP1000C	$109.00

NOVATION

J-Cat	$89.00
Cat	$129.00
Smart Cat 103/	$169.00
Smart Cat 103/212	$389.00
AutoCat	$209.00
212 Auto Cat	$539.00
Apple Cat II	$239.00
212 Apple Cat	$439.00
Apple Cat 212 (Upgrade)	$249.00
Smart Cat Plus	$359.00

Hayes

Smartmodem 300	$199.00
Smartmodem 1200	$469.00
Smartmodem 1200b	$399.00
Micromodem IIe	$249.00
Micromodem 100	$289.00
Chronograph	$179.00

ANCHOR

Volksmodem	$55.99
Mark VII (auto ans/dial)	$95.99
Mark VII (1200 band)	$259.00

HES

HES Games 84	22.95
Omni Writer/Spell	34.95
HES Mon 64	23.95
Microsoft Multiplan	55.00
Type N Write	19.95
Turtle Graphics II	23.95
Cell Defense	22.95
Paint Brush	12.95
Tri Math	22.95
Graphics Basic	27.95
HES Kit	29.95
Millionaire	23.95
64 Forth	24.95
HES Writer 64	24.95

Westridge C-64 ... Call

Total Telecommunications C-64 ... Call

Mitey Mo C-64 ... Call

AMERICA'S MAIL ORDER HEADQUARTERS

LYCO COMPUTER

WORLD'S LEADER IN SALES & SERVICE

TO ORDER
CALL TOLL FREE
800-233-8760
In PA 1 717-327-1824
Lyco Computer
P.O. Box 5088
Jersey Shore, PA 17740

Lyco Computer Marketing & Consultants

"PEOPLE WHO KNOW WHAT THEY WANT AND KNOW HOW TO USE IT RECEIVE THE LOWEST PRICES AT LYCO"

SAVE ON THESE IN-STOCK PRINTERS

MANNESMANN TALLY

SPIRIT 80	$255.00
MTL-160L	$549.00
MTL-180L	$739.00

JUKI

Juki 6100	$389
Tractor kit	$119

Epson

RX80	$229.00
RX80FT	$269.00
RX100	$369.00
FX80	$369.00
FX100	$555.00
JX80	$1089.00
LQ1500P includes Kit	$1149.00
LQ1500S	$529.00

Citoh

Prowiter 8510A	$289.00
8510BC2	$399.00
8510BP1	$349.00
8510SP	$399.00
8510SR	$409.00
8510SCP	$419.00
8510SCR	$499.00
1550P	$489.00
1550BCD	$539.00
A10-20P	$469.00
F1040PU or RDU	$899.00
F1055PU or RDU	$1099.00

PANASONIC

1090	$219.00
1091	$279.00
1092	$415.00
1093	$599.00
3151	$469.00

BLUE CHIPS

M12010	$275.00
M12010 C-64	$275.00
D4015	$1389.00

OKIDATA

80	$159
82A	$229.00
83A	$549.00
84	$649.00
92	$359.00
93	$569.00

LEGEND

880	$259.00
1000	$279.00
1200	CALL
1500	CALL
1081	CALL

PRINTER INTERFACING Available

NEC

NEC 8025	$699.00
NEC 8027	$359.00

STAR MICRONICS

Gemini 10x	$229.00
Gemini 15x	$345.00
Delta 10	$339.00
Delta 15	$449.00
Radix 10	$499.00
Radix 15	$589.00
Powertype	$309.00
Sweet P 100	$549.00

GEMINI 10X
$229 ★

CARDCO

LQ1	$449.00
LQ3	$339.00
PRINTER INTERFACE	$39.75
PRINTER INTERFACE W/ FULL GRAPHICS	$65.75

OVER 2000 SOFTWARE TITLES IN STOCK

COMPUTER CARE

BIB

DISK DRIVE CLEANER	$12.75
COMPUTER CARE KIT	$19.75

NORTRONICS DISK DRIVE CLEANER with software for IBM-PC, Atari, Vic,

DISK DRIVE CLEANER with software for IBM-PC, Atari, Vic, Apple, TI	$29.75
DISK CLEANER REFILL	$14.75
CASSDRIVE CLEANER	$9.95
MEDIA BULK ERASER	$46.75

NEC

PC8201 Portable	$429
NECB1 64K Computer System	$1049
NECB2 128 K Computer System	$1299
PC8221 Thermal Printer	$ 139
PC8201 8K RAM Chip	$ 99
PC8206 32K RAM Cart	$ 299
PC300 Modem	$ 65
PC8801 MSDOS 16 Bit Card	$ 339

PRINTING PAPER

3000 SHEETS FANFOLD	$42.75
1000 SHEETS FANFOLD	$19.75
1000 SHEET LETTER	$21.95
200 SHEETS LETTER	$8.99
150 RAG STATIONARY	$10.99
MAILING LABELS (1 in.)	$9.95
14 x 11 1000 FANFOLD	$24.75

INNOVATIVE CONCEPTS

FLIP-N-FILE 10	$3.75
FLIP-N-FILE 15	$8.95
FLIP-N-FILE 25	$18.95
FLIP-N-FILE 50	$17.75
FLIP-N-FILE (ROM HOLDER)	$17.75

DRIVES

MSD

SD1 DRIVE	$259.00
SD2 DRIVE	$475.00

INDUS

GT Atari 269

GT Atari	269
GT Commodore	CALL
GT Apple w/controller	219
GT Apple	169

DISKETTES

SKC (Box 10)

SKC-SSSD	$12.99
SKC-SSDD	$15.99
SKC-DSDD	$18.99

ELEPHANT (Box 10)

5¼"SSSD	$14.99
5¼"SSDD	$16.99
5¼"DSDD	$21.99

MAXELL (Box 10)

5¼"MD-1	$17.95
5¼"MD-2	$23.95

IBM-PC SOFT-WARE

Scarborough

Songwriter	$32.75
Picturewrite	$32.75
Phi Beta F	$32.75
Mastertype	$32.75
Run f Money	$32.75
Net Worth	$CALL

Spinnaker

Alphabet	$18.75
Kids on Keys	$19.75
Grandma	$19.75
Kindercomp	$17.75
Facemaker	$19.75
Kidwriter	$19.75

IBM-PC COMPATABLE

CORONA

PPC22A Portable 256K-Amber	$1995
PPC22G Portable 256K-Green	$1995
PPCXTA Portable 256K-10Meg	$3295
COR128K 128K RAM	$ 159

Zenith

Z-150	Call

Columbia Data

1600	Call

Televideo

TS1605	Call

Leading Edge

PC Compatable	Call

Microprose

Solo Flight	$22.75
NATO	$22. 75
Spitfire	$22.95

Graphics Tablet

Supersketch	$49.95
Kolala	$99.95
Illustrator	$99.95
Logo Design	$27.95
Grams Spell	$27.95

TOLL FREE 1-800-233-8760

MasterCard TO ORDER VISA

CALL TOLL FREE
800-233-8760

or send order to
Lyco Computer
P O Box 5088

Customer Service 1-717-327-1825 Jersey Shore PA 17740

RISK FREE POLICY

In-stock item shipped within 24 hours of order. No deposit on C.O.D. orders. Free shipping on prepaid cash orders within the Continental U.S. PA residents add sales tax. APO, FPO, and International orders add $5.00 plus 3% for priority mail service. Advertised prices show 4% discount for cash, add 4% for Master Card or Visa. Personal checks require 4 weeks clearance before shipping. All items subject to change without notice.

The Fastext-80, a dot-matrix printer from Smith-Corona, prints 80 characters per second.

and *Learn to Type*, a typing program, have been released for the IBM PC and PCjr computers.

Ultra-File retails for $195.00, *Property Management* for $495.00, and *Learn to Type* for $39.95. Educational versions of each program retail for $29.95.

Arrays, Inc./Continental Software
11223 South Hindry Ave.
Los Angeles, CA 90045

Dot-Matrix Printer

Smith-Corona has introduced the Fastext-80, a $259 dot-matrix printer which prints 80 characters per second.

Other features of the printer include bidirectional printing, six-pitch capability, full-line buffer, standard Centronics parallel interface, friction feed, self-test switch, and true descenders.

The printer, which prints ten characters per inch, uses a drop-in ribbon with a projected life span of one million characters.

Smith-Corona
65 Locust Ave.
New Canaan, CT 06840

Apple Word Processing Program

Apple Computer, Inc. has announced an enhanced version of its *Apple Writer II* word processing program.

New features include: horizontal scrolling; text display that shows the page and line count without having to print the document; built-in terminal mode that allows access to information services such as CompuServe and The Source; and a utility that enables users who do not have a ProDOS user's disk to format a blank disk for use with the program.

The enhanced version also includes a training disk, mail merge option, and built-in word processing language.

Data files created with previous DOS 3.3-based versions of *Apple Writer* can be converted to the ProDOS format by using a conversion utility on the ProDOS user's disk.

Apple Writer II retails for

ATARI

600XL ..... CALL

800XL ..... CALL

DISK DRIVES

Rana 1000	$239
Astra 2001	$549
Indus GT	$298
Trak AT-D2	$329
Trak AT-1	$319
Trak AT-D4	Call
Astra 1620 (Dual)	$499
Percom	Call
Atari 1050	$249

MEMORIES

Microbits 64K (XL)	$115
Mosaic 48K (400)	$98
Mosaic 64K (400/800)	Call
Mosaic 32K	$68
Atari 64K (600XL)	Call

INTERFACES

Axiom 846	Call
Ape Face	Call
Atari 850 (In Stock)	$169
Interfast 1	$150
Microbits 1150	Call
R-Verter	Call

DIRECT PRINTERS

Axiom AT-100	$195
Atari 1027	$269
Axiom 550 AT	$259
Axiom 700 AT	$469
Atari 1025	Call

DIRECT MODEMS

Microbits 1000C	$109

OTHER ATARI

400 Keyboard	Call
Koala Pad	$67
Chalkboard Pad	$75
Bit-3 80 Column	$228
ATR-8000 (64K)	$489
ATR-8000 (16K)	$359
Alien Voice Box	$98
1010 Recorder	$55

ATARI SOFTWARE

MISCELLANEOUS

Syn Calc (D)	$48
Syn File (D)	$48
Syn Trend (D)	$48
Syn Com (D)	$29
Syn Chron (D)	$29
Decathlon (R)	$29
Drols (D)	$23
Gyruss (R)	$31
Heist (D)	$23
Bruce Lee (C/D)	$27
Universe (D)	Call
Questron (D)	$34
Koala Coloring I	$20
Koala Logo Design	$27
Bumble Games (D)	$27
Miles Accounting	Call
World Gtst. Baseball	$23
Gridrunner (R)	$20
Sargon II (C/D)	$23
Millionaire (D)	$34
Castle Wolfenstein (D)	$20
Odesta Chess (D)	$46
Financial Wizard (D)	$41
Ultima III (D)	$39

ADVENTURE INT'L

Ultra Disassembler (D)	$33
Diskey (D)	$33
Adv. 1-12 (each) (C)	$18
Saga 1-12 (each) (D)	$27

ATARI

Atari Writer (R)	$68
Paint (D)	$30
Microsoft Basic II (R)	$64
Visicalc (D)	$139
Home File Mgr (D)	$36
Assembler Editor (R)	$44
Qix (R)	$32
Dig Dug (R)	$32
Atari Logo (R)	$72
Ms. Pac Man (R)	$33
Joust (R)	$33
Donkey Kong Jr. (R)	$35
Computer Chess (R)	$24

AVALON HILL

Telengard (C)	$16 (D) $19
Close Assault (C)	20 (D) 23
TAC (D)	$27

BRODERBUND

Arcade Machine (D)	$39
Bank St. Writer (D)	$43
Oper. Whirlwind (D)	$27
Choplifter (D)	$23 (R) $29

CBS SOFTWARE .... Call

CONTINENTAL

Home Accountant (D)	$44
Tax Advantage (D)	$45

COUNTERPOINT SW Call

DATASOFT

Pooyan (C/D)	$20
Teletalk (D)	$33
Graphic Master (D)	$23
Micropainter (D)	$23
Zaxxon (C/D)	$27

EASTERN HOUSE

Monkey Wrench II	$51

EDUCATIONAL SW

Tricky 1,2,3 or 4	$15
Tricky 5-13	$22

EPYX

Dragon Riders (C/D)	$27
Temple APS (C/D)	$27
Jumpman (C/D)	$27

FIRST STAR

Boulder Dsh (C/D)	20 (R) 27
Bristles (C/D)	$20
Flip Flop (C/D)	$20

GAMESTAR

Football (C/D)	$21
Baseball (C/D)	$21

INFOCOM

Zork I, II or III (D)	$27
Deadline (D)	$34
Starcross (D)	$27
Suspended (D)	$34
Witness (D)	$34
Planetfall (D)	$34
Enchanter (D)	$34
Infidel (D)	$34

KRELL SAT ..... Call

INTELL. STATEMENTS

Prof. Blackjack (D)	$46

LJK

Letter Perfect (D)	$58
Data Perfect (D)	$74
Spell Perfect (D)	$56
Letter Perfect (R)	$74

MICROPROSE

Solo Flight (D)	$23
Hellcat Ace (C/D)	$20

MONARCH

ABC Compiler (D)	$55

OPTIMIZED SYSTEMS

Action (R)	$65
Basic XL (R)	$65
Mac 65 (D)	$58
C-65 (D)	$58
Bug 65 (D)	$23

PARKER BROS

Astrochase (R)	$33
Death Star (R)	$33
Q-Bert (R)	$33
Popeye (R)	$33

QUALITY SW

Return of Hercules (D)	$22
Ali Baba (D)	$22

RESTON

Moviemaker (D)	$45

SCARBOROUGH SYS.

Mastertype (D/R)	$25
Songwriter (D)	$25

SCHOOL WIZWARE . Call

SIERRA ON-LINE

Homeword (D)	$46
Ultima II (D)	$39
Dark Crystal (D)	$26
Wiz. & Princess (D)	$22

SPINNAKER

Snooper Troop 1,2 (D)	$30
Most Amazing (D)	$27
Kids on Keys (D)	$20
Trains (D)	$27
Delta Drawing (R)	$27
Aerobics (D)	$34

STRATEGIC SIM.

Broadsides (D)	$27
Carrier Force (D)	$39
Combat Leader (D)	$27
Rails West (D)	$27
Epidemic (D)	$23
Eagles (D)	$27
Cosmic Bal I or II (D)	$27

SUBLOGIC

Flight Simulator II (D)	$36
Pinball (C/D)	$20

SYNAPSE

File Manager (R)	$54
Fort Apocalypse (C/D)	$23
Dimension X (C/D)	$23
Blue Max (C/D)	$23
Encounter (D/R)	$23
Zepplin (C/D)	$23
Pharoah's Curse (C/D)	$23

TRONIX

S.A.M. (D)	$39
P.M. Animator (D)	$29
Juice (C/D)	$20
Chatterbee (D)	$27

Printers/Etc.

DIABLO

630 Letter Qual.	$1559

SILVER REED

EXP 400 Ltr. Qual.	$288
EXP 500 Ltr. Qual.	$355
EXP 550 Ltr. Qual.	$419
EXP 770 Ltr. Qual.	$849

STAR

Gemini 10X	$239
Gemini 15X	$355
Delta 10	$339
Delta 15	$449
Radix 10	$498
Radix 15	$588
Power Type	$319

TOSHIBA

1340	$739
1351	$1249

NEC

3510	$1215
3530	$1215
3550	$1498
7710/7730	$1648

CITOH

Prowriter	$309
Prowriter II	$498
Starwriter	$909
Printmaster	$1189

OKIDATA

82A	$298
84P	$669
92	$368
93	$579

DAISYWRITER

2000	$985

MANNESMANN

160L	$559
Spirit	$267

JUKI

6100	$389
6300	Call

PANASONIC

1090	$228
1091	$288
1092	$439

MONITORS

AMDEK

V300 G	$119
V300 A	$139
V310 G (IBM)	$155
V310 A (IBM)	$159
Color I+	$269
Color II+	$399
Color III	$349
Color IV (IBM)	$699

PRINCETON GRAPHICS

MAX 12 (Amber)	$178
HX 12 (RGB)	$449
SR 12 (RGB)	$595

SUPER 5

100A (Amber)	$99
500G (IBM with tilt)	$126
500A (IBM with tilt)	$126

SAKATA

SC100 (Color)	$239
1000G (Green)	$99

TAXAN

100 Green	$115
105 Amber	$125
210 RGB/Composite	$259
400 RGB Med-Res.	$296
415 RGB Hi-Res.	$429
420 RGB Hi-Res (IBM)	$449

NEC

JB 1260 (Grn)	$99
JB 1201 (Grn)	$145
JB 1205 (Amber)	$145

ZENITH

Green	$85
Amber	$95

MODEMS

NOVATION

J-Cat	$99
Apple Cat II	$259
D-Cat	$149

HAYES

Smartmodem	$199
Smartmodem 1200	$469
Micromodem II	$259
Micromodem IIe	$225

PROMETHIUS

Promodem 1200	$329

COSMIC COMPUTERS

727 BREA CANYON RD., SUITE 16
WALNUT, CA 91789

OPEN MON-SAT

(800) 626-7642

PLEASE FOR ORDERS ONLY
SORRY, NO COD'S

IN CALIF. (714) 594-5204

FOR TECHNICAL INFO, ORDER INQUIRIES,

Add $2.50 shipping per software order in continental U.S. Add $5.00 shipping per software order for AK, HI, FPO-APO. Add $10.00 or 15% (whichever is greater) per software order for non-U.S. Call for cost of hardware shipping. Calif. residents add 6½% sales tax. Cashiers checks or money orders filled within 24 hours for items in stock. Personal checks require 4 weeks to clear. MasterCard and Visa OK for software only within continental U.S., add 3% surcharge. Include card no., expiration date and signature. Due to our low prices, all sales are final. All defective returns must have a return authorization number. Please call to obtain one before returning goods for replacement or repair. Prices & availability subject to change.

COMMODORE

INTERFACES

The Connection	$85
Bus Card	$138
Cardco G+	$65
Cardco B	Call
MSD (IEEE)	$98
Cardco 5 Slot	$48
Grappler C D	$109

DIRECT MODEMS

Hesmodem	$53
1650 Automodem	$99
1600 Modem	Call
Westridge Modem	Call

80 COLUMN BDS

Batteries 80 Col	$138
Video Pak 80	$129

DISK DRIVES

MSD (170K)	$309
MSD (Dual) (170Kx2)	$498
Indus GT	Call
Laser (170K)	$325
Commodore 1541	$239

RECORDERS

Cardco Recorder	$48
1530 Commodore	Call
Cassette Interface	$29
Phonemark Rec	$37

DIRECT PRINTERS

MPS 801	$219
Commodore 1526	$288
Cardco LQ/I	$498
1520 Color Printer	$129

64 SOFTWARE 64

MISCELLANEOUS

MAE Assembler (D)	$47
VIP Terminal (D)	$38
Star Wars (R)	$33
Super Base 64 (D)	$68
Doodle City (D)	$27
Summer Games (D)	$27
Pitfall II (D)	$25
Decathlon (D)	$25
50 Mission Crush (D)	$27
IFR. (C/D)	$20
Master Composer (D)	$27
Donkey Kong (R)	$29
Bruce Lee (D)	$23
Pro Football Stat. (D)	$56
Seastalker (D)	$27
Koala Coloring I	$20
Koala Logo Design	$27
Rockys Boots (D)	$33
Bumble Games (D)	$27
Beyond Wolfenstein	$23
Peachtree Account	Call
Odesta Chess (D)	$46
Ultima III (D)	$39
Prof. Blackjack (D)	$46
Homeword (D)	$46
Pers. Accountant (D)	$23
Karate Devils (D)	$27
Final Flight (D)	$22
Diskey (D)	$33
Barrons Sat (D)	$59
Millionaire (D)	$39
Sargon II (D)	$23
B-Graph (D)	$59
Castle Wolfenstein (D)	$20

ACCESS SOFTWARE

Beached (D)	$24
Neutral Zone (C/D)	$24
Spritemaster (D)	$25

AVALON HILL

Nuke War (C)	$12
Androm. Conquest (C)	$14
Midway Campaign (C)	$13
Computer Football (C)	$13
Telengard (C)	$16 (D) $19
Flying Ace (C)	$15
Moon Patrol (C)	$17

BATTERIES INCLUDED

Paper Clip (D)	$59
Consultant (D)	$64
Paper Clip w/Spell (D)	$79
Spell Pack (D)	$34
Organizer Series (Ea)	$22

BLUE SKY

Calc Result Adv.	$99
Calc Result Easy	$57

BRODERBUND

Bank St. Writer (D)	$43
Operat. Whrlwnd (D)	$27
Choplifter (R)	$27
Lode Runner (D)	23 (R) 27

CBS SOFTWARE

Success with Math (D)	$17
Wbstr Word Game (D)	$20

COMMODORE

Simons Basic (R)	$29
Magic Desk (R)	$48
Logo (D)	$50
Assembler 64 (D)	$36
Easy Script 64 (D)	$52

CONTINENTAL S.W.

Home Accountant (D)	$44
Tax Advantage (D)	$45
FCM (D)	$34

COUNTER POINT SW

Early Games (Ea)	$20

CREATIVE SOFTWARE

Moondust (R)	$23
Save New York (R)	$23
Pipes (R)	$23
Household Finance (C)	$20

DATASOFT

Pooyan (C/D)	$20
Moon Shuttle (D)	$20

ENTECH

Studio 64 (C/D)	$28
Database 64 (D)	$45

EPYX

Temple of APS (C/D)	$27
Jumpman (C/D)	$27
Dragonriders (C/D)	$27
Gateway to APS (R)	$27

FUTURE HOUSE

Comp. Pers. Account.	$56

HES

Omniwriter (D)	$45
Hesmon (D)	$27
64 Forth (R)	$39
Multiplan (D)	$65
Turtle Graphics (R)	$39

INFOCOM

Planet Fall (D)	$34
Enchanter (D)	$34
Infidel (D)	$34

JINSAM

Mini Jini (R)	$75

MICROSOFTWARE INT'L

Practicalc 64	(C) 34 (D) 36
Spreadsheet	(C) 49 (D) 52
Practifile 64 (D)	$36

MIRAGE CONCEPTS

Data Base (D)	$68
Word Processor (D)	$68

MICROSPEC

Database (D)	$44
Mailing List (D)	$32
Checkbook Mgr (D)	$47
G/L (D)	$44
A/P (D)	$47
A/R (D)	$47

PARKER BROS

Q-Bert (R)	$33
Popeye (R)	$33
Frogger (R)	$33

PROFESSIONAL SW

Wordpro W/Spell (D)	$68
Spellright (D)	$45

QUIKTEX

Quick Br. Fox (R)	$49

RAINBOW

File Assistant (D)	$46
Writers Assistant (D)	$46
Spreadsht Assist. (D)	$56

SCARBOROUGH

Mastertype (D/R)	$27
Song Writer (D)	$27

SOFTSMITH

Touch Typing (C/D)	$21

SOUTHERN SOLUTIONS

Businessman (D)	$48
Bill Payer (D)	$48
Bill Collector (D)	$48
Paymaster (D)	$48

SPINNAKER

Snooper 1 or 2 (D)	$27
Aerobics (D)	$34
Kids on Keys (D)	$20
Most Amazing (D)	$27
Kindercomp (D)	$20
Alphabet Zoo (D)	20
Trains (D)	$27
Delta Drawing (R)	$27

SUBLOGIC

Flight Simulator II (D)	$36
Pinball (C/D)	$22

SYNAPSE

Ft. Apocalypse (C/D)	$23
Necromancer (C/D)	$23
Zaxxon (D)	$23
Blue Max (D)	$23
Pharoah's Curse (C/D)	$23

SSI

Combat Leader (C/D)	$27
Computer Baseball (D)	$27
Eagles (D)	$27
Ringside Seat (D)	$27
Tigers In Snow (C/D)	$27
Battle Normandy (C/D)	$27

TIMEWORKS

Dungeons of Alg. (C/D)	$17
Robbers Lost (C/D)	$17
Money Mgr. (C/D)	$17
Wall Street (C/D)	$17
Data Manager (C/D)	$17
Elec. Checkbook (C/D)	$17

TOTL

Totl Text	(C) $32 (D) $34
Label	(C) $15 (D) $17
Time Mgr.	(C) $24 (D) $27
Rsrch Asst.	(C) $24 (D) $27

TRONIX

S.A.M. (D)	$39
Juice (D)	$23
Chatterbee (D)	$27

$149. Until February 1, 1985, Apple IIe users can upgrade their software to version 2.0 by sending their master disk, manual cover, and $50 to:

Apple Computer, Inc.
Apple Writer II Upgrade
P.O. Box 306
Half Moon Bar, CA 94019.

Test-Writing Programs For Apple, 64

Southern Oregon Video Enterprises, Inc. has introduced *SOVE Test Writer* and *SOVE Tester*, two test-writing programs for educators, for Apple and Commodore 64 computers.

Test Writer is menu-driven and allows teachers to format questions and store them in files. Up to 250 questions per file can be stored, and up to eight files per disk can be formatted.

Files generated by *Test Writer* can be used by *Tester*, a program which allows the generation of self-tests on a given topic with questions selected by the instructor.

Suggested retail price for *Test Writer* is $99.95. *Tester* retails for $39.95.

Computer Division
Southern Oregon Video Enterprises, Inc.
P.O. Box 400
Ashland, OR 97520

Atari 600XL Memory Expansion

An expansion module which can add up to 64K of memory to Atari 600XL computers has been announced by RC Systems, Inc.

The module plugs directly into the back of the computer and will not interfere with program cartridges. The AM64 is compatible with the *Atari Translator Disk.*

Model AM2 adds 32K of memory and retails for $79.95; the AM1 adds 48K and has a suggested price of $99.95. The AM64, which increases the memory by 64K, retails for $119.95.

RC Systmes, Inc.
121 West Winesap Rd.
Bothell, WA 98012

Commodore 64 Break Dancing

Break Street, a break dancing computer game for the Commodore 64, has been introduced by Creative Software.

As a break dancer, you compete against the Stingrays, a neighborhood gang, in head spins, moonwalks, snaking, and the tut dance moves. Each break dance has its own level of difficulty. For example, if you miss a key sequence move, your character will fall, turning the action over to the Stingrays. Entire dance sequences may be strung together, recorded, and replayed later.

The game is controlled by either the keyboard or a joystick and sells for a suggested $24.95 on disk.

Creative Software
230 East Caribbean Dr.
Sunnyvale, CA 94089

New Product releases are selected from submissions for reasons of timeliness, available space, and general interest to our readers. We regret that we are unable to select all new product submissions for publication. Readers should be aware that we present here some edited version of material submitted by vendors and are unable to vouch for its accuracy at time of publication. ©

Now, the lowest prices ever on 3M Scotch® DISKETTES

$1.53 ea. 5¼" SSDD QTY. 20 — **$2.06** ea. 5¼" DSDD QTY. 20

5¼" SSDD-96TPI → $2.29 ea. 5¼" DSDD-96TPI → $2.85 ea.
SOFT SECTOR ONLY! MINIMUM ORDER: 20 DISKETTES

These are factory-fresh 3M diskettes packed in boxes of 10 with Tyvek sleeves, reinforced hubs, identification labels and write-protect tabs.

3.5" MICRO-DISKETTES—SS-135 TPI → $2.89 ea.

LIFETIME WARRANTY ON ALL 3M SCOTCH DISKETTES!

FOR ORDERS ONLY: 1-800-621-6827 (In Illinois: 1-312-944-2788)
INFORMATION & INQUIRIES: 1-312-944-2788

HOURS: 8AM-5PM Central Time, Monday-Friday

WE WILL BEAT ANY NATIONALLY ADVERTISED PRICE ON THE SAME PRODUCTS AND QUANTITIES!

DISK WORLD!, Inc.
Suite 4806 • 30 East Huron Street • Chicago, Illinos 60611

DISK WORLD! Authorized Distributor Information Processing Products 3M

FANTASTIC LOW PRICES ON BASF QUALIMETRIC DISKETTES!

$1.41 ea. 5¼" SSDD Qty. 20 — **$1.59** ea. 5¼" DSDD Qty. 20

PACKED IN PLASTIC LIBRARY CASES!

BASF QUALIMETRIC DISKETTES have a LIFETIME WARRANTY and are packed in PLASTIC LIBRARY CASES with Tyvek sleeves, reinforced hubs, user identification labels and write-protect tabs.

SOFT SECTOR ONLY! MINIMUM ORDER: 20 DISKETTES

BASF 3.5" MICRO-FLOPPIES — SSDD-135 TPI →$3.02 ea.
BASF 5¼"HIGH DENSITY FOR IBM PC-AT — DSDD-HD →$4.72 ea.

FOR ORDERS ONLY: 1-800-621-6827 (In Illinois: 1-312-944-2788)
INFORMATION & INQUIRIES: 1-312-944-2788

HOURS: 8AM-5PM Central Time, Monday-Friday

WE WILL BEAT ANY NATIONALLY ADVERTISED PRICE ON THE SAME PRODUCTS AND QUANTITIES!

DISK WORLD! Authorized Reseller Information Processing Media BASF

Incredible value! Nashua™ Diskettes

LIFETIME WARRANTY!

$1.05 ea. 5¼" SSDD Qty. 50 — **$1.15** ea. 5¼" DSDD Qty. 50

These are poly-bagged diskettes packaged with Tyvek sleeves, reinforced hubs, user identification labels and write-protect tabs.

NASHUA Corporation is a half-billion dollar corporation and a recognized leader in magnetic media.

SOFT SECTOR ONLY! Sold in multiples of 50 only!

FOR ORDERS ONLY: 1-800-621-6827 (In Illinois: 1-312-944-2788)
INFORMATION & INQUIRIES: 1-312-944-2788

HOURS: 8AM-5PM Central Time, Monday-Friday

WE WILL BEAT ANY NATIONALLY ADVERTISED PRICE ON THE SAME PRODUCTS AND QUANTITIES!

DISK WORLD!, Inc.
Suite 4806 • 30 East Huron Street • Chicago, Illinos 60611

DISK WORLD! Authorized Distributor NASHUA MAGNETIC MEDIA

Save 20% to 60% Or More

On all your OFFICE & COMPUTER SUPPLIES!

Now, you can enjoy DISK WORLD! savings on more than 21,000 office and computer supply products! You name it, we got it...at tremendous savings.

Everything from Scotch™ Tape to Post-It Notes™ to paper clips and rubber bands...and thousands of computer products as well!

Our catalog is huge...more than 700 pages, listing more than 21,000 items.

We have to charge for it: $10.00 to be exact.

But we include a $50.00 worth of discount coupons that you can use on future orders.

Now, it's DISK WORLD! for every office or computer supply need...and always at tremendous savings!

This offer supercedes all prior catalog offers.
Not responsible for typographical errors.

FOR ORDERS ONLY: 1-800-621-6827 (In Illinois: 1-312-944-2788)
INFORMATION & INQUIRIES: 1-312-944-2788

HOURS: 8AM-5PM Central Time, Monday-Friday

WE WILL BEAT ANY NATIONALLY ADVERTISED PRICE ON THE SAME PRODUCTS AND QUANTITIES!

DISK WORLD!, Inc.
Suite 4806 • 30 East Huron Street • Chicago, Illinos 60611

DISK WORLD!

DISK WORLD! Ordering & Shipping Instructions

Shipping: 5¼" & 3.5" DISKETTES—Add $3.00 per each 100 or fewer diskettes. **Other Items:** Add shipping charges as shown in addition to other shipping charges. **Payment:** VISA and MASTERCARD accepted. **COD Orders:** Add additional $3.00 Special Handling charge. **APO, FPO, AK, HI & PR Orders:** Include shipping charges as shown **and** additional 2% of total order amount to cover PAL and insurance. **Taxes:** !"' nois residents only, add 8% sales tax.

Prices subject to change without notice.
This ad supercedes all other ads.
Not responsible for typographical errors.
MINIMUM TOTAL ORDER: $35.00

FOR ORDERS ONLY: 1-800-621-6827 (In Illinois: 1-312-944-2788)
INFORMATION & INQUIRIES: 1-312-944-2788

HOURS: 8AM-5PM Central Time
Monday-Friday

WE WILL BEAT ANY NATIONALLY ADVERTISED PRICE ON THE SAME PRODUCTS AND QUANTITIES!

DISK WORLD!, Inc.
Suite 4806 • 30 East Huron Street • Chicago, Illinos 60611

DISK WORLD!

ATHANA DISKETTES The great unknown!

99¢ ea. ← 5¼" SSDD Qty. 50 — **$1.09** ea. 5¼" DSDD → Qty. 50

You've used these diskettes hundreds of times...as copy-protected originals on some of the most popular software packages. They're packed in poly-bags of 25 with Tyvek sleeves, reinforced hubs, user identification labels and write-protect tabs.

LIFETIME WARRANTY!
SOFT SECTOR ONLY! Sold in multiples of 50 only.

FOR ORDERS ONLY: 1-800-621-6827 (In Illinois: 1-312-944-2788)
INFORMATION & INQUIRIES: 1-312-944-2788

HOURS: 8AM-5PM Central Time, Monday-Friday

WE WILL BEAT ANY NATIONALLY ADVERTISED PRICE ON THE SAME PRODUCTS AND QUANTITIES!

DISK WORLD!, Inc.
Suite 4806 • 30 East Huron Street • Chicago, Illinos 60611

DISK WORLD! Authorized Distributor ATHANA MAGNETIC MEDIA

DISKETTE STORAGE CASES

AMARAY MEDIA-MATE 50: A REVOLUTION IN DISKETTE STORAGE

Every once in a while, someone takes the simple and makes it elegant! This unit holds 50 5¼" diskettes, has grooves for easy stacking, inside nipples to keep diskettes from slipping and several other features. We like it!

$10.95 ea. + $2.00 Shpng.

DISKETTE 70 STORAGE: STILL A GREAT BUY.

Dust-free storage for 70 5¼" diskettes. Six dividers included. An excellent value.

$11.95 + $3.00 Shpng.

DISK CADDIES

The original flip-up holder for 10 5¼" diskettes. Beige or grey only.

$1.65 ea. + 20¢ Shpng.

FOR ORDERS ONLY: 1-800-621-6827 (In Illinois: 1-312-944-2788)
INFORMATION & INQUIRIES: 1-312-944-2788

HOURS: 8AM-5PM Central Time, Monday-Friday

WE WILL BEAT ANY NATIONALLY ADVERTISED PRICE ON THE SAME PRODUCTS AND QUANTITIES!

DISK WORLD!, Inc.
Suite 4806 • 30 East Huron Street • Chicago, Illinos 60611

DISK WORLD! The value leader in Computer supplies And accessories.

PRINTER RIBBONS: at extraordinary prices!

Brand new ribbons, manufactured to Original Equipment Manufacturer's specifications, in housings. (Not re-inked or spools only.)

LIFETIME WARRANTY!

Epson MX-70/80	**$3.58** ea.	+ 25¢ Shpng.
Epson MX-100	**$4.95** ea.	+ 25¢ Shpng.
Okidata Micro83	**$1.48** ea.	+ 25¢ Shpng.
Okidata Micro84	**$3.66** ea.	+ 25¢ Shpng.

FOR ORDERS ONLY: 1-800-621-6827 (In Illinois: 1-312-944-2788)
INFORMATION & INQUIRIES: 1-312-944-2788

HOURS: 8AM-5PM Central Time, Monday-Friday

WE WILL BEAT ANY NATIONALLY ADVERTISED PRICE ON THE SAME PRODUCTS AND QUANTITIES!

DISK WORLD!, Inc.
Suite 4806 • 30 East Huron Street • Chicago, Illinos 60611

DISK WORLD!

PAPER: Save 50% or more!

We buy paper by the TRUCKLOAD from the two biggest mills in the country.

Therefore, we charge a lot less than you've been paying!

For the complete DISK WORLD paper catalogs and price lists, call 1-312-944-2788 or write us. (Please do not use the "800" line for paper catalog requests, unless you are also placing an order.

Don't spend more than you have to for top-quality computer printer paper. Call DISK WORLD! today.

INFORMATION & INQUIRIES:
1-312-944-2788

HOURS: 8AM-5PM Central Time
Monday-Friday

WE WILL BEAT ANY NATIONALLY ADVERTISED PRICE ON THE SAME PRODUCTS AND QUANTITIES!

DISK WORLD!, Inc.
Suite 4806 • 30 East Huron Street • Chicago, Illinos 60611

DISK WORLD! Where paper Costs less!

Advertisers Index

Reader Service Number/Advertiser	Page
102 Abacus Software	91
103 Abacus Software	93
104 Abacus Software	95
AB Computers	67
105 American PEOPLE/LINK	87
106 Apropos Technology	4
Batteries Included	21
Batteries Included	57
107 Cardco, Inc.	IBC
108 CodeWriter	46
Commodore	BC
109 CompuServe	27
ComputAbility	64
110 Computer Mail Order	80,81
111 Computer Novelty Corp.	142
112 Cosmic Computers	157
113 Davidson & Associates	1
114 Discwasher	7
115 Disk World!	159
Eastman Kodak Company	2,3
Electronic Arts	41
Epyx	51
Epyx	53
Epyx	55
Fidelity Investors Xpress	108
Frontrunner Computer Industries	151
Happy Computers, Inc.	104
116 Harmony Video & Computers	142
IBM	14,15
117 Indus Systems	59
Infocom, Inc.	22,23
118 Inforunner Corporation	IFC
Jason-Ranheim	28
Jesse Jones Ind.	116
119 John Wiley & Sons, Inc.	54
120 J & R Music World	122
121 Krell Software Corp.	156
122 Legend Peripheral Products	107
123 Lyco Computer Marketing & Consultants	154,155
Maxell Corporation of America	11
124 MegaSoft Limited	89
125 Micro-W Distributing, Inc.	128
126 Micro-W Distributing, Inc.	133
127 Micro World Electronix, Inc.	122
128 Micro World Electronix, Inc.	142
129 Mimic Systems, Inc.	35
130 Mindscape, Inc.	25
131 Mindscape, Inc.	43
Navarone Industries, Inc.	156
132 Nibble Notch Computer Products	122
NRI Schools	83
133 Okidata	19
134 Pacific Exchanges	153
134 Pacific Exchanges	156
135 PE Systems, Inc.	158
136 Practical Programs	133
137 Protecto Enterprizes	74,75
137 Protecto Enterprizes	76,77
138 Quinsept, Inc.	46
139 Scarborough Systems	49
140 Sega Enterprises, Inc.	47
141 Smart Data Inc.	83
Spinnaker	9
142 Strategic Simulations, Inc.	29
143 subLOGIC Corporation	37
144 Tape World	158
Terin Software	158
145 Timeworks, Inc.	13

COMPUTE! Books New Releases	31
COMPUTE!'s First, Second & Third Book of Commodore 64	61
COMPUTE!'s Home Computer Wars	33
COMPUTE!'s Machine Language for Beginners & Second Book of Machine Language	63
COMPUTE! Subscription	17

IF YOU'RE A ☐ PARENT

☐ STUDENT ☐ TEACHER ☐ HOBBYIST

☐ INVESTOR ☐ THINKER ☐ GAMER

COMPUTE! IS FOR YOU!

SAVE 32%
OFF THE $35.40 COVER PRICE
☐ 1 YEAR (12 ISSUES) $24

SAVE 36%
OFF THE $70.80 COVER PRICE
☐ 2 YEARS (24 ISSUES) $45

NAME _____

ADDRESS _____ APT. _____

CITY/STATE/ZIP _____

☐ PAYMENT ENCLOSED ☐ PLEASE BILL ME ☐ CHECK HERE IF RENEWAL

PLEASE CHECK IF YOU OWN A 01☐ APPLE 02☐ ATARI 03☐ COMMODORE 64
04☐ VIC-20 05☐ IBM 06☐ TI-99/4A 99☐ OTHER _____
☐ DON'T YET HAVE ONE

PLEASE ALLOW 4-6 WEEKS FOR DELIVERY OF YOUR FIRST ISSUE. FOREIGN AND CANADIAN PLEASE ADD $6 (U.S.) PER YEAR POSTAGE. OFFER SUBJECT TO CHANGE WITHOUT NOTICE. 352101

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 2312 GREENSBORO, NC

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE!
P.O. Box 914
Farmingdale, NY 11737

I'd like to order my copy of *The Home Computer Wars*.

Please send ____ copies in hardback at $16.95 per copy $ _____
Add $2.00 per book shipping + handling _____
____ copies in paperback at $9.95 per copy _____
Add $2.00 per book shipping + handling _____
Sub-total _____
N.C. residents add 4.5% sales tax _____
Total payment enclosed $ _____

All orders must be prepaid (check, charge or money order).
All payments must be in U.S. funds.

□ Payment enclosed.
□ Charge my □ VISA □ MasterCard □ American Express

_____ _____/_____

Acct. No. Exp. Date

Name _____

Address _____

City _____ State _____ Zip _____

Please allow 4-6 weeks for delivery

*Please include $5.00 for overseas airmail.

452752C

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 2312 GREENSBORO, NC

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE! Books
P.O. Box 5406
Greensboro, NC 27403

COMPUTE!'s
FREE Reader Information Service

Use these cards to request FREE information about the products advertised in this issue. Clearly print or type your full name and address. Only one card should be used per person. Circle the numbers that correspond to the key number appearing in the advertisers index.

Send in the card and the advertisers will receive your inquiry. Although every effort is made to insure that only advertisers wishing to provide product information have reader service numbers, COMPUTE! cannot be responsible if advertisers do not provide literature to readers.

Please use these cards *only* for subscribing or for requesting product information. Editorial and customer service inquiries should be addressed to: COMPUTE!, P.O. Box 5406, Greensboro, NC 27403. Check the expiration date on the card to insure proper handling.

Use these cards and this address only for COMPUTE!'s Reader Information Service. Do not send with payment in any form.

COMPUTE!

101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117
118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134
135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151
152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168
169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185
186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202
203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236
237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253

Circle 101 for a one year new U.S. subscription to COMPUTE!: you will be billed for $24.

Please let us know. Do you own: plan to buy:

- ☐ 270 Apple ☐ 271
- ☐ 272 Atari ☐ 273
- ☐ 274 Commodore ☐ 275
- ☐ 276 IBM ☐ 277
- ☐ 278 TI-99/4A ☐ 279
- ☐ 280 Other (specify model) ☐ 281

Please print or type name and address.
Limit one card per person.

Name

Address

City

State/Province Zip

Country

Please include zip code. Expiration 4/30/85. C0285

SUBSCRIBE TO COMPUTE!

For Fastest Service, Call Our **Toll-Free** US Order Line
800-334-0868
In NC call 919-275-9809

My Computer Is:

01 ☐ Apple 02 ☐ Atari 03 ☐ Commodore 64
04 ☐ VIC-20 05 ☐ IBM 06 ☐ TI-99/4A
99 ☐ Other_____ ☐ Don't yet have one.

☐ $24.00 One Year US Subscription
☐ $45.00 Two Year US Subscription

(Readers outside of the US, please see our foreign readers subscription card or inquire for rates).

Name

Address

City State Zip

☐ Payment Enclosed ☐ Bill me
Charge my: ☐ VISA ☐ MasterCard ☐ American Express
Account No. Expires /

Your subscription will begin with the next available issue. Please allow 4-6 weeks for delivery of first issue. Subscription prices subject to change at any time.
The COMPUTE! subscriber list is made available to carefully screened organizations whose products or services may be of interest to our readers. If you prefer not to receive such mailings, please check this box ☐.

352104

Place
Stamp
Here

COMPUTE! Reader Service
P.O. Box 2141
Radnor, PA 19089

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 2312 GREENSBORO, NC

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE!
P.O. Box 914
Farmingdale, NY 11737

"The Complete CARDCO Line"

. . . and still growing!

CARDCO provides "Commodore-ready" computer accessories that will enhance your utilization of Commodore-64 and VIC-20 Computers, increase their capability, and add to your enjoyment and skill. AND, they're available for use with other personal computers, too.

Designed with the user in mind, CARDCO offers fine accessories including Printer Interfaces with and without graphics, Expansion Interfaces, Memory Expansions, Cassette Interfaces, Numeric Keypads PLUS "NOW" Software for your VIC-20 and C-64. These programs include the "WRITE NOW" Word Processor, "MAIL NOW" Mailing List, PRINTER UTILITY PROGRAMS on Tape and on Disk, "SPELL NOW" Spell Checker, "GRAPH NOW" including "PAINT NOW", and "FILE NOW".

CARDCO has three new Letter Quality PRINTERS with your choice of drumhead design (8 1/2" carriage), Daisy Wheel Design (13 inch carriage) and Daisy Wheel Design (11 inch carriage). "Commodore-ready" . . . plus; with compatible input for PC, PC jr., TRS-80 and many more personal computers. CARDCO's NEW "DATA CASSETTE RECORDER/PLAYER" is also "Commodore-ready" and ready for instant shipment at prices that will amaze you.

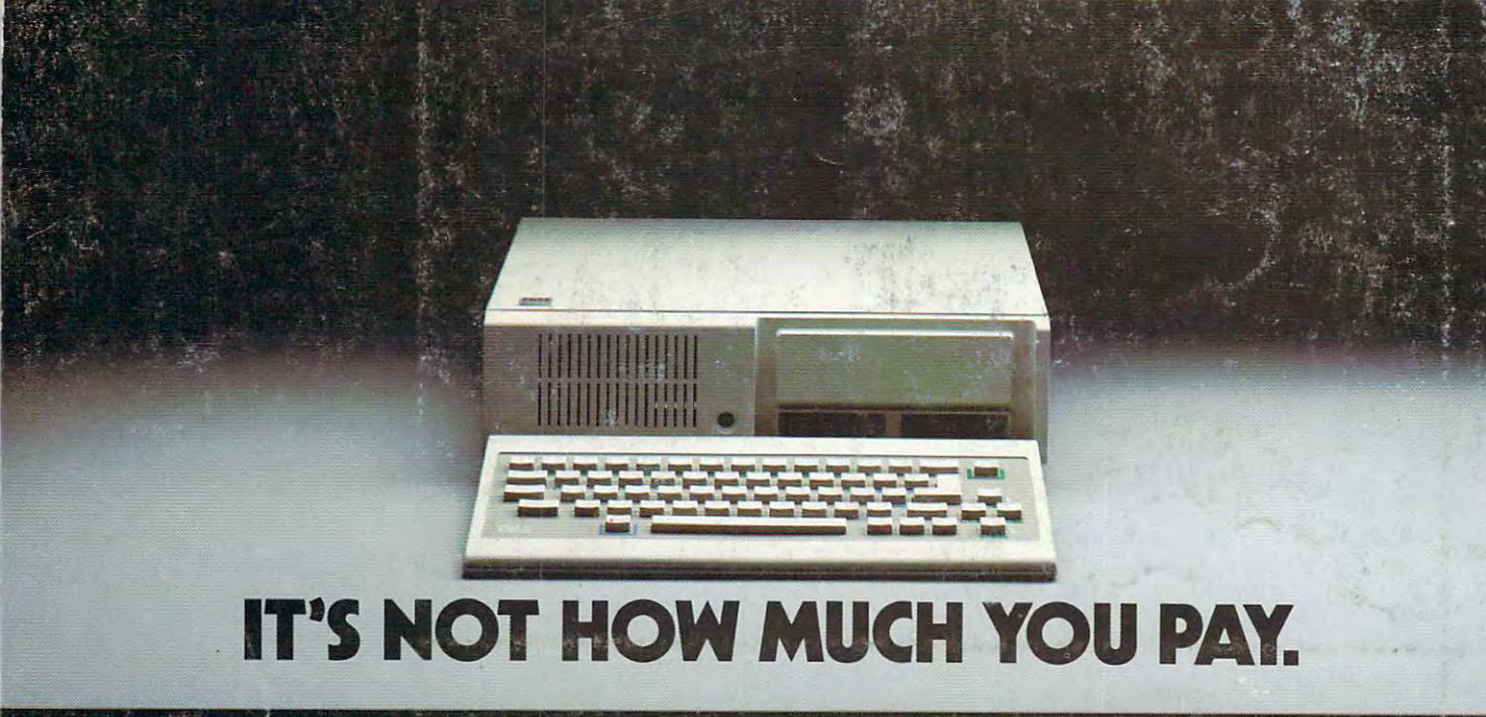
CARDCO will constantly increase its line with unique and new products to enhance the enjoyment of computer owners.

Write for illustrated literature and prices or see CARDCO Computer Accessories and Software wherever Computers are sold.

cardco, inc.

300 S. Topeka Wichita, Kansas 67202 (316) 267-6525

"The world's largest manufacturer of Commodore accessories."



IT'S NOT HOW MUCH YOU PAY.

IT'S HOW MUCH YOU GET.

The computer at the top has a 64K memory.

It has the initials I, B, and M. And you pay for those initials—about $669.

The Commodore 64™ has a 64K memory.

But you don't pay for the initials, you just pay for the computer: $215. About one third the price of the IBM PCjr.™

The Commodore 64 also has a typewriter-type keyboard with 66 typewriter-type keys. (Not rubber chicklet keys like the IBM PCjr.)

It has high resolution graphics with 320 x 200 pixel resolution, 16 available colors and eight 3-dimensional sprites.

It has 9-octave high fidelity sound.

The Commodore 64 is capable of running thousands of programs for home and office. And if you add a printer or color monitor, disk drive and a modem—all together it just about equals the price of the IBM PCjr all alone. With no peripherals.

So you can buy a computer for a lot of money.

Or buy a lot of computer for the money.

COMMODORE 64

IT'S NOT HOW LITTLE IT COSTS,
IT'S HOW MUCH YOU GET.

www.commodore.ca